Appendix

A Related Work

Global Planning Search-based planning algorithms, such as A* [**?** 1, 2], discretize the state space and perform a graph search to find an optimal path. While the graph search can be fast, complete, and guaranteed optimal, the requirement to construct a discrete graph hinders these algorithms in continuous spaces and for novel problems not well covered by the current graph. Sampling-based planners [3] function in a continuous state space by drawing samples and building a tree. When the tree has sufficient coverage of the planning problem, the algorithm traverses the tree to produce the final plan. Sampling-based planners are continuous, probabilistically complete, *i.e.* find a solution with probability 1, and some are even *asymptotically optimal* [4, 5, 6], but under practical time limitations, their random nature can produce erratic—though valid—paths.

Both of the aforementioned planner types are designed to optimize for path length in the given state space (*e.g.* configuration space) while avoiding collisions. An optimal path in configuration space is not necessarily optimal for the end effector in cartesian space. Human motion tends to minimize hand distance traveled [7], so what appears optimal for the algorithm may be unintuitive for a human partner or operator. In the manipulation domain, goals are typically represented in end effector task space [8, 9]. In a closed loop setting with a moving target, the traditional process of using IK to map task to configuration space can produce highly variable configurations, especially around obstacles. Motion Optimization [10, 11, 12] on the other hand, generates paths with non-linear optimization and can consider multiple objectives such as smoothness of the motion, obstacle avoidance, and convergence to an end effector pose. These algorithms require careful tuning of the respective cost functions to ensure convergence to a desirable path and are prone to local minima. Furthermore, non-linear optimization is computationally complex and can be slow for difficult planning problems.

Imitation Learning Inverse Reinforcement Learning [13, 14, 15] typically assumes expert optimality and learns a cost function accordingly, whereas Behavior Cloning [16, 17] directly learns the state-action mapping from demonstrations, regardless of the expert's optimality. We thus employ behavior cloning because producing optimal plans for continuous manipulation problems is challenging. Recent work demonstrates behavior cloning's efficacy for fine-grained manipulation tasks, such as chopstick use [18] and pick-and-place [19]. For long-horizon tasks like ours, however, distributional shift and data variance can hinder behavior cloning performance. Distribution shift during execution can lead to states unseen in training data [18]. Complex tasks often have a long tail of possible action states that are underrepresented in the data, leading to high data variance [20]. There are many techniques to address these challenges through randomization, noise injection, regret optimization, and expert correction [18, 21, 22, 23, 24]. These techniques, however, have not been demonstrated on a problem of our scale and complexity (see Appendix E for details on the range of data).

B Failure Modes Across All Test Sets

In the main paper, we presented the breakdown of the failure modes on the set of problems solvable by both the global and hybrid planners. In this section, we present the failure modes separately across the two test sets. The *Global Planner*-solvable test set is consistently the hardest for all methods, having the highest collision rates and target error. While STORM and Fabrics both see significant increases in target error, the change in collision rate is minor. When trained with the *Global Expert*, MπNets has the highest collision rate across all test sets, yet it also has the most consistent rollout accuracy. We attribute the collision rate to inconsistency in the *Global Planner's* motion and the rollout accuracy to the high coverage of the problem space. When evaluated on the *Global Planner*-solvable test set, MπNets trained with the *Hybrid Expert* also has its highest collision rate. We attribute this to distribution shift in the problem space.

C Expert Pipelines

We present more details of our planning pipeline in this section.

	% Env. Coll.	% Self Coll.	% Jnt Viol.	% Within 1cm	% Within 5cm	% Within 15°	% Within 30°
G. Fabrics [25]	8.17	0.00	0.39	68.56	73.33	82.06	84.00
STORM [26]	0.39	0.11	0.28	83.11	85.33	90.00	91.61
MπNets (Ours)							
Hybrid Expert	0.89	0.00	0.00	98.83	99.61	98.83	99.28
Global Expert	15.94	0.00	0.00	99.00	99.83	97.06	99.28

Table 1: Failure Modes on problems solvable by the hybrid planner

	% Env. Coll.	% Self Coll.	% Jnt Viol.	% Within 1cm	% Within 5cm	% Within 15°	% Within 30°
G. Fabrics [25]	7.83	0.50	0.33	45.67	57.33	74.39	78.22
STORM [26]	1.94	0.11	0.28	71.33	78.22	64.44	72.67
MπNets (Ours)							
Hybrid Expert	11.00	0.78	0.00	87.72	93.17	84.56	88.56
Global Expert	21.94	0.00	0.00	97.94	99.50	96.56	99.22

Table 2: Failure Modes on problems solvable by the global planner

Global Planner is composed of widely used off-the-shelf components. We first use inverse kinematics to convert our task space goals to configuration space, followed by AIT* [27] in configuration space, and finally, spline-based, collision-aware trajectory smoothing [28]. We use IKFast [29] for inverse kinematics, OMPL [30] for AIT*, and Pybullet Planning for the smoothing implementation [31]. To manage the compute load when generating a large dataset of trajectories, we employed a time-out with AIT* of 20 seconds.

Hybrid Expert is designed to produce consistent motion in task space. We start by using AIT* [27] with a 2 second timeout to plan for a floating end effector, *i.e.* one not attached to a robot arm, and then use Geometric Fabrics [25] to follow the path. Geometric Fabrics are deterministic and geometrically consistent [25] local controllers, but they struggle to solve the problems in our dataset without assistance from a global planner. Geometric Fabrics are highly local, and even with dense waypoints given by a global planner, they can run into local minima, which in turn generate trajectories with highly variable velocity. We use a combination of spline-based smoothing and downsampling [32] to create a consistent configuration space velocity profile across our dataset.

