EDMP: Ensemble-of-costs-guided Diffusion for Motion Planning

Jayaram Reddy*[1], Kallol Saha*[1], Vishal Mandadi*[1], Ajit Srikanth[1],
Aditya Agarwal[2], Bipasha Sen[2], Arun Singh[3], and Madhava Krishna[1]
[1]Robotic Research Center, IIIT Hyderabad, [2]Massachusetts Institute of Technology, [3]University of Tartu

***Abstract*— Classical motion planning for robotic manipulation includes a set of general algorithms that aim to minimize a *scene-specific cost* of executing a given plan to generate a potentially *valid collision-free* trajectory. This approach offers remarkable adaptability, as they can be directly used for any new scene without needing specific training datasets. However, without a prior understanding of what diverse valid trajectories are and without specially designed cost functions for a given scene, the overall solutions tend to have low success rates. While deep-learning-based algorithms tremendously improve success rates, they are much harder to adopt without specialized training datasets. We propose EDMP, an Ensemble-of-costs-guided Diffusion for Motion Planning that aims to combine the strengths of classical and deep-learning-based motion planning. Our diffusion-based network is trained on a set of diverse general *valid* trajectories enabling the model to implicitly learn the properties of a valid trajectory. Like classical planning, for any new scene at the time of inference, we compute collision costs and incorporate this cost at each timestep of the diffusion network to generate *valid collision-free* trajectories. Instead of a single collision cost that may be insufficient in capturing diverse cues across scenes, we use an ensemble of collision costs to guide the diffusion process, significantly improving the success rate compared to classical planners. As we show in our experiments, EDMP outperforms SOTA deep-learning-based methods in most cases while retaining the generalization capabilities primarily associated with classical planners.**

I. INTRODUCTION

Planning a trajectory from a start to goal position while avoiding collisions (self-collisions and collisions with the objects around the robot) is a fundamental challenge in robotic manipulation. Over the years, many approaches have been introduced to tackle this challenge, including classical planning algorithms like CHOMP [1], TrajOpt [2], and more recent deep-learning-based algorithms like MPNets [3] and πnets [4]. While the latter approaches tremendously improved the success rate in the manipulation tasks (determined by the number of times the manipulator reaches the goal while avoiding collisions), classical planners greatly reduce the dependency on specialized training datasets, generalizing to any random scene and therefore are still the go-to off-the-shelf choice for motion planning.

In a classical planning approach, a motion plan for a new scene is generated on the go by minimizing a *cost function* computed on the given scene. For example, minimizing a cost function that includes the collision cost, path length, and reaching a goal. This fundamental concept enables such planning algorithms for any general scene. However, designing this cost function with multiple constraints requires many hit-and-trials, and careful fine-tuning, and still may not capture the diversity across scene structures.

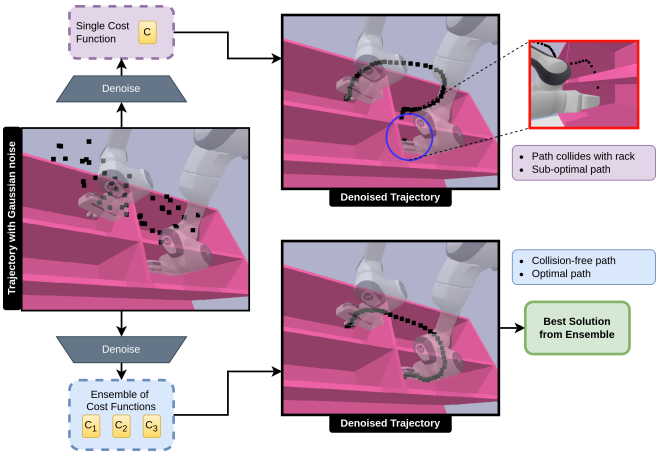


Fig. 1: **Importance of Ensemble**. Using a single cost function alongside a diffusion model may risk collision or generate sub-optimal trajectories. EDMP utilizes an ensemble of cost functions with varying hyperparameters along with the diffusion model. This ensures that an optimal solution can be found even in complex scenes.

Deep learning-based methods [5] improve upon this by adopting data-driven techniques that learn a mapping from a given context (or the scene) to a solution using a neural network. This allows the network to gain an overall understanding of different scene structures and map the structures to a viable solution. Such methods are significantly faster and more accurate. Despite the gains, they falter in terms of generalization – performance on out-of-distribution scenes is significantly impacted. In this work, we aim to bridge the gap between the two approaches by first learning a prior (as in deep-learning-based methods) of "valid" trajectories that meet generic constraints of what a trajectory should look like for *any* scene and then incorporating scene-specific cost, such as collision cost, directly at the time of inference (as in classical planners). As we show in the experimental section, this builds a powerful motion planner that generalizes to many diverse scenes. For example, the manipulator holding any arbitrary objects (as shown in Figure 1).

Recently, diffusion models [6], [7] have been widely adopted to learn classifier-guided policies [8]. Unlike generative models, such as GANs [9], and VAEs [10], diffusion models generate datapoints in multiple timesteps. In the

*Denotes equal contribution; authors list follows alphabetical order
Project website

backward pass (generation step), a diffusion policy generates the output by slowly removing "noise" in h steps to eventually denoise a given noisy input. This allows for a unique opportunity to "guide" the generative network to generate a plan based on "scene-specific cues," (such as a collision cost) as in the classical planners, at each of the intermediate steps. In such a case, a collision cost can be computed using a differentiable algorithm to incorporate collision gradients that can guide the planning. We develop on top of this approach to fuse the qualities of "classical" and "deep learning" planners, in which, we train general planning prior unaware of any scene-specific context, such as the "collisions", at the train time and guide the prior to generate a collision-free path directly at inference by incorporating a collision cost like GJK [11].

