EqM-MPD: Equivariant on-Manifold Motion Planning Diffusion

Anonymous authors
Paper under double-blind review

Abstract

Fast, reliable and versatile motion planning algorithms are essential for robots with many degrees of freedom in complex, dynamic environments. Diffusion models have been proposed as a faster alternative to classical planners by providing informative priors on distributions of trajectories. However, they are currently trained to overfit to environments with fixed object configurations and need to be re-trained when these conditions change. This limits applicability in tasks like robotic manipulation where environments change dynamically and initial configurations vary. We show that diffusion-guidance is not sufficient to adapt the model to large changes that can happen during execution or even from different initialization. Moreover, current approaches ignore the underlying topology of the state space thus requiring heavy guidance that dominates planning time and reduces efficiency dramatically. To address these, we propose a novel diffusion motion planner, *EqM-MPD* that *operates directly on the robot's state space manifold* and produces an *equivariant prior distribution* on trajectories. Our approach eliminates the need for retraining under rigid transformations. Moreover, our diffusion on state space manifold converges faster during guidance. We show that our approach achieves efficient, robust and generalizable planning that is especially useful for manipulation advancing beyond prior limitations.

1 Introduction

Motion planning is a crucial component of autonomous systems. The goal is to find smooth, feasible trajectories between given states while avoiding obstacles and respecting kinematic constraints. The problem is notoriously challenging for robots with many degrees of freedom in environments with intricate geometries and dynamic obstacles. Classical methods like sampling-based (Kavraki et al., 1996; Lavalle, 1998; Kuffner & LaValle, 2000; Gammell et al., 2014) and optimization-based approaches (Ratliff et al., 2009; Toussaint, 2009; Kalakrishnan et al., 2011) face issues such as computational intensity, non-smooth trajectories, and reliance on good initialization.

To overcome these limitations deep learning priors learned from previously successful plans have been proposed (Ichter et al., 2018; Wang et al., 2020; Bency et al., 2019) guiding optimization towards more promising regions and reducing planning time. Diffusion and score-based models (Sohl-Dickstein et al., 2015; Song & Ermon, 2019) in particular have shown promise in accelerating motion planning (Janner et al., 2022b; Carvalho et al., 2023a) by integrating efficient sampling from the diffusion prior with motion optimization costs through guidance (Dhariwal & Nichol, 2021a).

However, such models are usually trained from expert data to overfit to fixed object configurations rather than learning generalizable priors. This limits applicability in manipulation where the environment changes dynamically during execution or at initialization and full retraining is required. Performing diffusion-guidance during inference is not enough to adapt the model to both local and global changes. Moreover, current approaches largely ignore the topology of the state space which leads to ineffective training and heavier guidance that dominates the computation cost and reduces planning efficiency. To overcome these challenges, we propose a novel diffusion-based motion planning algorithm that *operates directly on the state space manifold* and *produces equivariant trajectory distributions*. Our contributions are two-fold:

1. Diffusion Motion Planning on the State Space Manifold: Our model accounts for the complex topology of the state space during all of the stages *sampling, denoising and guidance* by operating

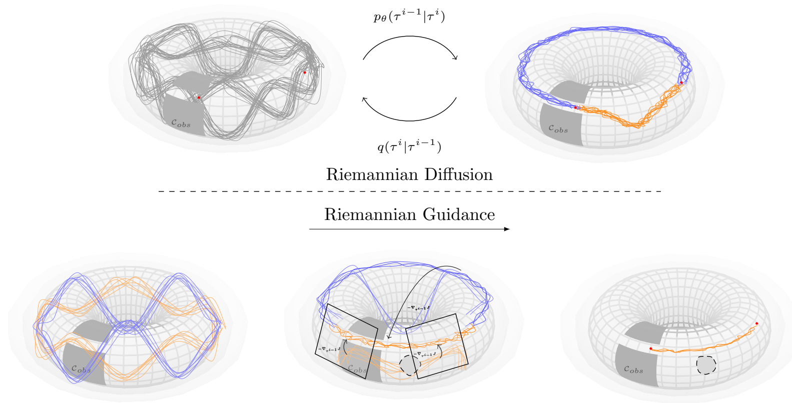


Figure 1: Riemannian Trajectory Diffusion Model. (Top Right to Left) Multi-scale Riemannian Diffusion is applied at the trajectory-level to create noisy state-space trajectories. During training, the network learns to predict this state-space noise. (Top Left to Right) During Riemannian Denoising $p_\theta(\tau^{i-1}|\tau^i)$ random trajectories are sampled on the state-space manifold, inpainted to start and goal states and provided to the network that iteratively predicts the noise on the state space until it creates feasible trajectories (e.g. avoid C-space obstacle in the figure). (Bottom Left to Right) Moreover, the denoising steps are interleaved with on-manifold guidance steps on user-defined costs $J(\tau)$ that can encode e.g. proximity to newly added obstacles (dotted circle in the figure). The model adapts the trajectories via Riemannian gradient descent ($\nabla_{\tau^{i-1}} J$) while denoising. This way the model can sample trajectories that are both kinematically-feasible (high prior) and cost-minimizing (high-likelihood). The blue and orange distributions show alternative paths between start and goal states for a continuous joint on which the diffusion model is trained. For a revolute joint, the branch that goes through joint limits will be discarded (Bottom Right).

on the embedded *hypertorus* instead of the *euclidean space*. This leads to stable training and faster inference with less guidance steps which was a bottleneck of previous methods. Diffusion on manifolds (Bortoli et al., 2022) is a promising direction to constrain the learned distributions. In our setting, diffusion operates at the trajectory-level so the representation has to account for continuity between the states too which motivates our representation in an embedding space instead of the quotient representation. We perform on-manifold guidance via Riemannian gradient descent which has not been explored for diffusion models on non-flat spaces to our knowledge. The diffusion steps are visualized in Fig. 1.

