2024 IEEE International Conference on Robotics and Automation (ICRA 2024)
May 13-17, 2024. Yokohama, Japan

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

Kallol Saha*[1], Vishal Mandadi*[1], Jayaram Reddy*[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

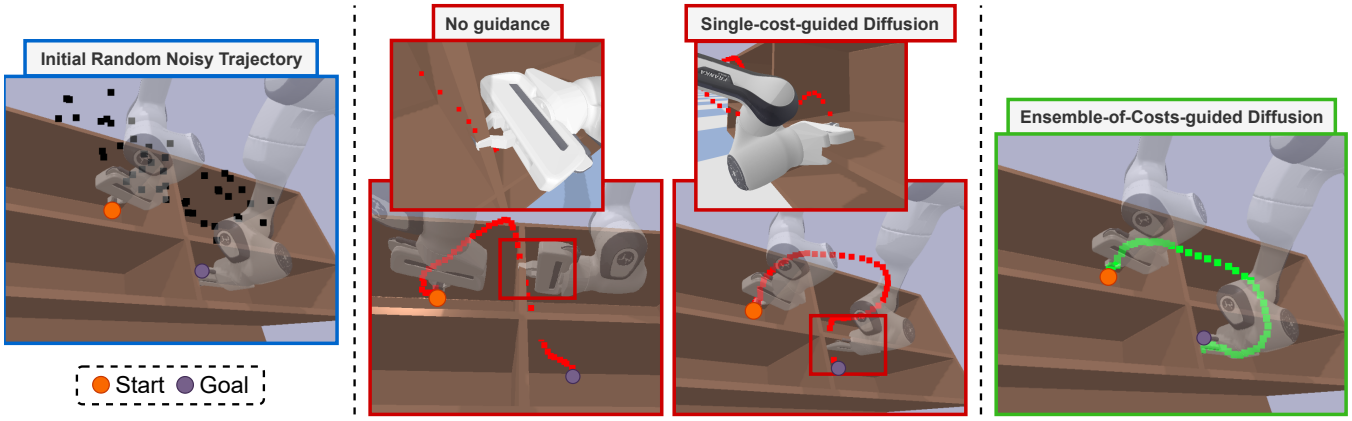

Fig. 1: **E**nsemble-of-costs-guided **D**iffusion for **M**otion **P**lanning combines the strength of classical planning and deep-learning. We leverage a diffusion policy to learn a prior over kinematically *valid* trajectories and guide it directly at the time of inference using scene-specific costs such as "collision-cost" to also satisfy the scene-specific constraints. Instead of using a single-cost, we propose the use of multiple cost functions (ensemble-of-cost-guidance) to capture variations across scenes, enabling us to generalize to diverse scenes.

***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. This approach offers remarkable adaptability, as they can be directly used off-the-shelf 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 within a certain time limit. 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 kinematically valid trajectories. Like classical planning, for any new scene at the time of inference, we compute scene-specific costs such as "collision cost" and guide the diffusion to generate valid trajectories that satisfy the scene-specific constraints. Further, instead of a single cost function that may be insufficient in capturing diversity across scenes, we use an *ensemble* of costs to guide the diffusion process, significantly improving the success rate compared to classical planners. EDMP performs comparably with SOTA deep-learning-based methods 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 [1]–[3]. Over the years, many approaches have been introduced to tackle this challenge, including classical planning algorithms like [4]–[16], and more recent deep-learning-based algorithms like MPNets [17], and MπNets [18]. 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 novel scenes and therefore are still widely 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 collision cost, path length, and the distance to goal. 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.

Deep learning-based methods [19] 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 via behavior cloning [20]. 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 generalization – performance on out-of-distribution scenes is significantly impacted. They also require a large number of high-quality demonstrations and are shown to be inefficient when fed with multimodal datasets [21], for example, data collected using teleoperation [22].

In this work, we aim to bridge the gap between the two approaches by first learning a prior (as in deep-learning-based methods) over kinematically *valid* trajectory for *any*

*Denotes equal contribution
Project website & codebase: https://ensemble-of-costs-diffusion.github.io/

scene and then incorporating *scene-specific cost*, such as collision cost, directly at the time of inference (as in classical planners). As shown in the experimental section, this builds a powerful motion planner that generalizes to diverse scenes.

Recently, diffusion models [23], [24] have gained tremendous popularity for generating diverse sets of modalities such as vision [25]–[27], language [28], and more recently in motion planning [29]–[32]. Unlike generative models like GANs [33] and VAEs [34], diffusion models generate datapoints in multiple timesteps. In the backward pass (generation step), a diffusion process generates the output by iteratively removing "noise" in h steps to eventually denoise a given noisy input. For example, generating a kinematically valid trajectory from a noisy trajectory. As shown in [29], [35], this allows for a unique opportunity to "guide" the generation process by incorporating "scene-specific cues," (such as a collision cost) as in the classical planners, at each of the intermediate steps, thereby producing a kinematically valid trajectory that also satisfies the scene-specific constraints. Taking inspiration from this, we propose, **EDMP**, an **E**nsemble-of-costs-guided **D**iffusion for **M**otion **P**lanning, that first learns a prior of kinematically valid trajectories and then is guided at the time of inference to satisfy scene-specific constraints while retaining the kinematic validity. Further, instead of relying on a single cost function, we incorporate N different cost functions (ensemble-of-costs) to guide the diffusion process, thereby generating trajectories that can solve diverse challenging scenes (as shown in Fig 1).