Consistency We use the term *consistency* to describe a qualitative characteristic of a planner and its learnability. Specifically, we use it to describe two quantities: 1) expert quality and 2) repeatability of the planner. Mandlekar et al. [19] demonstrate how Imitation Learning performance varies depending on expert quality. Among the metrics they use to describe expert quality, they demonstrate the importance of expert trajectory length. MπNets employs task-space goals, and the *Hybrid Planner* produces shorter task-space paths. Across our test dataset of global and hybrid solvable problems, the *Hybrid Planner*'s end effector paths average 57cm ± 31cm and the total orientation distance traveled is 95° ± 52°. Meanwhile, the *Global Planner*'s paths average 61cm ± 39cm and 113° ± 55°, respectively.

Repeatable input-output datasets are important for deep learning systems. Prior works have shown that deep learning systems deteriorate or require more data when using noisy labels [33, 34]. Both the *Global Planner* and *Hybrid Planner* are sampling-based planners and do not produce repeatable paths by their very nature. Yet, the *Hybrid Planner* uses sampling to plan in a lower-dimensional state space—6D pose space—while the *Global Planner* samples in 7D configuration space. We use a naive sampler, so the lower dimensionality of the *Hybrid Planner's* sampler implies that its typical convergence rate will be faster. After planning, the *Hybrid Planner* employs Geometric Fabrics [25] to follow the task-space trajectory. Geometric Fabrics are deterministic, which further promotes repeatability in the final, configuration space trajectories. Meanwhile, the *Global Planner* uses a randomized smoothing algorithm that is not deterministic. Taking these individual components together, we expect the *Hybrid Planner's* solutions on similar problems to be typically more alike than the *Global Planner's* solutions to the same problems.

D Network Architecture

Our PointNet++ architecture has three set abstraction groups followed by three fully connected layers. The first set abstraction layer performs iterative furthest point sampling to construct a set of 512 points, then it does a grouping query within 5cm of at most 128 points. Finally, there is a local PointNet [35] made up of layers of size 4, 64, 64, 64 respectively. The second set abstraction is lower resolution, sampling 128 furthest points and then grouping at most 128 points within a 30cm radius. The corresponding PointNet is made up of layers of size 64, 128, 128, and 256 respectively. Our third set abstraction layer skips the furthest point sampling, groups all points together, and uses a final PointNet with layers of size 256, 512, 512, 1,024 respectively. Finally, after the set abstraction layers, we have three fully connected layers with 4,096, 4,096, and 2,048 dimensions respectively. In between these layers, we use group norm and Leaky ReLU.

The output of our point cloud encoder is a 2,048 dimensional embedding. The robot configuration encoder and the displacement decoder are both fully connected multilayer perceptrons with Leaky ReLU activation functions [36]. The robot configuration encoder maps our 7-dimensional input to a 64-dimensional output and has four hidden layers with 32, 64, 128, and 128 dimensions respectively. The displacement decoder maps the combined embeddings from the point cloud and robot configuration encoders, which together have 2,112 dimensions, to the 7 dimensional normalized displacement space. The decoder has three hidden layers with 512, 256, and 128 dimensions respectively. Our entire architecture together has 19 million parameters.

E Data Generation Pipeline

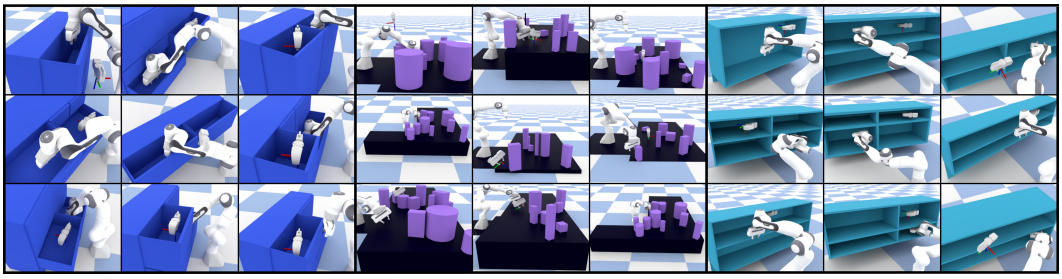

Figure 1: MπNets is trained with a dataset consisting of solutions to 3.27 million unique planning problems across over 575,000 unique, procedurally generated environments.

We used the same procedural data generation pipeline to generate data for training as well as inference test problems. We will be releasing the code to generate the data alongside our generated data sets upon publication.

Tabletop The dimensions of the table, including height, are randomized, as well as whether the table has an L-bend around the robot. The table itself is always axis-oriented. Table height ranges from 0 to 40cm. Table edges are chosen independently, *e.g.* the maximum x value for a table is chosen from a uniform distribution, and the center of the tables is not fixed. The front table can range between 90 and 110cm deep and between 205 and 240cm wide. When there is an L-bend, the side table ranges from 90 to 247.5cm deep and 42.5 to 72.5cm wide. After generating the table, a random assortment of boxes and cylinders are placed on the table facing upward, *i.e.* cylinders are on their flat edge. There are between 3 and 15 objects in each scene. These objects are between 5 and 35cm tall. The side dimensions of the boxes, as well as the radius of the cylinders, are between 5 and 15cm.