2. Equivariance via Positive-Negative Embedding: We propose a novel way to account for symmetry of the trajectories while handling symmetry-breaking effects which we term *positive-negative embedding*. In our setting, joint limits break full symmetry e.g. when the environment and the robot both rotate around the base. While the trajectories can be equivariant between joint limits, i.e. they can be adapted by a simple transformation of the sequence of states when the environment and start and goal states rotate, beyond the joint limits such a simple transformation of a feasible trajectory would result in an infeasible one. However, in such cases, we can utilize an alternative feasible trajectory that does not go through the joint limits to create an *equivariant pair* of trajectories. At most one of the two will be feasible for each environment/base angle. We properly query the expert to provide such a pair in an imitation learning setting and then we encode both trajectories into a common prior distribution in a *learned canonicalized environment* (Fig.2). After denoising and decanonicalization, the infeasible branch of the trajectories due to joint limits is discarded, thus breaking the symmetry. We perform experiments in cluttered environments and pick-and-place tasks to demonstrate the effectiveness and efficiency of our planner.

2 Related Work

Diffusion Models for Planning: In Janner et al. (2022a), diffusion models were combined with motion optimization via guidance for long-horizon trajectory generation. The idea is to predict all timesteps simultaneously by iteratively refining sampled trajectories. MPD (Carvalho et al., 2023a) built on this idea introducing guidance costs for manipulation. Recent surveys categorize current methods in motion planning (Ubukata et al., 2024) and beyond (Urain et al., 2024). MPD has been used for long range composition tasks conditioned on visual and language input (Liang et al., 2024) as well as in hierarchical control (Chen et al., 2024).

Diffusion models on Manifolds: Generalizating diffusion or score-based models on riemannian manifolds (Bortoli et al., 2022; Huang et al., 2022; Lou et al., 2023) has spiked interest recently. Jing et al. (2022) design a diffusion model in the intrinsic representation of the torus for conformal molecule generation. Our diffusion model, on the other hand, operates on the embedded hypertorus to account for trajectory continuity. (Leach et al., 2022) performs $SO(3)$ denoising for rotational alignment. Manifold preserving guidance for diffusion models is only discussed for linear subspaces of the data distribution (He et al., 2024; Chang et al., 2023). We perform guidance on the state space manifold via a Riemannian Gradient descent on the hypertorus.

Equivariance: Geometric deep learning (Bronstein et al., 2021) provides strong structural inductive biases to deep neural networks via the design of constrained layers (Cohen & Welling, 2016), or learned canonicalization (Kaba et al., 2022). In robotics, equivariant policy learning (Yang et al., 2023; Wang et al., 2024), provides a solution to the problem of learning from few demonstrations. Differently than our approach, the learned policies are in the gripper space and defer motion planning to a classical algorithm which needs to solve the hard problem of inverse kinematics while avoiding obstacles. Equivariance has been particularly useful for finding grasp poses for manipulation (Simeonov et al., 2022). Urain et al. (2023) uses equivariant diffusion models on $\mathrm{SE}(3)$ to learn distributions over poses; motion planning is deferred to guidance. Reversely, we learn trajectories on the state space and use the end-effector poses for guidance. Closer to our motivation EDGI (Brehmer et al., 2023) proposes Euclidean diffusion motion planning for problems with $\mathrm{SE}(3)$ symmetry. Differently, we perform diffusion, denoising and guidance on the state space manifold and instead of conditioning the model to high-dimensional raw observations we achieve equivariance via a novel *positive-negative embedding* framework.

3 METHOD

In this section we introduce our method - Equivariant on-Manifold Motion Planning Diffusion (EqMMPD). After formulating the problem we discuss how to perform diffusion, denoising and guidance on the state space manifold and how to incorporate equivariance in the system.

Let $\mathcal{E} = \{O_1, \cdots, O_K\}$ describe an environment with K collidable objects represented as 3d point clouds i.e. $O_i = \{p_i \in \mathbb{R}^3 | i \in [K_i]\}$. Let $\mathcal{C}$ be the configuration space of the robot, $q := (\theta_1, \cdots, \theta_n) \in \mathcal{C}$ a configuration consisting of n 1-DoF joints and S the state space of the robot i.e. $s := [q, \dot{q}] \in S$ including the joint velocities $\dot{q} \in T_q\mathcal{C}$. Given two points from the state space s_{start}, s_{goal} our goal is to find a smooth, collision-free trajectory represented as T ordered waypoints $\tau = (s_1, s_2, \cdots, s_T)$ with $s_1 = s_{start}, s_T = s_{goal}$ respecting physical, kinematic constraints and other user defined costs encoded in the functional $J : S^T \to \mathbb{R}_+$. We consider kinematic motion planning on the state space (which can account for non-holonomic constraints) and assume that a low-level controller will execute the state transition. Detailed preliminaries on Diffusion Models for Motion Planning are provided in Appendix 5.2.