Our general prior informs the network of what a *valid* trajectory would look like. For instance, going from point A to B in the joint space without any drastic motion that could harm the manipulator, feasible trajectory, and avoiding self-collision. At the time of inference, a random noisy trajectory is sampled, which is guided through "collision cost" to generate a collision-free valid trajectory. However, we observe that the chances of a successful path are highly dependent on the kind of cost. For example, as shown in Fig 1, a single cost function C fails to retract the manipulator as it pushes the waypoints on either side of the rack, To tackle this, we propose, **EDMP**, an "**E**nsemble-of-costs-guided **D**iffusion for **M**otion **P**lanning, in which, we rely on N different classifiers to guide the diffusion process, thereby generating a diverse set of trajectories with a very high success rate. Interestingly, based on the concepts of diffusion, our approach generates multi-modal trajectories containing many valid collision-free trajectories that could be handpicked on different parameters such as path length, smoothness, etc., as we show later in the experimental section.

Overall, our contributions are as follows,

1) We propose **EDMP**, an **E**nsemble-of-costs-guided **D**iffusion for **M**otion **P**lanning which combines the strength of classical planning with deep learning to generate versatile plans. Unlike previous methods that rely on a single classifier guide for specific tasks like obstacle avoidance, EDMP leverages multiple guides to shape the trajectory's posterior distribution.
2) As a result of incorporating an ensemble of guides to shape the trajectory posterior, our framework exhibits the advantageous capability of generalization to novel scenes and objects without the necessity for additional training at the same time outperforming the classical planners and performing comparably with the SOTA deep-learning-based planners.
3) The ensemble of guides enhances the diversity of the solution set that can be adequately exploited by donwstream planners by chosing a trajectory that evaluates to an optimal objective function from the solution set.

II. RELATED WORK

Apart from the works cited in Introduction, [12]–[14] have used diffusion models for motion planning to plan a smooth trajectory from given start and goal states. [15] improves behavior cloning methods in generating different robot behaviors using class conditional diffusion models and classifier-free guidance. Unlike them, we work explicitly on motion planning for collision avoidance with classifier-based guided diffusion and portray substantial performance gain over prior works. [16] shares certain similarities with ours. It introduces classifier-guided diffusion for manipulator motion planning using Signed Distance Function of environment as a classifier. However, it only shows generalization capabilities across different start-goal pairs and new obstacles within the same scene. In contrast, our approach showcases generalization over a variety of scenes not encountered during the training of the denoiser exploiting the advantages provided by the ensemble of guides. We also conduct various ablation studies, including sensitivity to the dataset and multimodality guided by contrastive loss, while benchmarking against state-of-the-art planners.

III. BACKGROUND

Diffusion models are a category of generative models where a datapoint is converted into an isotropic Gaussian noise by iteratively adding Gaussian noise through a fixed forward diffusion process $q(x_t|x_{t-1}) = \mathcal{N}(x_t; \sqrt{1-\beta_t}x_{t-1}, \beta_t\mathbf{I})$, where β_t is the variance schedule and t is the diffusion timestep. A forward diffusion process of T timesteps can be reversed by sampling $x_T \sim \mathcal{N}(0, \mathrm{I})$ from a standard normal distribution and iteratively removing gaussian noise using the trainable reverse process $p_\theta(x_{t-1}/x_t) = \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \Sigma_t)$ parametrized by θ.

IV. ENSEMBLE-OF-COSTS GUIDED DIFFUSION

Given a robotic manipulator made of m joints corresponding to a given joint state $s_i \in \mathbb{R}^m$ for the i^{th} time-step, and a scene, E, our goal is to predict a *valid* collision-free trajectory, τ, in the manipulator's joint space between a start and a goal configuration, s_0 and s_{h-1}, respectively, as a sequence of joint states over a horizon h. This is given as:

$$\tau = [s_0, s_1, \dots\ s_{h-1}]^\top \in \mathbb{R}^{m \times h} \tag{1}$$

A *valid* collision-free trajectory must comply with certain constraints: (1) The trajectory must be kinematically smooth and avoid abrupt motions, (2) The trajectory should avoid self-collisions, (3) The set of joint poses that make up the trajectory must be feasible for the manipulator to reach and lead to the goal configuration, and (4) The trajectory must minimize a collision cost, given as $J(\tau, E)$. We call a trajectory as *valid* when it complies with the first three constraints. A *valid* collision-free trajectory complies with all four constraints.

As shown in Figure 2, we first learn the concept of *valid* trajectories by training a prior. At the time of inference, we incorporate an ensemble of "collision cost" to guide the prior