Cubby The dimensions of the cubby, the wall thickness, the number of cubbies, and the orientation of the entire fixture are randomized. We start by constructing a two-by-two cubby and then modify it to randomize the number of cubby holes. The wall thickness is chosen to be between 1 and 2cm. Similar to the tabletop, cubby edges are chosen independently, which implicitly set the center. The overall fixture is ranges from 120 to 160cm wide, 20 to 35cm deep, and between 30 and 60cm tall. The horizontal and vertical center dividers are then placed randomly within a 20cm range. Finally, we apply a random yaw rotation of up to 40°around the fixture's central axis. For roughly half of the cubby environments, we modify the cubby to reduce the number of cubby holes. To do this, we

select two random, collision-free robot configurations in two separate cubby holes and then merge the cubby holes necessary to create a collision-free path between them.

Dresser The dimensions of the dresser, the placement of the drawers, and the orientation of the entire fixture are randomized. The dresser side walls, drawer side walls, and drawer faces are always 1, 1.9, and 0.4cm thick respectively. Unlike the other two environments, the dimensions for the dresser are chosen randomly, as is the center point for the fixture. The dresser dimensions range from 80 to 120cm wide, 20 to 40cm deep, and 55 to 85cm tall. The dresser is always placed on the ground randomly in the reachable space of the robot, with a random orientation around its central yaw axis. We next construct the drawers. We randomly choose a direction in which to split the dresser and then split it into two drawers. We perform this recursively within each drawer, stopping according to a decaying probability function. Finally, we open two drawers within reachable space.

Initial Configurations and Target Poses After generating a random fixture, we search for valid start and goal configurations. We first look for target poses with reasonable orientations–in a grasping pose for the tabletop, pointing approximately inward for a cubby, or pointing approximately downward in a drawer. We choose pairs of these targets, solve for a collision-free inverse kinematics solution for each target, and consider these configuration space solutions to be candidates for the start or end of a trajectory. We also add a set of collision-free neutral configurations to the mix. These neutral configurations are generated by adding uniform randomness to a fixed neutral configuration. From this set of task-space targets and corresponding collision-free configuration space solutions, we select pairs to represent a single planning problem. For each pair selected, we use the *Global Planner* to verify that a smooth, collision-free planning solution exists.

F Training MπNets

MπNets is trained for single-step prediction, but during inference, we use it recursively for closed-loop rollouts. The compounded noise in subsequent inputs equates covariate shift [22, 24]. To promote robustness, we augment our training data with random perturbations in two ways. We apply Gaussian noise to the joint angles of each input configuration, which in turn affects the corresponding points in the point cloud, passed as input to the network [18, 37]. Additionally, for each training example, we generate a unique point cloud during training, *i.e.* during each epoch, the network sees 163.5M unique point clouds. We train our network with a single set of weights across our entire dataset.

We implemented MπNets in PyTorch and used the Adam optimizer with a learning rate of 0.0004. We trained it across 8 NVIDIA Tesla V100 GPUs for a week.

G Inference with MπNets

We used separate inference hardware for our simulated experiments and the hardware demonstrations For our simulated experiments, we use a desktop with CPU Intel(R) Core(TM) i9-9820X CPU @ 3.30GHz, GPU NVIDIA A6000, and 64GB of RAM. For our hardware demonstrations, we used a desktop with CPU Intel(R) Core(TM) i7-7800X CPU @ 3.50GHz, GPU NVIDIA Titan RTX, and 32GB of RAM.

H Quantitative Metrics

Success Rate A trajectory is considered a success if the rollout position and orientation target errors are below $1\,\text{cm}$ and 15° respectively and there are no physical violations. To avoid erroneously passing a trajectory that ends on the wrong side of a narrow structure, we also ensure that the end effector is within the correct final volume and likewise avoids incorrect volumes. For the cubby and dresser environments, these volumes are individual cubbies or drawers.

Time After setting up each planning problem, we measured the wall time for each *successful* trajectory. We also measure *Cold Start (CS) Time*, the average time to react to a new planning problem. While both expert pipelines have to compute the entire path, the local controllers only need time to compute a single action. We only consider the cold-start time here, but if the new

planning problem is sufficiently similar to a previous one–such as a minor change in the environment or target–a global planning system could employ an optimizer that can replan quickly [38].

Rollout Target Error We calculate both position and orientation errors from the target for the final end effector pose in the trajectory. We measure position error with Euclidean distance and orientation error with the metric described by Wunsch et al. [39].

Collisions A trajectory can have two types of fatal collisions–when the robot collides with itself or when the robot collides with the scene. When checking for collisions, we use an ensemble of collision checkers to ensure fairness. Collision checking varies across algorithmic implementations, *e.g.* our AIT* implementation uses meshes to check scene collisions, while STORM [26] and Geometric Fabrics [25] use a sphere-based approximation of the robot's geometry. A trajectory is only considered to be in collision if the entire ensemble agrees.

Smoothness We use Spectral Arc Length (SPARC) [40] to measure smoothness. Balasubramanian et al. [40] use a SPARC threshold of -1.6 as sufficiently smooth for reaching tasks. This measurement qualitatively describes the behavior of our benchmark algorithms well, so we used the same threshold for sufficiency. We therefore consider a path to be smooth if both its joint-space trajectory and end effector trajectory have SPARC values below -1.6.

I Local Policy Implementations

Both STORM [26] and Geometric Fabrics [25] require expert tuning to achieve compelling performance, and we worked closely with the authors of these papers to tune them as best as possible for our evaluation. We train a single network on all three environment types, so similarly use a single set of tuning parameters for each algorithm over the entire evaluation set.