Interestingly, based on the concepts of diffusion, EDMP can generate multimodal trajectories (multiple ways of getting from point A to point B). The ensemble-of-costs further improves the diversity in multimodality that could be handpicked on different parameters such as path length or smoothness (see Figure 9). Our key contributions are:

1) We propose **EDMP**, an **E**nsemble-of-costs-guided **D**iffusion for **M**otion **P**lanning which combines the strengths of classical and deep learning-based motion planning by first learning a prior over kinematically valid trajectories and then using multiple cost functions to capture diverse scene-specific cost guidance directly at the time of inference.
2) EDMP generalizes to diverse novel and even out-of-distribution scenes (such as planning for a manipulator holding an arbitrary object) while performing comparably with SOTA deep-learning-based methods on specific datasets.
3) Based on diffusion, EDMP generates multimodal trajectories. Further, the ensemble-of-costs improves the diversity of multimodality. This can be exploited by downstream planners to choose a trajectory from the set of solutions that evaluates to an optimal objective.

II. Problem Formulation

Consider a robotic manipulator with m joints corresponding to a given joint state $\boldsymbol{s}_i \in \mathbb{R}^m$ for the i^{th} trajectory-time-step where a trajectory, $\boldsymbol{\tau}$, is a sequence of joint states over a horizon h between a start and goal configuration, $\boldsymbol{s}_0$ and $\boldsymbol{s}_{h-1}$, respectively, given as, $\boldsymbol{\tau} = [s_0, s_1, ... \ \boldsymbol{s}_{h-1}]^\top \in \mathbb{R}^{m \times h}$. We are interested in solving the following optimization problem for obtaining a trajectory for a given task,

$$\min_{\boldsymbol{\tau}} J(\boldsymbol{\tau}; \mathbf{o}, \boldsymbol{E}) \tag{1}$$

The vector $\mathbf{o}$ represents the hyperparameters of the cost function J. The variable $\boldsymbol{E}$ represents the scene. The cost function has two distinct parts. The first part models aspects that depend purely on the manipulator's kinematics, such as smoothness and feasibility (e.g. self-collision) of the joint trajectory. The second part represents the scene-specific cost that couples the manipulator motion with the scene.

A. Diffusion Based Optimal Solution

Diffusion models [23], [24] 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 $q(\boldsymbol{\tau}_t|\boldsymbol{\tau}_{t-1}) = \mathcal{N}(\boldsymbol{\tau}_t; \sqrt{1-\boldsymbol{\beta}_t}\boldsymbol{\tau}_{t-1}, \boldsymbol{\beta}_t \mathrm{I})$, where $\boldsymbol{\beta}_t$ is the variance schedule and t is the diffusion timestep. A forward diffusion process of T timesteps can be reversed by sampling $\boldsymbol{\tau}_T \sim \mathcal{N}(0, \mathrm{I})$ from a standard normal distribution and iteratively removing gaussian noise using the trainable reverse process $p_\theta(\boldsymbol{\tau}_{t-1}/\boldsymbol{\tau}_t) = \mathcal{N}(\boldsymbol{\tau}_{t-1}; \mu_\theta(\boldsymbol{\tau}_t, t), \boldsymbol{\Sigma}_t)$ parametrized by θ.

We train a diffusion model to learn a prior p_θ, over a set of smooth kinematically valid trajectories collected from a large-scale dataset [18]. For a given trajectory in the dataset, forward diffusion first corrupts it by gradually adding noise, and then the reverse diffusion process aims to reconstruct back the original trajectory from the noise. This is shown in Figure 3-(Diffusion Model).

Conditioning on start and goal: As we train the diffusion prior on the set of trajectories from the dataset, we repeatedly fix $\boldsymbol{s}_0$ and $\boldsymbol{s}_{h-1}$ at each diffusion-time-step to correspond to the original $\boldsymbol{s}_0$ and $\boldsymbol{s}_{h-1}$ at the first diffusion-time-step for each trajectory. As shown in Section V, this creates a conditioning effect on the start-goal pair, encouraging the network to generate a smooth trajectory between the two.