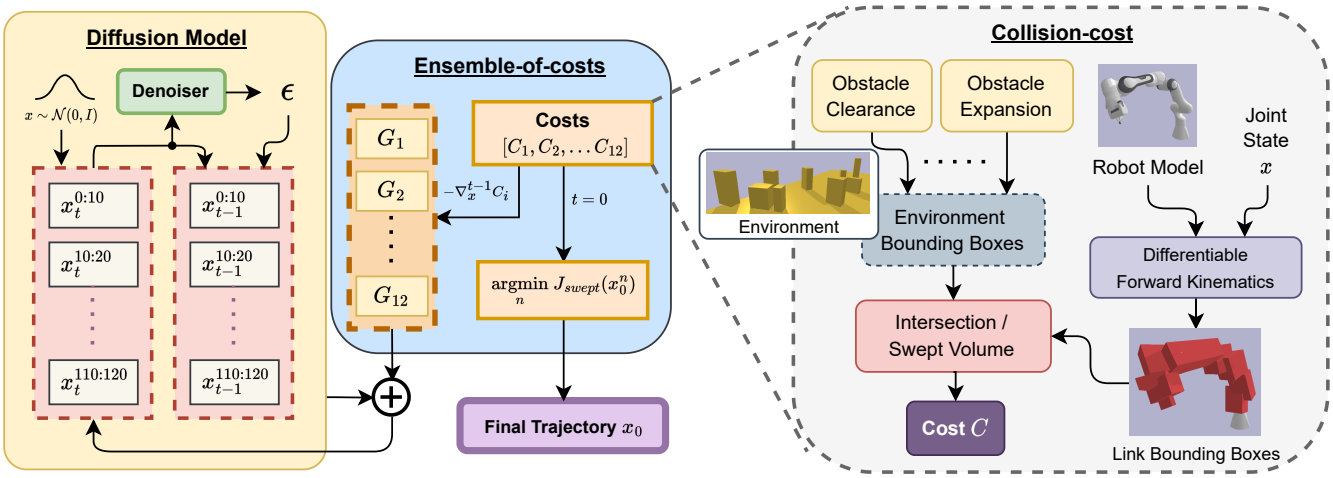


Fig. 2: **Architecture**. EDMP leverages a diffusion model alongside a 12-member cost ensemble. The diffusion model denoises a batch of trajectories while each cost in the ensemble guides a specific slice of the batch. The cost function calculates the intersection or swept volume from robot and environment bounding boxes. The trajectory with minimum swept volume is chosen as the final solution

into generating a valid *collision-free* trajectory. We elaborate on the two concepts in the following sub-sections. Which 2?

A. Learning a trajectory prior using Diffusion models

We want to build a generative model that implicitly understands the concept of *valid* trajectories that can be later used to generate valid *collision-free* trajectories. To achieve this, we train a diffusion model to learn a prior p_θ, over a set of valid trajectories collected from a large-scale dataset [4]. For each of the trajectories, we train a forward diffusion process by repeatedly adding Gaussian noise to the trajectory for h timesteps. Assuming the diffusion process as a Markov chain where a data sample x_{t-1} solely depends on x_t, we get the joint probability $p_\theta(x_{1:T}) = p(x_T) \prod_{t=2}^{T} p_\theta(x_{t-1}|x_t)$. This allows us to model the distribution:

$$x_0 \sim p_\theta(x_0) = \int p(x_T) \prod_{t=1}^{T} p_\theta(x_{t-1}|x_t)\, dx_{1:T} \qquad (2)$$

Note that in this stage of offline training, there is no notion of "collisions," and rather, we select trajectories that smoothly translate from a start to a goal configuration. That is, the trajectory avoids any abrupt motions (between any two sets of joint configurations) and avoids self-collisions. In the reverse diffusion process, an initial noisy trajectory sampled from isotropic gaussian noise (equation add eqn) is iteratively denoised to get a valid trajectory in h timesteps.

B. Collision Cost Guidance

TODO: Talk about how diffusion is enabling us to do this sort of cost-guidance.

Once an offline prior has been trained, we want to use this prior to find a valid *collision-free* trajectory for the specific scene, E, by introducing the notion of collision cost, $J(\tau, E)$. At the time of inference for a given scene E, we randomly sample a noisy trajectory from p_θ and denoise it iteratively. In each step of the denoising process, we modify the intermediate trajectory τ_t predicted by the denoiser at

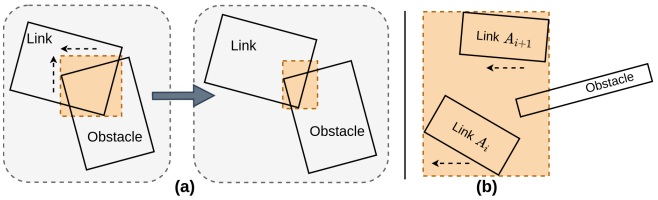


Fig. 3: (a) Gradient of link-obstacle intersection volume moves link away from obstacle. (b) Gradient of swept volume between consecutive link poses prevents collision between trajectory waypoints

the t^{th} timestep by adding gradients from the scene-specific collision cost function, before passing to the next step. This is given by,

$$\tau_t^* = \tau_t - \alpha \frac{\partial J(\tau_t, E)}{\partial \tau_t} \qquad (3)$$

where α is a hyperparameter. This form of conditioning guides the trajectory to collision-free regions. This is analogous to classifier-based guidance in [8], in which a trained classifier guides the diffusion towards a goal. Although, adding cost-conditioning directly at the time of inference results in a cost-conditioned posterior, denoising from the cost-guided posterior is equivalent to denoising from a Gaussian distribution, as shown in [16].

Computing the collision cost: The collision cost is a differentiable function that indicates the distance of the manipulator from the objects in the given environment. If the manipulator's state is colliding with any of the object's on the table, the collision cost should indicate the "penetration" (negative distance) that must be negated to exit the collision state.