J MPNets Implementation and Data

In the original paper, Qureshi et al. [41] trained MPNets for execution on the Baxter robot using a dataset of 10 different tabletop environments, each with 900 plans. Then, it was evaluated in the same environments using 100 unseen start and goal configurations in each. In total, their real-robot dataset was 10,000 problems.

To compare fairly to MPNets, we generated an analogous set of 10,000 problems within 10 tabletop environments, which we call the *MPNets-Style* dataset. We re-implemented the MPNets-algorithm based on their open-source implementation at https://github.com/anthonysimeonov/baxter_mpnet_experiments.

After we trained our implementation of their model on the MPNets-Style data, it achieved a similar success rate as the one quoted in their paper for the Baxter experiments (78% vs. 85%). We attribute the performance difference to the increased complexity of our environments, which, unlike the original dataset, have varying table geometry in addition to object placement. In the original paper, they quote planning as taking 1 second on average. Our re-implementation took 2.47 seconds on average with a median of 0.02 seconds. Again, we attribute this difference to the increased complexity, given that the median time is so far below the mean. Just as they do in the open source implementation, we employ hierarchical re-planning, but we do not fall back to a traditional planner. If given access to a collision checker, both MπNets and MPNets can use a similar fallback to re-plan, thus achieving theoretically complete performance.

We used the same training setup described in Appendix F to train MPNets. When trained on the MπNets data set, *i.e.* 3.27M demonstrations from the*Hybrid Planner*, MPNets converged within 15 hours.

K Additional Experiments

Training with Mean Squared Error Loss Increases Collisions When trained with a loss of mean-squared-error in configuration space, MπNets has a similar success rate–94.56% vs. 95.33%–but the scene collision rate is significantly higher at 2.39% vs 0.89%.

Representing the Target in Point Cloud Improves Performance When trained with the target fed explicitly through a separate MLP encoder as a position and quaternion, MπNets succeeds less–88.83% vs. 95.33% when the target is specified within the point cloud. In particular, only 91.61% of trajectories get within 1cm of the target vs. 98.83% with the point cloud-based target.

Training with Collision Loss Improves Collision Rate When trained without the collision loss, MπNets collides more often–2.11% vs 0.89% when trained with the collision loss.

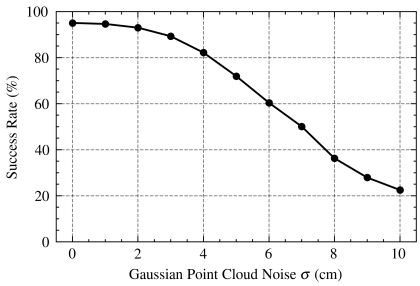

Figure 2: After injecting Gaussian noise into the point clouds, MπNets performance stays fairly constant up until $\sigma = 3\text{cm}$ when success rate is 89.28%.

Training with the Configuration Encoder Improves Success Rate When trained with no robot configuration encoder, *i.e.* with only the point cloud encoder, MπNets has a success rate of 94.17% vs 95.33% when trained with both encoders.

MπNets is Robust to Point Cloud Noise Up to 3.2cm Figure 2 shows MπNets success rate on the set of problems solvable by both planners when random Gaussian noise is added to the point cloud. Model performance stays above 90% until noise reaches 3cm at which point success drops to 89.28%.

MπNets is Robust to Varying Point Cloud Shapes To evaluate performance in out-of-distribution geometries, we replaced all tabletop objects in the test set of problems solvable by the *Hybrid Planner* with randomly meshes from the YCB dataset [42]. For each tabletop primitive, we sampled a mesh from the dataset and transformed it so that the bounding boxes of the primitive and mesh were aligned and of identical size. Note that in these modified scenes, the primitives-based *Hybrid Planner* solution is still valid. MπNets succeeded in 88.33% in this YCB-tabletop test set, whereas with the original primitives, it succeeds in 94.67%. Note that the network was not trained with these geometries—we would expect even higher performance if these meshes were included in the training set.

MπNets is Not Suitable for Unsolvable Problems To evaluate performance on unsolvable problems, we generated a set of 800 planning problems in randomized tabletops where the target is in collision with the table or an object on the table. When used for these problems, MπNets showed a 64.25% collision rate.

MπNets is Not Improved by Combining Experts We trained MπNets-C on a combination of 3.27M demonstrations each from the *Hybrid Planner* and *Global Planner*. Environments may have overlapped in these data sets, but entire problems, *i.e.* environment, start, and goal, did not. In problems solvable by the global planner, MπNets-C—like MπNets-G—outperformed MπNets-H in terms of target convergence (97.17% vs 87.72%). While its collision rate is lower than MπNets-G, (18.56% vs 21.94%) MπNets-C's collision rate is still significantly higher than MπNets-H (11%). The behavior of MπNets-C is essentially an average of MπNets-G and MπNets-H, which we attribute to the lack of easily learnable obstacle avoidance behavior by the *Global Planner*. These demonstrations equate to additional noise in the training data, which creates less successful obstacle avoidance behavior. In future work, we intend to explore how to robustly combine experts for improved performance.