B. Guidance Process

For a specific scene, we also want to incorporate a *scene-specific cost* function that can guide the diffusion process toward a trajectory that satisfies the scene-specific constraints, such as collision cost, denoted by $J(\boldsymbol{\tau}; \mathbf{o}, \boldsymbol{E})$. Specifically, at each step of the denoising-time-step, we modify the intermediate trajectory $\boldsymbol{\tau}_t$ predicted by the denoiser at the t^{th} diffusion-time-step by adding gradients from the scene-specific collision cost function, before passing to the next step as shown in Figure 3. This is given by,

$$\boldsymbol{\tau}_t^* = \boldsymbol{\tau}_t - \alpha \nabla J(\boldsymbol{\tau}; \mathbf{o}, \boldsymbol{E}) \tag{2}$$

where α is a hyperparameter. This form of conditioning is analogous to classifier-based guidance in [36], where a trained classifier guides the diffusion towards a goal. In our case, it guides the trajectory to collision-free regions. Although adding cost-conditioning directly at the time of inference results in a cost-guided posterior, denoising from

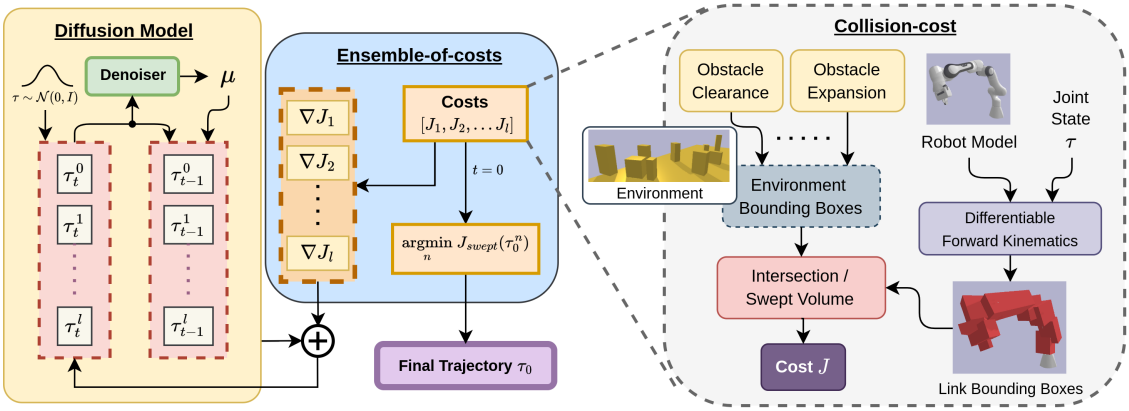

Fig. 3: **Architecture**. EDMP leverages a diffusion model alongside an ensemble of l cost functions. The diffusion model denoises a batch of trajectories $\boldsymbol{\tau}$ from $t = T$ to 0, while each cost in the ensemble guides a specific sub-batch. We calculate the gradient ∇J of each collision cost (intersection or swept volume) in the ensemble from robot and environment bounding boxes, using differentiable forward kinematics. After denoising is over, the trajectory with minimum swept volume is chosen as the solution.

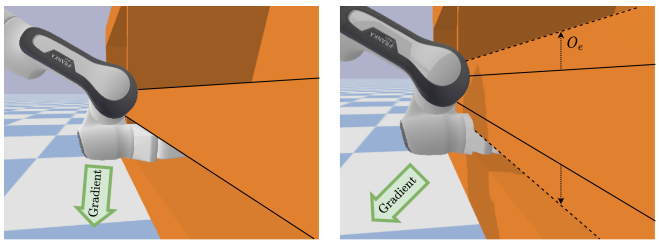

Fig. 4: **Effect of Obstacle Expansion.** (a) scene without obstacle expansion applied. (b) shows the change in the scene after applying obstacle expansion. Obstacle Expansion widens the thinner dimension by an amount O_e, thus altering the gradient direction which enables the manipulator to retract around the shelf.

the cost-guided posterior is equivalent to denoising from a Gaussian distribution, as shown in [35].

III. Ensemble-of-Costs Guided Diffusion

Equation 2 uses gradients from a scene-specific cost function to modify the trajectory obtained from the diffusion prior. Thus, the nature of the gradient dictates the efficacy of the process, which in turn depends on many factors, some of which we summarize below:

- **Algebraic form of the cost model:** Collision cost is highly scene-specific. For example, the signed-distance-field-based collision cost [4] is known to struggle in scenes with narrow racks between the start and the goal positions. Similarly, performance of collision costs modeled around convex collision checking [6] depends on whether we penalize each independent waypoint's intersection volume with the environment (intersection volume) or the swept volume between each of the adjacent waypoints (swept volume) as shown in Figure 5.
- **Gradient and Cost Hyperparameters:** Each collision cost model has several hyperparameters that also affect the gradient descent step in Equation 2. For example, gradient weight schedule, whether adaptive or independent, is an important parameter in gradient-based approaches. Similarly, we show that in some cases, choosing to expand the object over the initial period of optimization greatly helps in bringing out the retraction behavior, which otherwise seemed less likely as shown in Figure 4 (retraction prevents the manipulator from colliding with the shelf).