In Fig. 4, add (a) and (b) instead of left and right

To detect and compute the collision cost between the manipulator and the scene, we take inspiration from GJK [11] and the Expanding Polytope Algorithm (EPA) [17]. We compute the collision cost between two 3D bodies as a

function of overlap between them along the three axes. We refer to this as the penetration depth. To achieve this, we first approximate each object on the table as a cuboid corresponding to the smallest 3D bounding box that encloses the full object. Similarly, to compute the 3D bounding box for the manipulator, we use the robot's link (a function of its joint configuration defined by the differentiable forward kinematics (FKs)) given in its URDF. Given l links, we construct a sequence of l 3D bounding boxes across the planning horizon h defined by $FK(\tau) \in \mathbb{R}^{l \times h \times 4 \times 4}$. If there are n obstacles in the scene, we can approximate the scene as a list of n bounding boxes $\in \mathbb{R}^{n \times 4 \times 4}$. Each bounding box is represented as a transformation matrix of size 4×4 in the SE(3) Lie Group. We then compute the collision cost as the intersection volume, V, of each link state in τ with respect to each obstacle in E given as:

$$\mathrm{V}(\mathrm{FK}(\tau), E) = \sum_{i=0}^{h-1} \mathrm{V}(\mathrm{FK}(s_i), E) \tag{4}$$

Here, V is the intersection volume computed between the two cuboids through a simple min-max operation given as:

$$\mathrm{V}(A, B) = \mathrm{prod}(|\max(\min(A),\ \min(B)) - \min(\max(A),\ \max(B))|)$$

For a batch, b, of trajectories τ_t at a denoising step t, we can define the collision cost as a summation of the intersection volumes over the trajectories in the batch as:

$$J_{inter}(\tau_t, E) = \sum_{k=0}^{b-1} \mathrm{V}(\mathrm{FK}(\tau_t^k), E) \tag{5}$$

While it is possible to compute a more accurate collision cost through EPA, they entail intricate and lengthy computational graphs. Although, in classical optimization methods, such a precise collision cost is of great important, in our case - since the diffusion already understands the concept of *valid* trajectories, the collision-cost guidance is needed to solve a comparatively simpler problem of approximating the movement direction for a link to exit a collision state based on the penetration depth.

Our method of collision-cost-based guidance introduces a slight shift in the trajectory distribution at every denoising step. This shift progressively increases the likelihood of the final trajectory τ_0 being collision-free. As we show in the experiments, this simple collision cost enables collision avoidance even in complex scenes. Note that, in the denoising process, trajectory samples may include joint configurations that fall beyond the joint limits. To address this issue, we clip the joint values within the joint limits before applying guidance.

$$J_{swept}(x_t, E) = \sum_{k=0}^{b-2} \mathrm{V}(\mathrm{SV}(x_t^k, x_t^{k+1}), E) \tag{6}$$

C. Ensemble of Collision Costs

Even though collision cost-guided diffusion enables generalization to any arbitrary scene, often, a single cost function is not enough to model the variations in structures across diverse scenes, as shown in Figure 1. To solve this problem, deep learning based on previous methods tries to learn a context map between a scene context and a solution by training over large variations of scene structures. However, incorporating such diversities in cost-function is non-trivial for the classical planners. To tackle this, we propose "ensemble-of-cost" that accounts for many different cost functions to obtain the most best *valid* and *collision-free* trajectories for the given scene context.

Problem in Section – introducing a particular cost function – underrepresented for different kinds of scenes – as different scenes have varying structure – deep learning methods see varying configuration and then map it into the guidance

– not trivial to apply them to the classical planning algorithms – why???

– why an ensemble of classifier (cost function) to get a set of solutions – and then choose the solution that is the best among the solutions – **bring out the intermediate aspect – reduce the sampling time**

– our novelty is not in the classifiers themselves, but rather in the mechanism of combining different classifiers to achieve optimal trajectories. The empirical results are shown in Table X.

A mechanism to combine cost functions to improve the overall guidance.

Explanation of why an ensemble of classifiers works for diffusion models

Classical planners – doesn't work well with multiple guidance

Scenes with varying structure calls for a different kind of guidance

Incorporate multiple cost functions to improve the overall guidance

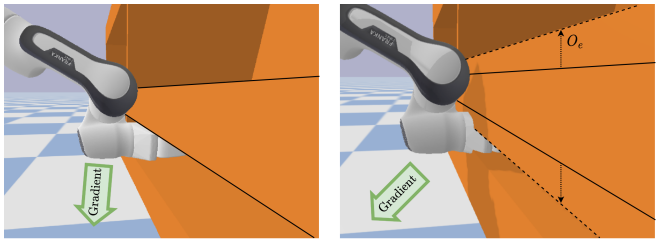
A single guide with a fixed hyperparameter set is underconstrained for generating trajectories for a diverse set of scenes. Notably, trajectory guidance through a single cost-function presents several challenges. For instance, an inherent challenge is that gradients tend to diminish as the link's penetration depth into an obstacle approache zero. Similarly, for obstacles with high aspect ratio, like a narrow horizontal shelf, gradient-driven movement will tend to move the manipulator upwards or downwards to evade collision, thus failing to reach a configuration that retracts the manipulator around the edge of the shelf. To tackle these challenges, we introduce a set of hyperparameters that improve the trajectory guidance process. Our collision classifier is defined using these hyperparameters as $C(J(x_t, E), O_c^t, O_e^t, \eta_t, 1_{norm})$, where J is the guidance method (J_{inter} intersection volume cost or J_{swept} swept volume cost), α is the learning rate, 1_{norm} is a boolean variable indicating whether to perform gradient normalization, and O_c^t and O_e^t are obstacle clearance and obstacle expansion hyperparamters for each step of the denoising

process. ~~O_c^t is used for expanding the dimensions of the obstacle by a marginal value and O_e^t limits the aspect ratio by expanding the narrow dimension~~. The gradient is thus defined as, $G = -\nabla_x^{t-1} C = \frac{\partial J(x_t, E)}{\partial x_t}$.