It is important to distinguish the group $\mathrm{SO}(2) = \{R_\theta \in \mathbb{R}^{2\times 2} | R_\theta^T R_\theta = I, \det(R_\theta) = 1, \theta \in [-\pi, \pi)\}$ of 2d rotations from its parametrization $\theta \in [-\pi, \pi) \subset \mathbb{R}$ and from the space $S^1 = \{(\cos\theta, \sin\theta) | \theta \in [-\pi, \pi)\} \subset \mathbb{R}^2$. To simplify the notation we take directly the real values as elements of the lie algebra, since $\mathfrak{so}(2) \simeq \mathbb{R}$ and overload the exponential map as $\exp(\theta) = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix}$. If we limit the domain to $[-\pi, \pi)$ its inverse is $\log : \mathrm{SO}(2) \to \mathfrak{so}(2)$ with $\log(R) = atan2(R_{21}, R_{11})$ with $atan2(y, x) = 2arctan\frac{y}{\sqrt{x^2+y^2}+x} \in (-\pi, \pi)$ for $(x, y) \in S^1 - \{(-1, 0), (1, 0), (0, -1)\}$ and $(\pm 1, 0) \mapsto \pm\pi/2$, $(0, -1) \mapsto -\pi$. We also define: $Exp : \mathbb{R} \to S^1$, $Exp(\theta) = (\cos\theta, \sin\theta)$ which restricted to $[-\pi, \pi)$ has inverse $Log : S^1 \to \mathbb{R}$, $Log(x, y) = atan2(y, x)$. Also, we define the operator $\mathrm{mod}2\pi$ to map real values to the interval $[-\pi, \pi)$ as $\theta mod 2\pi = \theta - 2\pi\lfloor\frac{\theta+\pi}{2\pi}\rfloor$. We distinguish 1) the planning problem time $t \in T$, which we use as subscript and 2) the diffusion process time $i \in N$ which we use as superscript i.e. $\tau^i = (s_t^i)_{t\in T}$ is the i-times diffused trajectory from $\tau = \tau^0$; the denoised trajectory. We denote by $\tau_q, \tau_{\dot{q}}$ the configuration and velocity parts of the state space trajectory.

3.1 State Space Manifold Motion Planning Diffusion

In this section we develop our method. To illustrate the advantages we focus on a state space of n revolute joints, which is the case for many popular static manipulators. Our method extends seamlessly to prismatic joints, but the state space has trivial topology thus reducing to standard architectures. In our case, $S \subset T\mathbb{T}^n \simeq \mathbb{T}^n \times \mathbb{R}^n$. One option to solve the wrap-around problem is to operate directly on the quotient space $\mathbb{R}/2\pi\mathbb{Z}$. However, this (mod) representation creates discontinuous target trajectories for the neural network that operates in Euclidean space and makes the denoising particularly hard. To solve both the wrap-around and the trajectory continuity problem we propose a diffusion model that operates on the *embedding* of the state space. While the hypertorus $\mathbb{T}^n$ can be embedded in many ways to some euclidean space we select to embed each degree of freedom separately and work on the product manifold $(S^1)^n \subset \mathbb{R}^{2n}$ so that the representation remains disentangled. The states are represented in this embedding space as: $\tilde{s} = \{(\cos\theta_1, \sin\theta_1, \cdots, \cos\theta_n, \sin\theta_n), (\omega_1, \cdots, \omega_n)\}$, i.e. $\tilde{\tau} \in \mathbb{R}^{(2n+n)T}$. Since we still use the intrinsic coordinates (θ's) and not project to unit circles in $\mathbb{R}^2$, a description in $SO(2)^n$ is more natural due to the connection with the lie algebra $\mathfrak{so}(2)^n$ which we will use to discuss the three stages of *diffusion, denoising, guidance* next. When appropriate we leverage the isomorphism $SO(2) \simeq S^1$. In S^1 the variables are denoted as $\tilde{\tau}$, in $SO(2)$ as ${}^R\tau$ and in the lie algebra as τ.

3.1.1 Riemannian Diffusion and Denoising

Inspired by Leach et al. (2022) who perform diffusion on $SO(3)$ using the isotropic normal, we design an $SO(2)$ analogue, which gives rise to $\mathcal{IG}_{\mathrm{SO}(2)}(R, \sigma^2)$. We can get exact samples from the distribution using the lie algebra $\mathfrak{so}(2)$ as: $\theta \sim \mathcal{N}(\theta; \mu, \sigma^2), R = \exp(\theta)$. Then, we say that $R \sim \mathcal{IG}_{SO(2)}(R_\mu, \sigma^2)$. Our distribution is induced by pushforwarding the gaussian measure through the exponential map i.e. $\mu_{\mathcal{IG}_{SO(2)}}(A) := \mu_{\mathcal{N}(\mu,\sigma^2)}(\log A), A \subset SO(2)$. Since our distribution is contained in a 1d-submanifold of $\mathbb{R}^{2\times 2}$ it is not absolutely continuous w.r.t. the Lebesque measure, thus it does not have a density. We can study a corresponding density of some intrinsic parametrization of $SO(2)$, such as the lie algebra $\mathfrak{so}(2)$ which reveals the connection with the representation of Jing et al. (2022) at least in the diffusion stage. Proof is included in Appendix 5.1.