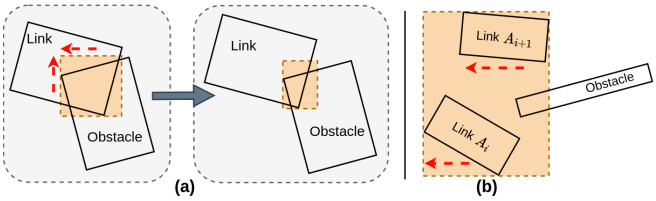

Fig. 5: **Intersection v/s Swept Volume**. (a) Gradient (red arrows) of link-obstacle intersection volume moves link away from obstacle. (b) Gradient (red arrows) of swept volume between consecutive link poses prevents collision between trajectory waypoints

The above discussion makes the case for guided diffusion with an adaptive learning rate and collision cost whose algebraic form and hyper-parameters are adaptive to the environment in which the manipulator is operating. However, to the best of our knowledge, there exists no general framework towards this end. We propose an ensemble-of-collision-costs where we run a batch of guided diffusion processes in parallel, as shown in Figure 3 where each sub-batch uses a different cost function. Moreover, we embed the hyperparameter tuning of the cost functions as a part of the diffusion process. Algorithm1 summarizes our proposed improvements.

We assume that we have l different choices of the cost function stemming from different collision models with each having their own hyperparameters. We also consider r different schedules to adapt the hyperparameter of each of the l costs. Thus, in Algorithm 1, we have a triple loop that runs $l \times r$ parallel diffusion processes. In practice, however, the outer two loops are decoupled from each other and run in batch fashion over GPUs.

A. Collision Cost Guidance

To detect and compute the collision cost between any two rigid bodies, we take inspiration from Gilbert-Johnson-Keerthi (GJK) [37] algorithm and the Expanding Polytope

Algorithm 1: Reverse Diffusion Guided with a Ensemble of Costs

Input: Learned reverse diffusion $\boldsymbol{\mu}_\theta(\boldsymbol{\tau}_t, t)$, Covariance schedule $\boldsymbol{\Sigma}_t$, learning rate schedule α_t, hyperparameter schedule $\mathbf{o}_t$

Initialization: $\boldsymbol{\tau}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, Start joint state $\boldsymbol{s}_0$, final joint state $\boldsymbol{s}_{h-1}$

for $j = 1, \ldots, \ldots l$ **do**
  for $i = 1, \ldots, \ldots r$ **do**
    for $t = T, \ldots, \ldots 1$ **do**
      ${}^{j,i}\boldsymbol{\tau}_{t-1} \sim \mathcal{N}({\boldsymbol{\mu}_\theta}({}^{j,i}\boldsymbol{\tau}_t, t) + {}^{j,i}\alpha_t \nabla_{\boldsymbol{\tau}}{}^{j,i} J(\mathbf{o}_t), \boldsymbol{\Sigma}_t)$

Return ${}^{j}\boldsymbol{\tau}_0$ corresponding to minimum collision cost.

Algorithm (EPA) [38]. We compute the collision cost between two bodies, A and B as a function of overlap between them along the three axes. We refer to this as the penetration depth. To achieve this, we enclose each object in a bounding box cuboid and compute the intersection volume V as:

$$\begin{aligned} \mathrm{V}(A, B) \quad &= \mathrm{prod}(|\max(\min(A),\ \min(B)) \\ &\quad - \min(\max(A),\ \max(B))|) \end{aligned} \tag{3}$$

To leverage this in manipulator collision avoidance, we first approximate each object on the table as a cuboid corresponding to a 3D bounding box that encloses the full object. Similarly, we represent each manipulator link as a 3D cuboid, which acts as a bounding box enclosing that link. We compute the poses of each of these bounding boxes for a given configuration s_k using differentiable forward kinematics $\mathrm{FK}(\boldsymbol{s}_k[i])$, where $\boldsymbol{s}_k[i]$ denotes the configuration of i^{th} link. If there are n obstacles in the scene, we can approximate the scene as a list of n bounding boxes $\boldsymbol{E} \in \mathbb{R}^{n \times 4 \times 4}$. 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 from the intersection volume, which we call "intersection volume cost", J_{inter}, of each link state in $\boldsymbol{\tau}$ with respect to each obstacle in E given as:

$$J_{inter}(\boldsymbol{\tau}, \boldsymbol{E}) = \sum_{k=0}^{h-1} \sum_{i=0}^{m-1} \mathrm{V}(\mathrm{FK}(\boldsymbol{s}_k[i]), \boldsymbol{E}) \tag{4}$$

The gradient of intersection volume cost ∇J_{inter} gives the penetration depth along each dimension. Averaging these values produces an average movement direction for a link along the penetration depth, to displace it out of collision, as shown in Figure 5. We show empirically that this simple collision cost enables collision avoidance even in complex scenes.