We make the case that for a particular scene, there are several trajectories that are collision-free. We empirically demonstrate that using a single guide with a specified hyperparameter setting may not identify these collision-free regions, thus resulting in low success rates add ref table **??**. This is primarily because different hyperparameter settings have varying rates of success across scenes. For instance, scenes with confined spaces such as drawers require lower clearance in comparison to tabletop scenes with larger obstacles. To deal with these scene-specific variations and ensure high success rate, we propose a mechanism to combine the guidance from multiple of cost-functions called "mixture of experts". more about this!. As shown in **??**, incorporating multiple cost functions improve the overall guidance significantly. However, incorporating such guidance in classical planners is non-trivial (why??. Deep-learning methods are adapt at incorporating variations in configurations, and mapping it into the guidance. Consequently, our method of a diffusion prior guided by cost-functions naturally enables such an extension. The ensemble of classifiers can be applied in parallel to a batch of trajectory distributions. Each classifier in the ensemble operates on a specific slice of the batch. For a batch size of b, we apply classifier C_i to the slice of trajectories $x_t^{(i-1)\frac{b}{n_c} \,:\, (i)\frac{b}{n_c}}$ After the denoising process is complete, we select the trajectory with the minimum swept volume $J_{swept}(x_0, E)$.

V. EXPERIMENTS

The goal of our experiments is to evaluate the capabilities of our planner in generating trajectories that avoid collisions to reach a desired target configuration from an initial configuration in the simulator for a wide variety of scenes. In particular, we are interested in evaluating, (1) the ability of the planner in generalizing to unknown scenes, and (2) the ability to work directly on out-of-distribution and object-in-hand scenarios without any retraining.



(a) (b)

Fig. 4: Obstacle Expansion widens the thinner dimension, thus altering the gradient direction. This helps the manipulator find a configuration for retracting around the shelf.

A. Experimental Setup

Our training consists of learning a diffusion prior over valid trajectories following the framework defined in (sec IV-A). During training, we condition the diffusion model on the initial and final configuration/position of the original trajectory. We see that without this explicit conditioning, the diffusion model aligns the start and the end point of the trajectory close to the immediately neighboring points, causing jerky movement close to the start and end point of the trajectory. We explain more about this in (sec VI-A). Our denoiser is modeled as a standard temporal UNet similar to [12], and is trained for $20k$ epochs on a single Nvidia GeForce 2080 Ti GPU taking approximately 9 hours. We test our model in the PyBullet simulation environment on the Franka Emika Panda 7-DOF manipulator, following a setup similar to MπNets.

Dataset We evaluate our approach against the baselines on a wide variety of dataset of varying difficulty. We evaluate against the datasets provided in MπNets [4]. MπNets provides a large dataset consisting of 6.54 million trajectories, of which 3.27 million trajectories are generated by a *Global Planner*, a typical state-of-the-art configuration space pipeline. Another 3.27 million trajectories are generated by the *Hybrid Planner*, which uses AIT* [18] for planning in the end-effector space and Geometric Fabrics [19] for producing a geometrically consistent motion conditioned on the generated end-effector waypoints. Each trajectory contains 50 waypoints, including the initial and final joint configurations. Similar to MπNets, we train three different planners; one on *global* data called EDMP-G, another on *hybrid* data called EDMP-H, and the last one on the combined dataset of *hybrid* and *global*, called *both* which we refer as EDMP-C.

Mpinets don't release the full global data, The metrics are taken from the paper directly.

More information regarding data collection can be found in the MπNets paper.

Baselines To demonstrate the strength of our approach, we compare our framework against different types of state-of-the-art planners. We compare against two behavior cloning-based planners (MπNets [4], MPNets [3]), two stochastic optimization-based planners (G. Fabrics [19], STORM [20]), one optimization-based planner with quintic-spline initialization (CHOMP [1]), and a set of sampling-based planners through OMPL [21] on *global*, *hybrid*, and *both* test sets.

Metrics We define success rate (SR $\%$) as the percentage of problems that are successfully solved by the given planner. We say that a problem is successfully solved if the planner can generate a trajectory that avoids all collisions in order to reach the goal in the simulation. We use this metric to compare our framework's performance against other baselines on the aforementioned datasets.

B. Results on benchmarking datasets/Planning on unknown scenes

Table I presents the comparative performance of EDMP against several motion planning approaches. Table I shows that EDMP outperforms CHOMP, OMPL, G. Fabrics,

STORM. Table II shows that EDMP outperforms MPNets, and MπNets global expert, but falls slightly short behind the MπNets hybrid expert on both *hybrid* and *both* datasets on the success rate. CHOMP (with quintic b-spline initialization), as expected, fails to solve a huge number of problems due to its tendency to get stuck in local minima during the optimization process pref a citation. OMPL, while probabilistically complete, struggles to find valid solutions within the specified 20-second time limit, especially in complex collision-free configurations like cubby, merged cubby, and dresser scenes. STORM and G. Fabrics also exhibit low success rates across all datasets primarily due to their local planning nature and susceptibility to local minima. To show distinction, its good to write as, however; unlike us, they fail at ...