Lemma 3.1. *If $R \sim \mathcal{IG}_{SO(2)}(R_\mu, \sigma^2)$ then $\theta := log(R) \sim \mathcal{WN}(\theta; \mu mod 2\pi, \sigma^2)$, where $\mathcal{WN}(\theta; \mu, \sigma^2)$ is the wrapped Gaussian with location,uncertainty parameters $\mu \in [-\pi, \pi)$, $\sigma > 0$ and density: $\mathcal{WN}(\theta; \mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \sum_{k=-\infty}^{\infty} \exp\left(-\frac{(\theta-\mu-2\pi k)^2}{2\sigma^2}\right)$, $\theta \in [-\pi, \pi)$.*

We remind that our values are still on $SO(2)$ and not $[-\pi, \pi)$. There are some qualitative differences between $\mathcal{WN}(\mu, \sigma^2)$ and $\mathcal{IG}_{SO(2)}(R, \sigma^2)$. For example, while in $[-\pi, \pi)$ the wrapped normal does not necessarily have mean proportional to μ we can use the circular mean (Mardia & Jupp, 2009) $\mathbb{E}[\cos\theta + i\sin\theta] = e^{-\sigma^2}(\cos\mu + i\sin\mu)$ to prove that $\mathcal{IG}_{SO(2)}(R_\mu, \sigma^2)$ indeed has a mean proportional to R_μ: $\mathbb{E}_{R\sim\mathcal{IG}_{SO(2)}(R_\mu,\sigma^2)}[R] = e^{-\sigma^2} R_\mu$. Our representation has deep connections to directional statistics (Mardia & Jupp, 2009).

Standard diffusion models perform efficient sampling of the forward process by performing multiscale diffusion in a single step. We can show that $\mu_{\mathcal{IG}_{\mathrm{SO}(2)}(R,\sigma^2)}$ is closed under (measure) convolutions. It is known that $\phi_1 \sim \mathcal{WN}(\mu_1, \sigma_1^2)$ and $\phi_2 \sim \mathcal{WN}(\mu_2, \sigma_2^2)$ then $\phi = (\phi_1 + \phi_2)\mathrm{mod}2\pi \sim \mathcal{WN}((\mu_1 + \mu_2)\mathrm{mod}2\pi, \sigma_1^2 + \sigma_2^2)$ (Jammalamadaka et al., 2001). From this using the exponential map it is straightforward to show that if $R_{1/2} \sim \mathcal{IG}_{SO(2)}(R_{\mu,1/2}, \sigma_{1/2}^2)$ then $R_1 R_2 \sim \mathcal{IG}_{SO(2)}(R_{\mu,1} R_{\mu,2}, \sigma_1^2 + \sigma_2^2)$. For the n-fold product measure we overload the notation: $R \sim \mathcal{IG}_{SO(2)^n}(R_\mu, \sigma)$, where now R, R_μ, σ are n-dimensional lists. Whenever an operation (like multiplication) is between lists it is assumed to be pointwise.

Diffusion: Suppose the expert planner $P(s_{start}, s_{goal}, T, \mathcal{E})$ is queried to provide trajectories of successful plans $\tau^0 = (s_t^0)_{t\in[T]}$ with $s_t^0 \in [-\pi, \pi)^n \times \mathbb{R}^n$ which we gather in a dataset $\mathcal{D}$. Then, let $\tau^0 \sim q(\tau^0|\mathcal{E})$. Using the closure under convolutions we can perform forward sampling in one step. For $i \in [N]$, $R^{\epsilon_i} \sim \mathcal{IG}_{SO(2)^{nT}}(I, (1 - \bar{\alpha}_i))$ then for the configuration part of the trajectory τ_q we create the noisy trajectories as ${}^R\tau_q^i = R_i^\epsilon \exp(\sqrt{\bar{\alpha}_i}\tau_q^0)$. Then, $\tilde{\tau}^i \leftarrow ({}^R\tau_q^i, \tau_{\dot{q}}^i)$ where we use the isomorphism $SO(2) \simeq S^1$ for the configuration space while for the angular velocities we use standard Euclidean diffusion.

Denoising: We focus on the configuration space and follow Leach et al. (2022). Since $SO(2)$ is compact we sample uniformly (as $\theta \in U[-\pi, \pi), R = \exp\theta$): ${}^R\tau_q^N \sim U_{SO(2)^{nT}}$. We parametrize the inverse process as $p_w({}^R\tau_q^{i-1}|{}^R\tau^i, i) = \mathcal{IG}_{SO(2)^{nT}}(R_\mu({}^R\tau^i, i; w), \tilde{\beta}_i)$, where $R_\mu({}^R\tau^i, i; w) = \exp\left(\frac{\sqrt{\alpha_{i-1}}(1-\bar{\alpha}_{i-1})}{1-\bar{\alpha}_i} \log {}^R\tau_q^i\right) \exp\left(\frac{\sqrt{\bar{\alpha}_{i-1}}\beta_i}{1-\bar{\alpha}_i} \log {}^R\mu_w({}^R\tau^i, i)\right)$. We further reparametrize the second term to predict the noise directly instead of the mean,

$$^R\mu_w({}^R\tau^i, i) = \exp\left(\frac{1}{\sqrt{\bar{\alpha}_i}} \log^R \tau^i\right) \exp\left(-\frac{1}{\sqrt{1-\bar{\alpha}_i}} Log\epsilon_w^q(\tilde{\tau}^i, i)\right).$$