L Dynamic Environments

MπNets is an instantaneous policy that assumes a static world at the time of inference. If the scene changes between inference steps, the policy will react accordingly. If the environment is continually changing–as is often the case in dynamic settings–MπNets implicitly approximates the dynamic movement as a sequence of static motions. When the scene changes are slow, this assumption works well. When the changes are fast, it does not. To demonstrate this, we evaluated MπNets in a static tabletop environment with a single, moving block placed on the table. We generated 1,000 planning problems across the table with the block placed at different locations. We specifically chose problems where MπNets succeeds when the block is stationary. When moving, the block

follows a periodic curve in x and y, but the two curves have indivisible periods, preventing repetitive movement. We then moved the block at three different speeds: slow, medium, and fast and measured the success rate. At these speeds, MπNets succeeds 88.1%, 57.4%, and 28.3% respectively.

M Real-World Experiments

We demonstrated MπNets in a variety of tabletop problems using a Franka Emika Panda 7-DOF manipulator. A calibrated Intel Realsense L515 RGB-D camera is placed in front of the robot's workspace, viewing the table and potential obstacles on top of it. Point cloud measurements are filtered to remove all points belonging to the robot geometry. The remaining cloud is downsampled to 4096 points and treated as the obstacle. The filtering process runs at $9\,\mathrm{Hz}$. We investigated two different control methods:

Open-Loop Motion: Using a fixed, user-defined goal location and the current depth observation, MπNets is rolled out over 80 timesteps or until goal convergence. The resulting path is used to compute a time-parametrized trajectory [43] which is then tracked by a position controller. The videos listed under "Open Loop Demonstrations" at https://mpinets.github.io show a mix of sequential motions toward pre-defined goals. In some of the examples, the objects are static throughout the video and in others, we re-arrange the objects throughout the video. Despite the changing scene, these are still open-loop demos. While the motions adapt to changing obstacles in the scene, the policy only considers scene changes that happen before the execution of a trajectory. This is because the point cloud observations are only updated once the robot reaches its previous target.

Closed-Loop Motion: To account for dynamic obstacles MπNets is rolled out for a single timestep at the same frequency as the point cloud filter operates ($9\,\mathrm{Hz}$). A time-parametrized trajectory is generated by linearly interpolating $\approx 70\%$ of the rolled out path. As in the open-loop case, the resulting trajectory is tracked by a PD controller at $1\,\mathrm{kHz}$. The videos listed under "Closed Loop, Dynamic Scene Demonstrations" at https://mpinets.github.io show examples of boxes thrown into the robot's path while it is moving towards a user-defined target. The evasive maneuver shows MπNets' ability to react to dynamic obstacles.

N Limitations

Training Distribution The limitations of the *Hybrid Planner* translate to limitations in the trained policy network. Certain target poses and starting configurations can create unanticipated behavior. When target poses are narrowly out of distribution, the rollout fails to converge to the target, but as a target poses drifts further from the training distribution, behavior becomes erratic. Likewise, random, initial configurations–such as from rejection-sampling based inverse kinematics–can create unexpected behavior, but we did not observe this in our real robot trials running the policy continuously to a sequence of points. With an improved expert, *e.g.* one with the consistency of our *Hybrid Expert* and guaranteed convergence of the *Global Planner*, we anticipate that the occurrence of failure cases will diminish. We also do not expect the network to generalize to wholly unseen geometries without more training data. But, in future work, we aim to improve the generalization of this method with more data, much in the way that Large Language Models [44] continue to improve generalization, as well as by employing strategies to address covariate shift such as DAgger [21] and domain adaptation.

Real Robot System When used on a real robot, performance will degrade as the robot's physical environment drifts from the training distribution. Likewise, performance will degrade with increasing point cloud noise. In order to ensure safe operation in a real-robot system, MπNets could be combined with a collision checker–either one with ground-truth or a learned, such as Scene Collision Net [45]. The collision checker could be used to a) stop the robot before hitting collisions b) make small perturbations to nudge the policy back into distribution or c) enable a traditional planner to plan to the goal. Alternatively, a safety component could be trained to detect whether the scene is outside of the training distribution to alert the operator that MπNets is ill-equipped to handle a particular scene.

		Success Rate (%)			
	Soln. Time (s)	Global	Hybrid	Both	Smooth (%)
Global Planner [27]	16.56 ± 0.88	100	73.50	100	56.86
Hybrid Planner	6.82 ± 1.50	44.33	100	100	99.22
G. Fabrics [25]	0.11 ± 0.06	37.83	66.67	65.83	88.61
STORM [26]	3.65 ± 1.64	53.50	76.67	77.33	59.72
MPNets [41]					
Hybrid Expert	2.68 ± 17.39	44.67	59.00	66.17	17.26
Random	0.06 ± 0.06	32.17	50.17	53.67	100.00
MπNets (Ours)					
Hybrid Expert	0.33 ± 0.08	67.00	94.33	93.17	93.06
Global Expert	0.34 ± 0.07	74.83	81.50	80.00	93.44

Table 3: Algorithm performance on cubby problems sets solvable by planner types. All prior methods use state-information and a oracle collision checker while MπNets only needs a point cloud

In a physical system, not all problems will have feasible solutions. As discussed in Appendix K, MπNets will often collide in these scenarios, underscoring the need for some additional safety mechanisms to prevent catastrophic behavior. Additionally, MπNets has no concept of history and can collide with the scene if, for example, the robot arm blocks the camera mid-trajectory. To mitigate this, the perception system could employ a historical buffer or filter to maintain some memory of the scene.