MPNets, which integrates classical planning with deep learning, offers improved planning performance by leveraging a learned neural sampler, via behavior cloning, to generate waypoints. This sampler acts as an informative scene prior, enhancing planning success compared to purely classical planners like OMPL. However, MPNets still lags behind MπNets and our framework in efficiently utilizing priors. write strongly here why MPNets fail even though they do similar stuff as us – intuitive reasoning – poor generalization capability, why? – purely a data-driven approach, empirical results - this is also evident in Table II, in which MPNets perform poorly on the global set when trained on the hybrid planner

Mention that we picked up the scores directly from MπNets

We outperform the MπNets global expert by a substantial margin on *hybrid* and *both* sets and approach the performance level on the *global* set. MπNets is a state-of-the-art motion planner known for its effectiveness when abundant expert demonstrations are available. Although, mpinets better performance on *global* could be attributed to its overfitting due to behavior cloning from expert demonstrations, our performance exceeds on both *hybrid* and *both* due to overall better generalizability. what are we beating mpinets again on? overfitted because of behavior cloning – bring out the central theme – we are more generalizable

These results demonstrate that integrating classical stochastic optimization methods with trajectory priors that capture common scene patterns significantly enhances performance, surpassing both traditional planners and data-driven deep learning approaches. This enhancement is achieved through an ensemble of cost guided diffusion model, where the guidance through cost functions emulates stochastic optimization across priors encoded by the diffusion model.

C. Out-of-distribution and Object-in-Hand

To showcase the generalizability of our approach to out-of-distribution scenarios, we generate scenes with randomly generated spherical objects at diverse spatial positions. Even on this challenging dataset, our model performs reasonably well on 8 out of 10 evaluated scenes as depicted in Fig. 7,

Test	CHOMP	OMPL	G. Fabrics	STORM	EDMP-C
Global	26.67	37.27	38.44	50.22	**72.28**
Hybrid	31.61	40.37	59.33	74.5	**85.941**
Both	32.2	42.6	60.06	76	**86.304**

TABLE I: Comparison of EDMP-C against classical motion planners in terms of success rate (%, ↑).

	Test		
Train/Expert	Global	Hybrid	Combined
MπNets-G	**75.06**	80.39	82.78
EDMP-G	71.667	**82.833**	82.xx
MPNets Hybrid Expert	41.33	65.28	67.67
MπNets-H	75.78	**95.33**	**95.06**
EDMP-H	74.33	86.133	85.0628
EDMP-C	72.28	85.941	86.304

TABLE II: Comparison of EDMP against behavior cloning-based motion planners in terms of success rate (%, ↑).

suggesting a fairly general capability in handling out-of-distribution scenarios. write more strongly, refer to figures and tables

No experiments to backup this claim: In cases where the manipulator has objects gripped between its fingers, conventional behavior cloning methods like MπNets require retraining to plan for collision avoidance between the held object and the environment. However, our framework easily adapts to this scenario by incorporating the object as an additional link with fixed joints and introducing an object-environment cost to the overall cost function. This adaptation allows us to guide the diffusion model to avoid both link-environment and object-environment collisions, making our method compatible with objects in hand, as illustrated in bottom row of Fig. 7.

D. Qualitative Results

VI. ABLATION STUDIES

A. Training with and without conditioning

Table III add table number illustrates that training with conditioning, where the initial and the final configurations of the trajectory are aligned with those of the ground truth trajectories generates smoother trajectories than training without conditioning. Notably, the smoothness between the initial two waypoints and the final two waypoints is substantially higher when the denoiser is trained with conditioning.

This discrepancy in smoothness can be attributed to the role of conditioning. When trained with conditioning, the denoiser's primary objective is to acquire the ability to generate a smooth trajectory between the specified start and goal points. Consequently, during inference, with fixed start and goal points, the denoiser endeavors to establish a smooth path, assuming the constancy of these reference points.

In contrast, in the absence of conditioning during denoiser training, the model focuses on seamlessly connecting all data points to construct a coherent trajectory, without assuming fixed start and goal points. Consequently, during inference,

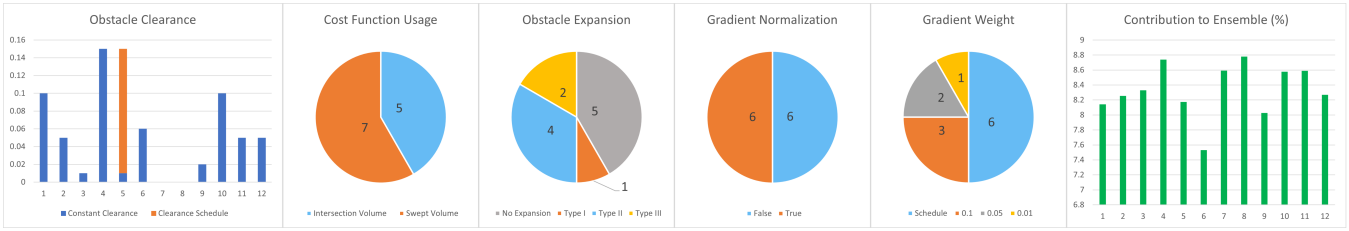


Fig. 5: Left-to-right: Obstacle clearance per guide, guide count per cost function type, guide count per obstacle expansion type, guide count normalizing gradients, guide count per gradient weight, and individual guide contributions to the ensemble.

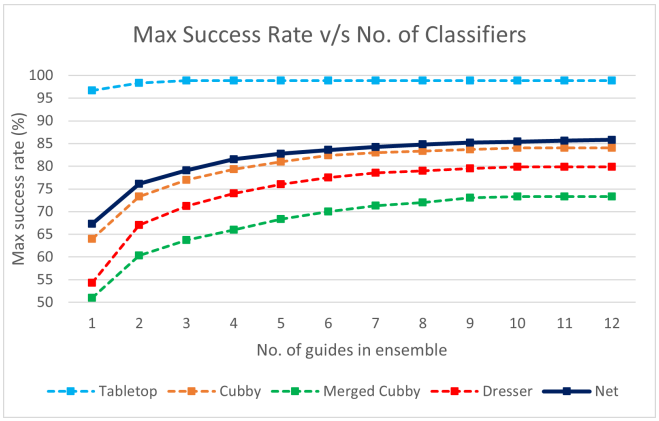


Fig. 6: Graph showing the asymptotic rise in maximum success rate with increasing number of guides in the ensemble, demonstrated across tabletop, cubby, merged cubby, and dresser scenes.