The network is $\epsilon_w : (S^1)^n \times \mathbb{R}^n \times [N] \to (S^1)^n \times \mathbb{R}^n$ and $\epsilon_w^q, \epsilon_w^{\dot{q}}$ the configuration and velocity parts. Here we leverage $SO(2) \simeq S^1$ to provide as input to the network the state-space trajectories in S^1 without unnecessarily increasing the input size. The network predicts the noise on S^1 which is easier than predicting in $SO(2)$ since the former only needs a normalization, while the later has orthogonality and determinant constraints that need to be satisfied too. Note that, if we move between ϵ and $(\cos\epsilon, \sin\epsilon)$ we do not actually need to normalize the output since this is done already by $arctan$. Our representation imposes continuity in the input of the network for continuous trajectories. A quotient representation that uses $\theta mod 2\pi$ to represent the input trajectories would not be continuous since the operator is not continuous.

Training Loss: $\mathcal{L}(w) = \mathbb{E}\left[d\left(Exp(\frac{1}{\sqrt{1-\bar{\alpha}_i}} \log R^{\epsilon_q}), \epsilon_w^q(\tilde{\tau}^i, i)\right) + \|\epsilon_{\dot{q}} - \epsilon_w^{\dot{q}}(\tilde{\tau}^i, i)\|_2^2\right]$ where the expectation is over $(i, \epsilon_{\dot{q}}, \tau^0) \sim \mathcal{U}(1, N) \times \mathcal{N}(0, I) \times q(\tau^0|\mathcal{E})$ and $\epsilon_q|i \sim \mathcal{IG}_{SO(2)^{nT}}(I, \sqrt{1-\bar{\alpha}_i})$. $\tilde{\tau}^i$ is described in the diffusion step and $d : \mathbb{T}^n \times \mathbb{T}^n \to \mathbb{R}_+$ is the chordal distance on torus here.

Riemannian Guidance: We observe that the update step in MPD during posterior sampling can be conceived as a two-stage process: 1. deterministic denoising: $\tau^{i-1} \leftarrow \mu_w(\tau^i, i)$ and 2. (noisy) gradient descent (around J^{i-1}): $\tau^{i-1} \leftarrow \tau^{i-1} - \eta_{i-1}\nabla J^{i-1}(\tau^{i-1}) + \tilde{\beta}_i\mathbf{z}, \mathbf{z} \sim \mathcal{N}(0, I)$. Thus, guidance tries to minimize the cost term J^{i-1} in a neighborhood around the denoised sample τ^i. However, this optimization uses gradient descent in the Euclidean space which requires clipping or small gradient steps to converge which is inefficient. First we note that costs are expressed on the trajectories which live on the state space manifold and not in the Euclidean space. Then, $c(\theta)$ with $\theta \in [-\pi, \pi)$ can be written in $SO(2)$ as $c(\theta) = c(\log(R_\theta)) = (c \circ \log)(R_\theta)$. By operating on the manifold we can also introduce more useful metrics that have been well studied on $SO(2)$ (Chirikjian & Kyatkin, 2016). For example, depending on the requirements we can either optimize the cost that accounts for the longer arclength $d(R_1, R_2) = \|\log R_1 - \log R_2\|_2$ as before, or costs that minimize the geodesic distance $d(R_1, R_2) = \|\log(R_1^T R_2)\|$. To optimize $J : SO(2)^n \to \mathbb{R}_+$ we will perform Riemannian gradient descent (RGD) (Boumal, 2023) on $SO(2)^n$. At iteration k, given gradient step α_k, RGD updates the current state as $R_{k+1} = R_k \exp(-\alpha_k R_k^{-1}\nabla J(R_k))$, where multiplication between lists is pointwise. We use the uniformity of tangent spaces on lie groups to simplify to an expression amenable to automatic differentiation. The proof is included in Appendix 5.1.

Lemma 3.2. *The RGD update on $SO(2)^n$ can be written:* $R_{k+1} = R_k \exp\left(-\alpha_k \nabla_\theta J(R_k \exp(\theta))|_{\theta=0}\right)$

All trajectories are on the manifold after each guidance step. These are subsequently given as input to the network for the next denoising step on the manifold. Reversely, denoising provides RGD with a trajectory on the manifold to initialize the updates. We visualize all steps in Fig. 1. We use similar costs as in Carvalho et al. (2023a) but reparametrized on the manifold as explained above.