We define a swept volume cost function from swept volume SV, inspired from [37], as:

$$\begin{aligned} &SV(\boldsymbol{s}_k[i], \boldsymbol{s}_k[i+1]) = \\ &\mathrm{prod}(|\max(\mathrm{FK}(\boldsymbol{s}_k[i]), \mathrm{FK}(\boldsymbol{s}_k[i+1])) \\ &- \min(\mathrm{FK}(\boldsymbol{s}_k[i]), \mathrm{FK}(\boldsymbol{s}_k[i+1]))|) \end{aligned} \tag{5}$$

The swept volume approximates the volume swept out by a link between two adjacent waypoints. Such a cost function helps account for collisions that may happen between consecutive joint states in a trajectory, as in Figure 5.

For a given trajectory $\boldsymbol{\tau}$, and environment $\boldsymbol{E}$, we define swept-volume cost as the sum of volumes swept out by the links between each pair of adjacent waypoints. As the swept volume is also modelled as a cuboid, we can plug it into Equation 4 to get the swept volume cost:

$$J_{swept}(\boldsymbol{\tau}, \boldsymbol{E}) = \sum_{i=0}^{h-2} \sum_{i=0}^{m-1} \mathrm{V}(\mathrm{SV}(\boldsymbol{s}_k[i], \boldsymbol{s}_k[i+1]), \boldsymbol{E}) \tag{6}$$

Our architecture is also outlined in Figure 3.

IV. EXPERIMENTS

All our experiments are conducted using the Pybullet simulator [39], employing the Franka Panda robot.

Dataset: We benchmark on the MπNets [18] dataset.

The **training** dataset consists of 6.54 million collision-free trajectories. These 6.54 million trajectories are generated on various scenes (such as tabletop, dresser, and cubby) using two different classical planning pipelines: **Global** Planner (3.7 million trajectories) based on AIT* [40] and **Hybrid** Planner (3.7 million trajectories) that combined AIT* [40] for planning in the end-effector space and Geometric Fabrics [41] for producing a geometrically consistent motion conditioned on the generated end-effector waypoints.

The **test** dataset consists of three datasets: (1) Global Solvable Problem (***global***): consists of 1800 scenes that could be solved by only the Global Planner, that is, Global Planner could generate a valid collision-free trajectory for these scenes. (2) Hybrid Solvable Problems (***hybrid***): consists of 1800 scenes solvable by only the Hybrid Planner. (3) Both Solvable Problem (***both***): a set of 1800 scenes that could be solved by both, Global and Hybrid Planners. More information regarding data collection can be found in [18].

Training and architectural details: Similar to MπNets, we train two different planners; one on *global* and another on *hybrid* data. Each denoiser is modeled as a temporal UNet similar to [29], and is trained for 20k steps on a 2080 GeForce GTX GPU for approximately 9 hours. Each **scene** consists of obstacle configurations, a start position (in joint configuration), a goal position (in joint configuration during training and in end-effector space during test). We use inverse kinematics to compute the joint configuration from the end-effector position. Each **trajectory** consists of 50 waypoints, including the initial and final joint configurations. For our experiments, we use an ensemble of 12 cost functions, 5 with intersection and 7 with swept volume cost. However, one can define their own number and type of cost functions.

Baselines: We compare our framework against different types of SOTA planners: two behavior cloning-based planners (MπNets [18], MPNets [17]), two stochastic optimization-based planners (G. Fabrics [41], STORM [5]), one optimization-based planner with quintic-spline initialization (CHOMP [4]), and a set of sampling-based planners

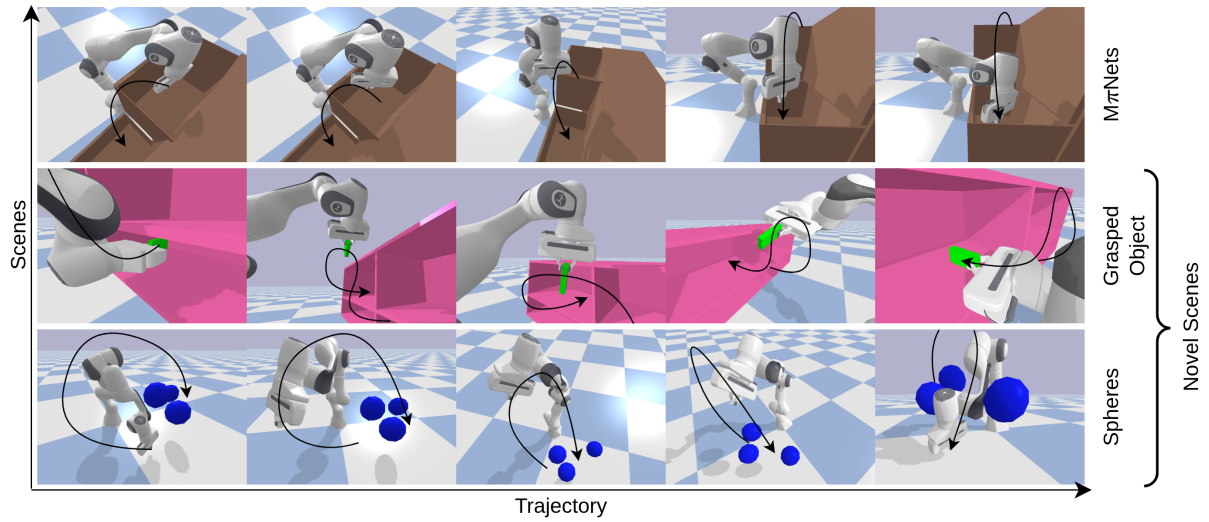

Fig. 6: **Qualitative Results**: From top-to-bottom: MπNets scene (dresser) and out-of-distribution scenes: merged cubby with handheld cuboid and a collision spheres scene.