Emergent Behavior In some of our test problems, we observed that MπNets produces a rollout where the final gripper orientation is $180°$ off from the target about the gripper's central axis (*i.e.* the central axis parallel to the fingers). In the test set of problems solvable by the *Global Planner*, this occurs in 2.44% of rollouts. We suspect this behavior is due to the near-symmetry in the gripper's mesh about this axis. The minor differences between the two sides of the gripper may not provide enough information for the Pointnet++ encoder to distinguish between these two orientations. While the rollout does not match the requested problem, this behavior can be desirable in some circumstances. For example, because grasps are symmetric with the Franka Panda gripper, a $180°$ rotation is preferable if it reduces the likelihood of a collision. For applications where this behavior is unacceptable, we could replace the target representation in the point cloud with points sampled from a mesh with no symmetry.

O Experimental Results per Environment

In this section, we present the evaluation metrics broken down by environment type. However, we omit Cold Start Time because for global methods, it is the same as the total time and for local methods, the type of environment does not affect startup or reaction time.

The Tabletop environment is the least challenging with the highest success rates for all methods. In general, the dresser environment is the most challenging due to its complex geometry, as evidenced by the high collision rates. When trained with the *Hybrid Expert*, MπNets has the highest rollout target error in the cubby problems solvable by *Global Planner*. Since MπNets trained with the *Global Expert* does not have this issue, we attribute it to a lack of adequate coverage in the training dataset.

References

[1] M. Likhachev, G. J. Gordon, and S. Thrun. Ara*: Anytime a* with provable bounds on sub-optimality. In *NIPS*, 2003.

[2] M. Likhachev, D. Ferguson, G. J. Gordon, A. Stentz, and S. Thrun. Anytime dynamic a*: An anytime, replanning algorithm. In *ICAPS*, 2005.

	% Env. Coll.	% Self Coll.	% Jnt Viol.	% Within 1cm	% Within 5cm	% Within 15°	% Within 30°
G. Fabrics [25]	5.00	0.17	0.67	40.17	57.83	84.67	89.17
STORM [26]	0.50	0.00	0.50	79.33	85.33	69.17	80.33
MπNets (Ours)							
Hybrid Expert	10.67	0.17	0.00	75.83	84.50	75.83	81.67
Global Expert	23.17	0.00	0.00	99.17	100.00	99.33	100.00

Table 4: Failure Modes on cubby problems solvable by the global planner

	% Env. Coll.	% Self Coll.	% Jnt Viol.	% Within 1cm	% Within 5cm	% Within 15°	% Within 30°
G. Fabrics [25]	4.83	0.00	1.00	72.50	83.00	95.83	96.33
STORM [26]	0.17	0.17	0.33	87.33	89.33	89.17	91.67
MπNets (Ours)							
Hybrid Expert	0.50	0.00	0.00	99.83	99.83	100.00	100.00
Global Expert	16.67	0.00	0.00	99.50	100.00	99.83	100.00

Table 5: Failure Modes on cubby problems solvable by the hybrid planner

[3] S. M. LaValle. Rapidly-exploring random trees : a new tool for path planning. *The annual research report*, 1998.

[4] S. Karaman and E. Frazzoli. Sampling-based algorithms for optimal motion planning. *The International Journal of Robotics Research*, 30:846 – 894, 2011.

[5] 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. *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3067–3074, 2015.

[6] M. P. Strub and J. D. Gammell. Advanced bit* (abit*): Sampling-based planning with advanced graph-search techniques. *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 130–136, 2020.

[7] Y. Uno, M. Kawato, and R. Suzuki. Formation and control of optimal trajectory in human multijoint arm movement. *Biological Cybernetics*, 61:89–101, 1989.

[8] A. H. Qureshi, A. Mousavian, C. Paxton, M. C. Yip, and D. Fox. Nerp: Neural rearrangement planning for unknown objects. *ArXiv*, abs/2106.01352, 2021.

[9] M. Sundermeyer, A. Mousavian, R. Triebel, and D. Fox. Contact-graspnet: Efficient 6-dof grasp generation in cluttered scenes. *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 13438–13444, 2021.

[10] N. D. Ratliff, M. Zucker, J. A. Bagnell, and S. S. Srinivasa. Chomp: Gradient optimization techniques for efficient motion planning. *2009 IEEE International Conference on Robotics and Automation*, pages 489–494, 2009.

[11] J. Schulman, Y. Duan, J. Ho, A. X. 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*, 33:1251 – 1270, 2014.

[12] N. D. Ratliff, M. Toussaint, and S. Schaal. Understanding the geometry of workspace obstacles in motion optimization. *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4202–4209, 2015.

[13] S. J. Russell. Learning agents for uncertain environments (extended abstract). In *COLT' 98*, 1998.

	% Env. Coll.	% Self Coll.	% Jnt Viol.	% Within 1cm	% Within 5cm	% Within 15°	% Within 30°
G. Fabrics [25]	5.00	0.00	1.17	72.33	84.33	96.33	97.33
STORM [26]	0.00	0.00	0.00	88.33	89.00	89.33	91.67
MπNets (Ours)							
Hybrid Expert	0.50	0.00	0.00	99.83	100.00	99.83	100.00
Global Expert	18.17	0.00	0.00	99.00	100.00	100.00	100.00

Table 6: Failure Modes on cubby problems solvable by both the global and hybrid planners