3.2 Equivariant Motion Planning by Positive-Negative Embedding

In this section we adapt the planner to global changes in the environment. We propose to learn a more generalizable prior using symmetries in the trajectories. In particular, we discuss $SO(2)$ symmetries when the environment and the base both rotate. Local changes such as addition of new obstacles will still be handled by diffusion guidance, but when these local changes are combined with global transformations our generalizable treatment plus the diffusion guidance will be more effective. As in previous works Carvalho et al. (2023a) we avoid to condition the diffusion model on the environment to keep it lightweight. However, in this case the model is only a function of start and goal states s, g so how can we make the model predict feasible trajectories for a different environment $\mathcal{E}'$ and same s, g? We propose to *learn an equivariant frame* that aligns the environment to the environment that the model has been trained on. Moreover, the frame has to equivary smoothly with the rotations of the environment. For example, the PCA eigenvectors on the point cloud $\mathcal{E}$ do not satisfy the smoothness requirement due to sign ambiguity that creates

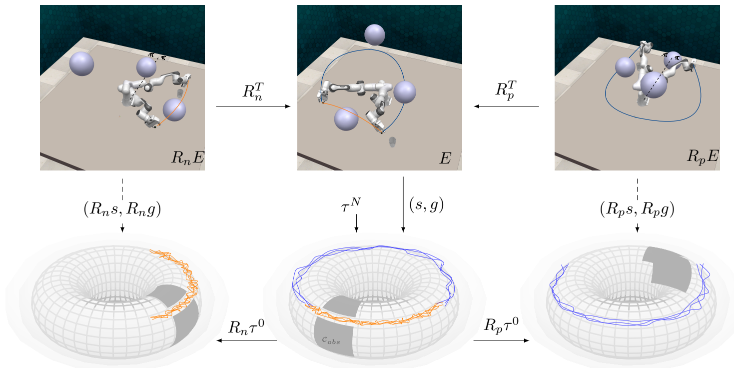


Figure 2: Equivariant Diffusion: Given an environment (represented as point clouds), we first canonicalize it with an equivariant frame (R_n, R_p in the figure), together with the start and goal states. We perform diffusion on the canonical triplet $(\mathcal{E}, s, g)$ and then decanonicalize the resulting trajectories by applying the same frame reversely. Following the arrows one can see that this is the desired output as if we applied the diffusion on one of $\{(R_n\mathcal{E}, R_n s, R_n g), (R_p\mathcal{E}, R_p s, R_p g)\}$ (depending on whether the sampled trajectory is orange or blue). In the figure you can see the joint limits (at $-\pi, \pi$ for simplicity). Only one branch of trajectories is feasible for a specific rotation of the environment. Both are encoded in the diffusion model which is trained on the canonical environment (middle) but the one that goes through the joint limits will be discarded after de-rotation.

many possible frames for each rotation. They are also very susceptible to noise. Instead we use a small $SO(3)$-equivariant network (Thomas et al., 2018) built using e3nn library (Geiger et al., 2022). The network f does not output a 3d frame only a single 3d vector $\vec{v} = f(\mathcal{E})$, from which we create $R_{\vec{v}} = \begin{pmatrix} \hat{\vec{v}}_x & -\hat{\vec{v}}_y & 0 \\ \hat{\vec{v}}_y & \hat{\vec{v}}_x & 0 \\ 0 & 0 & 1 \end{pmatrix}^T$, where $\hat{\vec{v}} = \frac{\vec{v}}{\|(\vec{v}_x, \vec{v}_y)\|}$. The canonical environment is then computed as $\mathcal{E}^c = R_{\vec{v}}\mathcal{E}$ where $R\mathcal{E} = (R\mathcal{O}_i)_{i\in[K]}$. It is easy to see that the canonical environment remains invariant to $SO(2)$-rotations. For all $R \in \mathrm{SO}(2)_z \subset \mathrm{SO}(3)$ ($SO(2)_z$ is isomorphic to $SO(2)$ but lifted along z to act on 3d vectors), due to equivariance of f we have $f(R\mathcal{E}) = Rf(\mathcal{E}) = R\vec{v}$. Then, $\hat{R\vec{v}} = R\hat{\vec{v}}$ and $R_{R\vec{v}} = R_{\vec{v}}R^T$ from which we get $(R_{\vec{v}}R^T)(R\mathcal{E}) = R_{\vec{v}}\mathcal{E} = \mathcal{E}^c$. Also the frame is a smooth function of the environment by construction. We denote the action of the rotation on the state space as $R_{\vec{v}}s = ((\log(R_{\vec{v}} \exp s_1^\theta), \omega_1), s_2, \cdots, s_n)$. An important intricacy arises due to joint limits. If $\tau \in P(s, g, T, \mathcal{E})$ (here inclusion means that the trajectory is kinematically feasible for the problem) then $R^T\tau \in P(Rs, Rg, T, R\mathcal{E})$ only for some rotations $R \in SO(2)$ as long as the resulting trajectory does not hit the joint limits. Any deterministic equivariant model (independently of whether the environment is conditioning the model or not) is doomed to predict the rotated trajectory even beyond joint limits thus overconstraining the problem.