Test	CHOMP	OMPL	G. Fabrics	STORM	EDMP
Global	26.67	37.27	38.44	50.22	**75.93**
Hybrid	31.61	40.37	59.33	74.5	**86.13**
Both	32.2	42.6	60.06	76	**85.06**

TABLE I: Comparison of EDMP (trained on Hybrid [18] dataset) against classical motion planners in terms of success rate (%, ↑).

	Global-trained Planners		Hybrid-trained Planners		
Test	MπNets	EDMP	MPNets	MπNets	EDMP
Global	**75.06**	71.67	41.33	75.78	**75.93**
Hybrid	80.39	**82.84**	65.28	**95.33**	86.13
Both	82.78	**82.79**	67.67	**95.06**	85.06

TABLE II: Comparison against deep-learning based methods in terms of success rate (%, ↑). EDMP is only marginally affected by the training data since its major improvement is a result of the ensemble-of-costs-guidance common across all settings.

through OMPL [42] on *global*, *hybrid*, and *both* test sets. For fair comparison, we set a planning time limit equal to EDMP's planning time in corresponding scenes.

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. MπNets doesn't release the full global data, therefore the metrics are taken from the paper directly.

A. Performance on MπNets Dataset

Table I and II presents the success rates of EDMP against the baselines. As shown in Table I, EDMP outperforms all of the classical planners significantly and MPNets (Table II). Although MπNets is better than EDMP in many of the settings, it is important to note that our method is agnostic of the variations in the dataset, whereas the performance of methods like MπNets is dependent on the dataset it is behaviorally cloned on. This is because our main performance improvement comes from the ensemble of cost functions that is applied directly at the time of inference and is common across all of the settings (*global*, *hybrid*, and *both*). Moreover, the prior encodes kinematic constraints, which can be learned even from much simpler training datasets like *global* instead of *hybrid*. Moreover, unlike MπNets, EDMP generalizes to out-of-distribution scenes (Section IV-B) and generates multimodal trajectories (Section IV-C).

B. Generalization to Out-of-distribution Scenes

We test EDMP's performance on scenes outside of the training distribution. First, we make the manipulator hold arbitrary objects and see if EDMP can find a *valid collision-free* trajectory for the manipulator along with the object. As can be seen in Figure 6, EDMP works out of the box for such scenes. Deep-learning-based approaches like MπNets and MPNets require retraining or specialized training datasets to take objects into account. In our case, we treat the object as just another link (as explained in Section III-A) with fixed joints and adding an object-environment collision-cost to the overall cost function (Equation 5).

We also test EDMP by generating a scene with collision spheres (Figure 6). Such a scene is very different from the types of scenes the training trajectories were generated on. Even in this highly challenging setup, EDMP succeeds in 8 out of 10 evaluated scenes with spheres generated at random spatial positions, indicating that EDMP has learned a generic prior that is highly adaptable to diverse scenes.

C. Multimodality

Multimodal trajectories can enable one to achieve the same goal in multiple different ways. This is much closer to how humans plan – out of many different ways to execute a task; we often pick the one that is most optimal for the *current* situation. Based on the concept of diffusion, we inherently support such multimodal trajectories. Moreover, our ensemble-of-costs further improves the diversity in the generated trajectories.

We assess trajectory diversity for 100 random scenes from the MπNets validation dataset. We use Average Cosine

Method	SR(↑)	ACSM(↓)
Avg. of single guides	79.25	0.981
EDMP	**89.30**	**0.892**

TABLE III: Comparison of EDMP against the individual guides on 100 MπNet validation scenes.

Condition	AR(↓)	MRESG(↓)	RF2W(↓)	RL2W(↓)
w/cond	**0.0589**	**0.0821**	**0.0266**	**0.0266**
wo/cond	0.1121	0.3171	0.6459	0.4960

TABLE IV: **Ablation on start-goal conditioned trajectory**: As we generate the trajectory through the denoising process conditioned on the start and goal, the trajectories are smoother. The metrics are defined in Section V

.

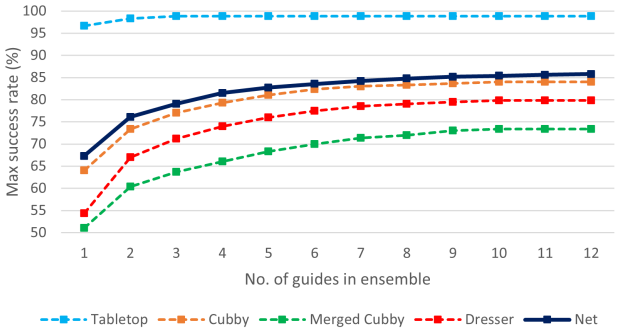

Fig. 7: **Effect of adding guides.** There is an asymptotic rise in the overall Success Rates with the number of guides demonstrated across diverse scenes - tabletop, cubby, merged cubby, and dresser.