		Success Rate (%)			
	Soln. Time (s)	Global	Hybrid	Both	Smooth (%)
Global Planner [27]	16.97 ± 0.81	100	66.83	100	75.63
Hybrid Planner	9.19 ± 2.81	37.33	100	100	99.82
G. Fabrics [25]	0.26 ± 0.12	15.00	25.83	28.50	78.94
STORM [26]	5.54 ± 1.84	24.17	58.50	62.00	83.22
MPNets [41]					
Hybrid Expert	15.55 ± 46.31	12.83	41.83	41.67	26.68
Random	1.61 ± 7.38	8.33	27.50	31.17	100.00
MπNets (Ours)					
Hybrid Expert	0.34 ± 0.06	78.67	97.00	96.33	91.56
Global Expert	0.33 ± 0.05	72.33	77.33	82.17	94.89

Table 7: Algorithm performance on dresser problems sets solvable by planner types. All prior methods use state-information and a oracle collision checker while MπNets only needs a point cloud

[14] A. Y. Ng and S. J. Russell. Algorithms for inverse reinforcement learning. In *Proceedings of the Seventeenth International Conference on Machine Learning*, ICML '00, page 663–670, San Francisco, CA, USA, 2000. Morgan Kaufmann Publishers Inc. ISBN 1558607072.

[15] B. D. Ziebart, A. L. Maas, J. A. Bagnell, and A. K. Dey. Maximum entropy inverse reinforcement learning. In *AAAI*, 2008.

[16] D. A. Pomerleau. Alvinn: An autonomous land vehicle in a neural network. In *NIPS*, 1988.

[17] M. Bain and C. Sammut. A framework for behavioural cloning. In *Machine Intelligence 15*, 1995.

[18] L. Ke, J. Wang, T. Bhattacharjee, B. Boots, and S. S. Srinivasa. Grasping with chopsticks: Combating covariate shift in model-free imitation learning for fine manipulation. *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6185–6191, 2021.

[19] A. Mandlekar, D. Xu, J. Wong, S. Nasiriany, C. Wang, R. Kulkarni, L. Fei-Fei, S. Savarese, Y. Zhu, and R. Martín-Martín. What matters in learning from offline human demonstrations for robot manipulation. In *Conference on Robot Learning (CoRL)*, 2021.

[20] F. Codevilla, E. Santana, A. M. López, and A. Gaidon. Exploring the limitations of behavior cloning for autonomous driving. *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 9328–9337, 2019.

[21] S. Ross, G. J. Gordon, and J. A. Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *AISTATS*, 2011.

[22] S. Ross and A. Bagnell. Efficient reductions for imitation learning. In *International Conference on Artificial Intelligence and Statistics*, page 661–668, 2010.

[23] M. Laskey, J. N. Lee, R. Fox, A. D. Dragan, and K. Goldberg. Dart: Noise injection for robust imitation learning. In *CoRL*, 2017.

	% Env. Coll.	% Self Coll.	% Jnt Viol.	% Within			
				1cm	5cm	15°	30°
G. Fabrics [25]	17.17	0.83	0.17	19.83	26.33	57.83	62.33
STORM [26]	4.83	0.17	0.33	42.67	51.67	45.17	53.83
MπNets (Ours)							
Hybrid Expert	17.00	0.83	0.00	98.00	98.67	93.50	94.33
Global Expert	26.67	0.00	0.00	100.00	100.00	99.00	99.83

Table 8: Failure Modes on dresser problems solvable by the global planner

	% Env. Coll.	% Self Coll.	% Jnt Viol.	% Within			
				1cm	5cm	15°	30°
G. Fabrics [25]	18.33	0.00	0.17	36.00	39.00	61.00	66.00
STORM [26]	0.83	0.00	0.33	65.33	67.67	90.17	91.00
MπNets (Ours)							
Hybrid Expert	1.50	0.00	0.00	99.50	99.50	98.83	99.00
Global Expert	19.67	0.00	0.00	100.00	100.00	97.33	99.50

Table 9: Failure Modes on dresser problems solvable by the hybrid planner

[24] S. Ross, G. Gordon, and A. Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *arXiv preprint arXiv:1011.0686*, 2010.

[25] K. V. Wyk, M. Xie, A. Li, M. A. Rana, B. Babich, B. N. Peele, Q. Wan, I. Akinola, B. Sundaralingam, D. Fox, B. Boots, and N. D. Ratliff. Geometric fabrics: Generalizing classical mechanics to capture the physics of behavior. *IEEE Robotics and Automation Letters*, 7:3202–3209, 2022.

[26] M. Bhardwaj, B. Sundaralingam, A. Mousavian, N. D. Ratliff, D. Fox, F. Ramos, and B. Boots. STORM: An integrated framework for fast joint-space model-predictive control for reactive manipulation. 2021.

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

[28] K. K. Hauser and V. Ng-Thow-Hing. Fast smoothing of manipulator trajectories using optimal bounded-acceleration shortcuts. *2010 IEEE International Conference on Robotics and Automation*, pages 2493–2498, 2010.

[29] R. Diankov. *Automated Construction of Robotic Manipulation Programs*. PhD thesis, Carnegie Mellon University, Robotics Institute, August 2010. URL http://www.programmingvision.com/rosen_diankov_thesis.pdf.

[30] I. A. Şucan, M. Moll, and L. E. Kavraki. The Open Motion Planning Library. *IEEE Robotics & Automation Magazine*, 19(4):72–82, December 2012. doi:10.1109/MRA.2012.2205651. https://ompl.kavrakilab.org.

[31] C. R. Garrett. Pybullet planning. https://pypi.org/project/pybullet-planning/, 2018.

[32] K. K. Hauser. Fast interpolation and time-optimization on implicit contact submanifolds. In *Robotics: Science and Systems*, 2013.