However, if we assume feasibility between s, g i.e. that for a given environment, start and goal there is a feasible trajectory contained between the start and goal then even if we cannot get a feasible trajectory by decanonicalizing the predicted trajectory (call it *positive*) there is another (not necessarily unique) trajectory (call it *negative*) that starts and ends at the corresponding states but is feasible and the union of the ranges cover the whole $SO(2)$. At most one of the two is feasible for a given rotation of the environment (the ranges though depend on the trajectories). Thus, together they form an *equivariant pair* Fig. 2. The *negative* trajectory might differ a lot from the original depending on the environment. We cannot construct it as a transformation of the *positive* but we can query the planner appropriately to provide one such negative (if it exists) as we show next. If the joint limits are at $-\pi \leq \theta_{min} < \theta_{max} \leq \pi$ then for a fixed environment $\mathcal{E}$ and (s, g) we query:

$$\tau^p \sim R^T_{\theta_{min}-s_0} P(R_{\theta_{min}-s_0}s, R_{\theta_{min}-s_0}g, T, R_{\theta_{min}-s_0}\mathcal{E}) \tag{1a}$$

$$\tau^n \sim R^T_{\theta_{max}-s_0} P(R_{\theta_{max}-s_0}s, R_{\theta_{max}-s_0}g, T, R_{\theta_{max}-s_0}\mathcal{E}) \tag{1b}$$

With these queries we guarantee that the planner will return the total equivariant pair for which the union of the ranges cover $SO(2)$ (if one exists) and not for example two positive trajectories, since any feasible trajectory for the first equation necessarily crosses the joint limits in the second equation. Thus, the expert planner has to select the opposite path that connects s, g in the circle if one exists.

We will utilize the power of multimodality of diffusion models in order to embed both trajectories (that are feasible for *different* environments) in the same start and goal position for the *canonical environment* that the model is trained on thus creating an equivariant prior. I.e. $(s, g) \mapsto (R_{\vec{v}}s, R_{\vec{v}}g)$ and $(\tau^p, \tau^n) \mapsto (R_{\vec{v}}\tau^p, R_{\vec{v}}\tau^n)$. We can have multiple positive (and negative) trajectories from the same start and goal. Every query to a non deterministic expert (e.g. sampling-based) will give a different one. We query them in pairs. The distribution on the trajectories given s, g does not need to be uniform although having them balanced would help during generation. During training the diffusion model $\epsilon(\tilde{\tau}^i, i)$ will learn to denoise both $\tilde{\tau}^p, \tilde{\tau}^n$ in the canonical environment. The model is indifferent to joint limits. Moreover, we do not use the joint limit cost for guidance since we want the diffusion model, which is ignorant to the actual rotation of the environment, to sample trajectories for any rotated environment. In particular, it needs to sample trajectories that cross the joint limits for the canonical environment but when the actual rotation is applied in the output, the final trajectory does not cross joint limits anymore. So even if the adapted (s, g) fall into the joint limits, the model can predict a path between them that will become feasible in some other rotated environment after de-canonicalizing. During inference given $(s, g, \mathcal{E})$ we canonicalize the environment and start and goal and sample trajectories from $(f(\mathcal{E})s, f(\mathcal{E})g)$ including both branches. The costs are also canonicalized using $f(\mathcal{E})J$ and since all costs are assumed to be scalar-fields for rotation ($J(\tau) \in \mathbb{R}_+$), the particular action of $SO(2)$ is: $(f(\mathcal{E})J)(\tau) = J(f(\mathcal{E})^{-1}\tau)$. Guidance will be performed in both branches and the generated trajectories will be decanonicalized as $f(\mathcal{E})^T\tau$. The infeasible branch will be discarded after simulating the trajectory from the original s, g by checking which of the two crosses the joint limits. See Fig.2.

4 EXPERIMENTS

In this section, we verify our claims through simulation experiments, and answer the following questions: (1) Is our on-manifold diffusion model more effective in achieving lower costs and hence, better performance with fewer guidance steps? (2) Can we learn feasible plans that are also generalizable to different transformations of the environment?

Environments and Tasks We evaluate our algorithm on the 7-dof Emika Franka Panda arm that is deployed in two environments - the `PandaSpheres` in Isaac Gym, as described in Carvalho et al. (2023a), and a custom environment shown below in `CoppeliaSim` integrated with `RLBench`. The custom environment has spherical obstacles on the right, and a shelf in close vicinity on the left while being restricted by the table from below. The task in `PandaSpheres` environment is to generate feasible trajectories from random initial and final states, while minimizing an objective cost function, thus, providing a venue for fair baseline comparison and planner assessment. Our custom environment is more task-oriented, where the success criterion requires not only collision-free navigation but also planning/replanning feasible trajectories from random-initialized positions, picking up the cup, and placing it at a given position on the shelf while keeping it upright.

Algorithms and Baselines We compare our proposed algorithm's performance against the RRT-Connect + GPMP which is a sampling-based optimization planner, and MPD (Carvalho et al., 2023a) in the canonical environment (MPD Canonical). The MPD Canonical model operates over 5 cycles where each cycle consists of 25 diffusion steps, followed by 5 guidance steps, whereas our model executes the same 5-cycle process, but with only 2 guidance steps per cycle. Moreover, in the rotation augmentation test, the trajectories are first obtained in the canonical environment and then rotated back to the original environment. We also consider EQ-prior-guidance, which consists of our Equivariant On-manifold diffusion-based planner as Equivariant priors that are denoised for a total of 125 steps, followed by 10 guidance steps.