Training type	Avg. Roughness (↓)	Max Roughness excluding start and goal (↓)	Roughness between first two waypoints (↓)	Roughness between last two waypoints (↓)
with Conditioning	**0.0589**	**0.0821**	**0.0266**	**0.0266**
without Conditioning	0.1121	0.3171	0.6459	0.4960

TABLE III: Various roughness metrics to show that training with conditioning naturally provides more smoother trajectories than training without conditioning

the denoiser attempts to adjust not only the intermediate points but also the start and goal points, resulting in increased roughness between the initial two and final two waypoints when these points are reverted to their original values. We show this quantitatively in Table III. As can be seen, conditioning results in smooth trajectories between the start and end configuration, whereas, w/o conditioning, the denoiser fails to produce a smooth trajectory between the start and end configuration.

B. Multimodality Gradient

I would this section with the aim, what are the benefits of multimodality, how diffusion models seamlessly integrate/enable this (or merge with how our overall method incorporates this), and what's our approach - how do we incorporate multimodality gradients?, good to mention why its non-trivial for classical planners Inspired by the CLIP based guidance [22], [23] in text to image generation, we apply additional gradients which we refer to as multimodality gradients guided by the contrastive loss (eq 7) during reverse

Method	Success Rate (%) (↑)	ACSM (↓)	ACSR (↓)	ACSV (↑)
w/o MM	79.25	0.981	0.124	0.001
w/ MM (0.5)	79.17	0.969	0.212	0.002
w/ MM (0.75)	78.83	0.946	0.465	0.0088
w/ MM (1.0)	75.92	**0.865**	0.782	0.049
EDMP	Kallol, fill this	0.892	**0.899**	**0.067**

TABLE IV: **Comparision of EDMP against the average of classifiers with varying weights of multimodality gradients. Best results are bold.**

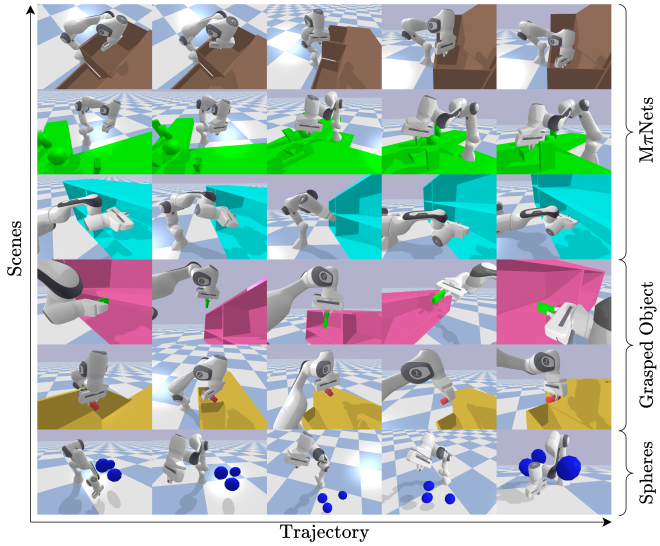


Fig. 7: Our experiments include MπNets dataset scenes, trajectory generation with a grasped object, and out-of-distribution scenes like collision spheres. Displayed from top-to-bottom: dresser, tabletop, merged cubby, merged cubby with handheld cuboid, dresser with handheld cylinder, and a collision spheres scene

diffusion process apart from the classifier based gradients which is primarily used for avoiding collisions, we aim to make trajectories more diverse by applying these multimodality gradients. To make this possible, we compute cosine similarity matrix C on the batch of 120 trajectories and we apply contrastive loss (eq 7) which is cross entropy between similarity matrix(C) and Identity matrix(I) in addition to the swept volume cost (eq 6) and use the gradients of this combined cost function (eq 8) to make our classifier guided diffusion model more multimodal in nature.

$$J_{similarity}(C, I) = -\sum_{i,j} I_{ij} \log(C_{ij}) \tag{7}$$

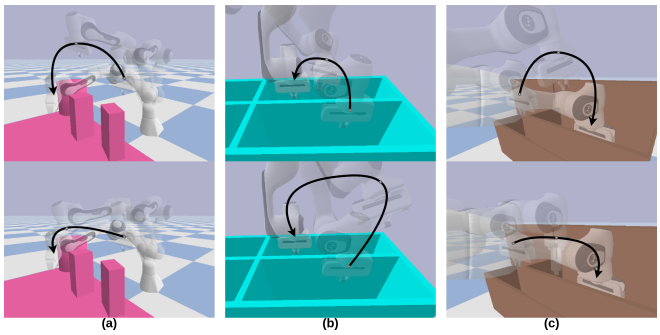


Fig. 8: Left column: Multimodal trajectories in a tabletop scene. Middle column: Multimodal trajectories in a merged cubby scene. Multimodal trajectories in a dresser scene.

$$J_{combined} = J_{swept} + J_{similarity} \tag{8}$$

Metrics: We employ three key metrics based on Cosine Similarity to assess trajectory diversity for 100 random scenes from the MπNets validation dataset. Lower ACSM (Average of Cosine Similarity Mean), higher ACSR (Average Cosine Similarity Range) and , higher ACSV (Average Cosine Similarity Variance) values all suggest greater diversity among trajectories. All these metrics are averaged across 12 classifiers in our study for robust assessment.