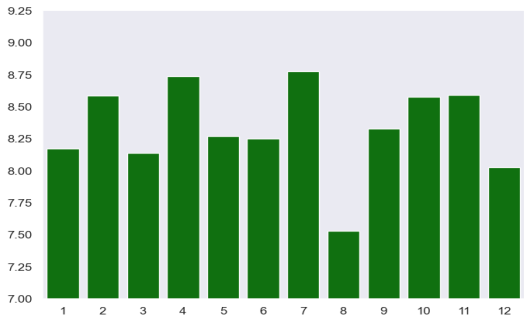

Fig. 8: **Contribution of each guide.** x-axis and y-axis denote the guide index and contribution (%) of each guide to overall Success Rate. The contribution of a guide is proportional to the likelihood that a successful path was generated by it.

Similarity Mean (ACSM) metric that calculates the mean of cosine similarities among all pairs of trajectories within a batch to measure the diversity in trajectories. Lower ACSM suggests greater diversity. As shown in Table III, our ensemble-of-costs guidance outperforms the average of 12 costs function when measured individually, demonstrating the advantage of multiple cost-guidance over a single one.

V. ABLATION STUDIES

Ensemble-of-costs. In this section we show the effect of ensemble-of-costs in Figure 7 and 8. As can be seen in Figure 7, as the number of guides increase, the overall Success Rate increases. This is because diverse cost functions capture different parameters and aspects across scenes, as explained

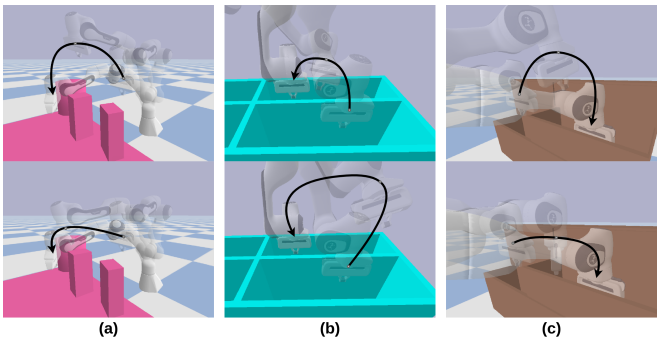

Fig. 9: **Multimodal trajectory predictions in** (a) tabletop scene, (b) merged cubby scene, and (c) dresser scene.

in Section III. Although these are specially designed cost functions, the idea of combining multiple costs to solve diverse scenes can be applied in any domain using custom cost functions. Figure 8 further emphasizes the contribution of each guide to the overall success rate.

Conditioning with start and goal. One primary condition of generating a trajectory is avoiding any abrupt manipulator motion that could cause serious damage in the real-world. We evaluate the effect of conditioning the trajectory with the start and goal. We defined Roughness as the L2 norm between adjacent waypoints. To evaluate this, we consider four metrics - Average Roughness (AR), Max Roughness Excluding Start and Goal (MRESG), Roughness between first two waypoints (RF2W), and Roughness between last two waypoints (RL2W). As shown in Table IV, conditional training generates significantly smoother trajectories.

VI. CONCLUSION & FUTURE WORK

In this paper, we propose diffusion models as a mechanism to learn a prior over kinematically valid trajectories complemented by scene-specific guidance through gradient-based cost functions directly at inference. We introduce an ensemble of cost-functions that capture scene-specific nuances across diverse scenes and improve the generated trajectories significantly. Our empirical results concretely point towards our approach's ability to effectively exploit the strengths of classical motion planners and the recent deep-learning-based planners, by showcasing remarkable generalization capability across diverse scenes, such as object manipulation, without any object-specific retraining. Moreover, EDMP is capable of generating diverse multimodal trajectories for a given scene, facilitating optimal trajectory selection based on specific downstream parameters. We believe that our research provides significant strides towards more generalizable planning with broad applications to robotic manipulation.

Future work. Despite showcasing impressive performance and generalization, EDMP still relies on a set of handcrafted cost functions to capture diversity across the scenes. Automating the design of cost-functions based on scene diversity could be an interesting future work.

ACKNOWLEDGMENT

We thank Adam Fishman for assisting with MπNets and providing valuable insights into benchmarking.

References

[1] A. Agarwal, B. Sen, S. Narayanan V, V. R. Mandadi, B. Bhowmick, and K. M. Krishna, "Approaches and challenges in robotic perception for table-top rearrangement and planning," *arXiv preprint arXiv:2205.04090*, 2022.