[33] I. Misra, C. L. Zitnick, M. Mitchell, and R. B. Girshick. Seeing through the human reporting bias: Visual classifiers from noisy human-centric labels. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2930–2939, 2016.

[34] A. Joulin, L. van der Maaten, A. Jabri, and N. Vasilache. Learning visual features from large weakly supervised data. In *ECCV*, 2016.

	% Env. Coll.	% Self Coll.	% Jnt Viol.	% Within 1cm	% Within 5cm	% Within 15°	% Within 30°
G. Fabrics [25]	19.50	0.33	0.17	40.00	42.67	64.50	68.17
STORM [26]	1.17	0.17	0.50	69.50	72.83	91.00	92.33
MπNets (Ours)							
Hybrid Expert	1.83	0.00	0.00	99.67	99.67	98.50	98.67
Global Expert	14.50	0.00	0.00	100.00	100.00	96.83	99.17

Table 10: Failure Modes on dresser-problems solvable by both the global and hybrid planners

		Success Rate (%)			
	Soln. Time (s)	Global	Hybrid	Both	Smooth (%)
Global Planner [27]	16.01 ± 0.74	100	95.00	100	28.27
Hybrid Planner	6.43 ± 1.18	69.00	96.33	100	100
G. Fabrics [25]	0.14 ± 0.07	62.50	85.50	85.83	88.61
STORM [26]	3.49 ± 1.65	73.00	88.33	88.67	43.83
MPNets [41]					
Hybrid Expert	1.36 ± 7.98	65.67	94.00	94.50	8.23
Random	0.05 ± 0.05	58.17	85.83	89.67	99.94
MπNets (Ours)					
Hybrid Expert	0.33 ± 0.10	81.67	94.67	95.67	96.83
Global Expert	0.33 ± 0.11	78.00	82.33	86.17	80.67

Table 11: Algorithm performance on tabletop problems sets solvable by planner types. All prior methods use state-information and a oracle collision checker while MπNets only needs a point cloud

[35] C. Qi, H. Su, K. Mo, and L. J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 77–85, 2017.

[36] A. L. Maas, A. Y. Hannun, and A. Y. Ng. Rectifier nonlinearities improve neural network acoustic models. In *in ICML Workshop on Deep Learning for Audio, Speech and Language Processing*, 2013.

[37] M. Laskey, J. Lee, R. Fox, A. Dragan, and K. Goldberg. Dart: Noise injection for robust imitation learning. *Conference on Robot Learning*, pages 143–156, 2017.

[38] M. Mukadam, J. Dong, X. Yan, F. Dellaert, and B. Boots. Continuous-time Gaussian process motion planning via probabilistic inference. volume 37, pages 1319–1340, 2018.

[39] P. Wunsch, S. Winkler, and G. Hirzinger. Real-time pose estimation of 3d objects from camera images using neural networks. *Proceedings of International Conference on Robotics and Automation*, 4:3232–3237 vol.4, 1997.

[40] S. Balasubramanian, A. Melendez-Calderon, A. Roby-Brami, and E. Burdet. On the analysis of movement smoothness. *Journal of NeuroEngineering and Rehabilitation*, 12, 2015.

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

[42] B. Çalli, A. Singh, A. Walsman, S. S. Srinivasa, P. Abbeel, and A. M. Dollar. The ycb object and model set: Towards common benchmarks for manipulation research. *2015 International Conference on Advanced Robotics (ICAR)*, pages 510–517, 2015.

[43] T. Kunz and M. Stilman. Time-optimal trajectory generation for path following with bounded acceleration and velocity. *Robotics: Science and Systems VIII*, pages 1–8, 2012.

	% Env. Coll.	% Self Coll.	% Jnt Viol.	% Within 1cm	% Within 5cm	% Within 15°	% Within 30°
G. Fabrics [25]	1.33	0.50	0.17	77.00	87.83	80.67	83.17
STORM [26]	0.50	0.17	0.00	92.00	97.67	79.00	83.83
MπNets (Ours)							
Hybrid Expert	5.33	1.33	0.00	89.33	96.33	84.33	89.67
Global Expert	16.00	0.00	0.00	94.67	98.50	91.33	97.83

Table 12: Failure Modes on tabletop problems solvable by the global planner

	% Env. Coll.	% Self Coll.	% Jnt Viol.	% Within 1cm	% Within 5cm	% Within 15°	% Within 30°
G. Fabrics [25]	1.33	0.00	0.00	97.17	98.00	89.33	89.67
STORM [26]	0.17	0.17	0.17	96.67	99.00	90.67	92.17
MπNets (Ours)							
Hybrid Expert	0.67	0.00	0.00	97.17	99.50	96.17	98.83
Global Expert	11.50	0.00	0.00	97.50	99.50	94.00	98.33

Table 13: Failure Modes on tabletop problems solvable by the hybrid planner

[44] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, G. Krueger, and I. Sutskever. Learning transferable visual models from natural language supervision. In *arXiv:2103.00020*, 2021.

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

	% Env. Coll.	% Self Coll.	% Jnt Viol.	% Within			
				1cm	5cm	15°	30°
G. Fabrics [25]	1.33	0.00	0.00	97.33	98.50	89.50	89.83
STORM [26]	0.17	0.17	0.17	97.17	99.33	90.50	91.83
MπNets (Ours)							
Hybrid Expert	0.50	0.00	0.00	97.33	99.50	96.33	98.33
Global Expert	8.67	0.17	0.00	97.00	99.67	95.83	98.17

Table 14: Failure Modes on tabletop problems solvable by both the global and hybrid planners