We find that ensemble of classifiers is better in making trajectories more diverse which indicates that our model EDMP implicitly brings more multimodality compared to any of the single classifier-guided diffusion models additionally guided by the contrastive Loss.

VII. CONCLUSION

generalizable and easily adaptable without any further training Our method showcases the complementary benefits of classical planners and data-driven deep-learning approaches learning a prior using diffusion models allows our model to seamlessly generalize to unknown scenes with . Our model makes it trivial to modify the cost function . We additionally showcase that by far our model is the most general on scenes with varying complexity, and how it generalizes to object-in-hand scenes (completely different from the scenes in training dataset) without any retraining. Interestingly, we see multimodality as a byproduct of our model that allows xyz. What are the disadvantages of our model?

While the diffusion model harnesses the benefits of sampling-based planning methods, our gradient-based guidance harnesses the advantages of optimization-based planners. When combined, we get a potent generative model which can produce collision-free trajectories for a manipulator in an environment.

ACKNOWLEDGMENT

The Authors would like to thank Adam Fishman for assisting with the MPINET baseline and providing valuable insights into collision checking during experimentation and benchmarking.

REFERENCES

[1] 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*, 2009, pp. 489–494. DOI: 10.1109/ROBOT.2009.5152817.

[2] J. Schulman, J. Ho, A. Lee, I. Awwal, H. Bradlow, and P. Abbeel, "Finding locally optimal, collision-free trajectories with sequential convex optimization," Jun. 2013. DOI: 10.15607/RSS.2013.IX.031.

[3] A. H. Qureshi, A. Simeonov, M. J. Bency, and M. C. Yip, "Motion planning networks," in *2019 International Conference on Robotics and Automation (ICRA)*, 2019, pp. 2118–2124. DOI: 10.1109/ICRA.2019.8793889.

[4] 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.

[5] 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)*, IEEE, 2021, pp. 6010–6017.

[6] J. Sohl-Dickstein, E. Weiss, N. Maheswaranathan, and S. Ganguli, "Deep unsupervised learning using nonequilibrium thermodynamics," in *International conference on machine learning*, PMLR, 2015, pp. 2256–2265.

[7] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," *Advances in neural information processing systems*, vol. 33, pp. 6840–6851, 2020.

[8] P. Dhariwal and A. Nichol, *Diffusion models beat gans on image synthesis*, 2021. arXiv: 2105.05233 [cs.LG].

[9] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, *et al.*, *Generative adversarial networks*, 2014. arXiv: 1406.2661 [stat.ML].

[10] D. P. Kingma and M. Welling, *Auto-encoding variational bayes*, 2022. arXiv: 1312.6114 [stat.ML].

[11] E. Gilbert, D. Johnson, and S. Keerthi, "A fast procedure for computing the distance between complex objects in three-dimensional space," *IEEE Journal on Robotics and Automation*, vol. 4, no. 2, pp. 193–203, 1988. DOI: 10.1109/56.2083.

[12] M. Janner, Y. Du, J. B. Tenenbaum, and S. Levine, "Planning with diffusion for flexible behavior synthesis," *arXiv preprint arXiv:2205.09991*, 2022.

[13] A. Ajay, Y. Du, A. Gupta, J. Tenenbaum, T. Jaakkola, and P. Agrawal, "Is conditional generative modeling all you need for decision-making?" *arXiv preprint arXiv:2211.15657*, 2022.

[14] Z. Liang, Y. Mu, M. Ding, F. Ni, M. Tomizuka, and P. Luo, *Adaptdiffuser: Diffusion models as adaptive self-evolving planners*, 2023. arXiv: 2302.01877 [cs.LG].

[15] C. Chi, S. Feng, Y. Du, *et al.*, "Diffusion policy: Visuomotor policy learning via action diffusion," *arXiv preprint arXiv:2303.04137*, 2023.

[16] J. Carvalho, A. T. Le, M. Baierl, D. Koert, and J. Peters, "Motion planning diffusion: Learning and planning of robot motions with diffusion models," *arXiv preprint arXiv:2308.01557*, 2023.

[17] G. Bergen, "Proximity queries and penetration depth computation on 3d game objects," Jan. 2001.

[18] 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.

[19] M. Xie, K. V. Wyk, A. Li, *et al.*, *Geometric fabrics for the acceleration-based design of robotic motion*, 2021. arXiv: 2010.14750 [cs.RO].

[20] M. Bhardwaj, B. Sundaralingam, A. Mousavian, *et al.*, "Storm: An integrated framework for fast joint-space model-predictive control for reactive manipulation," in *Conference on Robot Learning*, PMLR, 2022, pp. 750–759.

[21] I. A. Sucan, M. Moll, and L. E. Kavraki, "The open motion planning library," *IEEE Robotics Automation Magazine*, vol. 19, no. 4, pp. 72–82, 2012. DOI: 10.1109/MRA.2012.2205651.

[22] C. Saharia, W. Chan, S. Saxena, *et al.*, "Photorealistic text-to-image diffusion models with deep language understanding," *Advances in Neural Information Processing Systems*, vol. 35, pp. 36 479–36 494, 2022.

[23] A. Ramesh, P. Dhariwal, A. Nichol, C. Chu, and M. Chen, *Hierarchical text-conditional image generation with clip latents*, 2022. arXiv: 2204.06125 [cs.CV].