[2] V. R. Mandadi, K. Saha, D. Guhathakurta, *et al.*, "Disentangling planning and control for non-prehensile tabletop manipulation," in *2023 IEEE 19th International Conference on Automation Science and Engineering (CASE)*, 2023, pp. 1–6. DOI: 10.1109/CASE56687.2023.10260462.

[3] C. Zhou, B. Huang, and P. Fränti, "A review of motion planning algorithms for intelligent robots," *Journal of Intelligent Manufacturing*, vol. 33, no. 2, pp. 387–424, 2022.

[4] 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.

[5] 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.

[6] 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.

[7] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, 1968. DOI: 10.1109/TSSC.1968.300136.

[8] M. Likhachev, D. Ferguson, G. Gordon, A. Stentz, and S. Thrun, "Anytime dynamic a*: An anytime, replanning algorithm.," Jan. 2005, pp. 262–271.

[9] M. Likhachev, G. J. Gordon, and S. Thrun, "Ara*: Anytime a* with provable bounds on sub-optimality.," in *NIPS*, S. Thrun, L. K. Saul, and B. Schölkopf, Eds., MIT Press, 2003, pp. 767–774, ISBN: 0-262-20152-6. [Online]. Available: http://dblp.uni-trier.de/db/conf/nips/nips2003.html#LikhachevGT03.

[10] J. J. Kuffner and S. M. LaValle, "Rrt-connect: An efficient approach to single-query path planning," *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065)*, vol. 2, 995–1001 vol.2, 2000. [Online]. Available: https://api.semanticscholar.org/CorpusID:17124403.

[11] S. M. LaValle, "Rapidly-exploring random trees : A new tool for path planning," *The annual research report*, 1998. [Online]. Available: https://api.semanticscholar.org/CorpusID:14744621.

[12] M. Bangura, "Real-time model predictive control for quadrotors," vol. 47, Aug. 2014, pp. 11 773–11 780. DOI: 10.3182/20140824-6-ZA-1003.00203.

[13] T. Erez, K. Lowrey, Y. Tassa, V. Kumar, S. Kolev, and E. Todorov, "An integrated system for real-time model predictive control of humanoid robots," in *2013 13th IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, 2013, pp. 292–299. DOI: 10.1109/HUMANOIDS.2013.7029990.

[14] G. Williams, A. Aldrich, and E. A. Theodorou, "Model predictive path integral control using covariance variable importance sampling," *ArXiv*, vol. abs/1509.01149, 2015. [Online]. Available: https://api.semanticscholar.org/CorpusID:14146342.

[15] J. Jankowski, L. Brudermüller, N. Hawes, and S. Calinon, "Vp-sto: Via-point-based stochastic trajectory optimization for reactive robot behavior," *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 10 125–10 131, 2022. [Online]. Available: https://api.semanticscholar.org/CorpusID:252780630.

[16] G. Williams, P. Drews, B. Goldfain, J. M. Rehg, and E. A. Theodorou, "Aggressive driving with model predictive path integral control," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, 2016, pp. 1433–1440. DOI: 10.1109/ICRA.2016.7487277.

[17] 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.

[18] 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.

[19] 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.

[20] T. Osa, J. Pajarinen, G. Neumann, J. A. Bagnell, P. Abbeel, J. Peters, *et al.*, "An algorithmic perspective on imitation learning," *Foundations and Trends® in Robotics*, vol. 7, no. 1-2, pp. 1–179, 2018.

[21] T. Pearce, T. Rashid, A. Kanervisto, *et al.*, "Imitating human behaviour with diffusion models," *arXiv preprint arXiv:2301.10677*, 2023.

[22] T. Z. Zhao, V. Kumar, S. Levine, and C. Finn, "Learning fine-grained bimanual manipulation with low-cost hardware," *arXiv preprint arXiv:2304.13705*, 2023.

[23] J. Sohl-Dickstein, E. Weiss, N. Maheswaranathan, and S. Ganguli, "Deep unsupervised learning using nonequilibrium thermodynamics," in *Interna-*

tional conference on machine learning, PMLR, 2015, pp. 2256–2265.

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

[25] A. Ramesh, M. Pavlov, G. Goh, *et al.*, "Zero-shot text-to-image generation," in *International Conference on Machine Learning*, PMLR, 2021, pp. 8821–8831.

[26] A. Ramesh, P. Dhariwal, A. Nichol, C. Chu, and M. Chen, "Hierarchical text-conditional image generation with clip latents," *arXiv preprint arXiv:2204.06125*, vol. 1, no. 2, p. 3, 2022.

[27] 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.

[28] L. Ruan, Y. Ma, H. Yang, *et al.*, "Mm-diffusion: Learning multi-modal diffusion models for joint audio and video generation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 10 219–10 228.

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

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

[31] 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].

[32] 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.

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

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

[35] 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.

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

[37] 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.

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

[39] E. Coumans and Y. Bai, *Pybullet, a python module for physics simulation for games, robotics and machine learning*, http://pybullet.org, 2016–2021.

[40] 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.

[41] 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].

[42] 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.