Metrics We chose 5 metrics to assess our performance with that of the baselines - (1) $\mathcal{S}$ denotes the success rate of the trajectory i.e., for a given trial, the success rate is one if at least one of the trajectories in the output batch is feasible. (2) $\mathcal{C}_s$ denotes the smoothness cost, which is a measure of how smooth on average, the trajectories are in the batch. While the Gaussian process promotes smoothness and thus, lowers $\mathcal{C}_s$ we compute it as the average of the sum of pairwise norm of the velocities of the trajectories, as done in Carvalho et al. (2023a) to keep the comparison fair. (3) $\mathcal{C}_p$ denotes the path length cost that is computed as the average of the sum of the pairwise norm of the joint angles. (4) $\mathcal{C}_b$ denotes the *best cost* (least) (sum of path length and smoothness costs) that a trajectory exhibited in the batch. (5) t - denotes the overall inference time that the planner took to output a batch of 50 trajectories.

Discussion The results in the `PandaSpheres` Environment are summarized in Table 1. The first three rows depict the performance of each of the algorithms in the canonical frame only, where the results are averaged over 10 initial and final configurations, randomly chosen. The last three

	$\mathcal{S} \uparrow$	$\mathcal{C}_s \downarrow$	$\mathcal{C}_p \downarrow$	$\mathcal{C}_b \downarrow$	$t \downarrow$	
RRTC+GPMP	1.0	–	8.1 ± 1.1	–	226.14 ± 13.4	$\mathbb{E}_{q_i,q_f}[.]$
MPD Canonical	1.0 ± 0.0	7.6 ± 3.41	6.5 ± 2.74	11.54 ± 6.07	23.12 ± 1.1	
EQ-MMPD	1.0 ± 0.0	8.7 ± 1.7	7 ± 1.45	12.6 ± 2.92	9.98 ± 0.9	
MPD Canonical	0.43 ± 0.22	8.77 ± 2.82	4.65 ± 1.49	11.15 ± 3.68	–	$\mathbb{E}_{q_i,q_f}\mathbb{E}_g[.]$
EQ-MMPD	0.97 ± 0.03	8.61 ± 1.57	7.01 ± 0.9	12.9 ± 2.31	–	
EQ-Prior-Guidance	0.85 ± 0.17	9.3 ± 1.46	7.51 ± 1.83	13.96 ± 3.02	10.31 ± 1.85	

Table 1: Metrics are reported as ($\mu \pm \sigma$) with $\uparrow$ indicate that higher values are better, and with $\downarrow$ indicate that lower values are better.

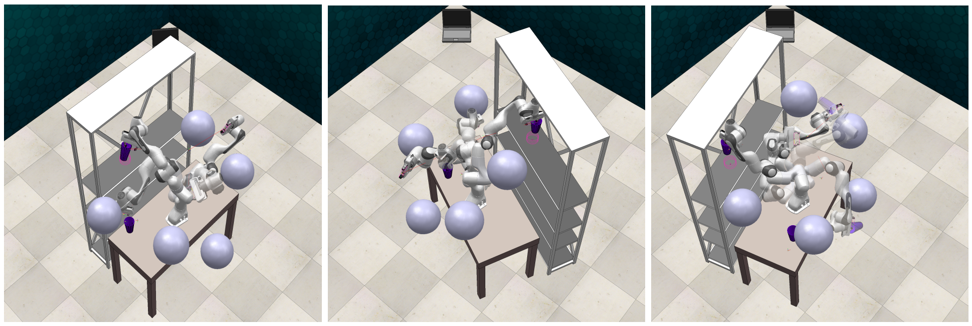


Figure 3: The left figure illustrates the canonical environment. The middle figure illustrates a random rotation transformation of the environment. The right figure illustrates a random rotation of the environment, while the spherical obstacles are slightly perturbed away from their usual positions (global+local).

rows depict the performance of each of the algorithms with 10 randomly sampled initial and final configurations, but each consisting of 72 rotation transformations i.e., 5° increments in the range of $[0 - 360)^{\circ}$ of the environment. We can infer from the table that (1) EQ-MMPD compares similarly to MPD in the canonical frame, albeit with lower variations and lower guidance steps. (2) EQ-MMPD obtains an overall success rate of 98%, while the MPD canonical achieves about 43%, due to the lack of the *negative trajectory*. Although our path length appears to be high, this is precisely because our formulation can predict the *negative* trajectory in cases where MPD Canonical fails. We do so with fewer guidance steps as compared to MPD canonical, thereby achieving a more efficient planner with faster inference time. (3) Our performance across all metrics is better than the EQ-Prior-Guidance model, even though the total number of denoising steps and guidance steps remains the same. This empirically shows us that the optimization is more effective when *interleaved* with smaller chunks of denoising steps over multiple loops.

RLBench experiments: Traditional motion planners often infer the configuration-space trajectory based on the end-effector path by solving the inverse-kinematics problem at each stage, with some suboptimal velocity profile like constant velocity. This may be particularly disadvantageous in a cluttered environment. We base our custom environment in this regard by making the workspace cluttered with spherical obstacles and following a *task-based* theme, that allows for replanning between two waypoints. The arm configuration and the cup position (within the workspace in the front) are randomly initialized in the beginning, and the goal is fixed on a shelf in close vicinity to the arm. Fig. 3 describes the simulation environment, along with two variations. In all three cases, we can see that the proposed EqMMPD succeeds in finding a feasible trajectory, and hence completing the task by succeeding with an average of 8.2 out of 10 success rate, within an average inference time of $9.12s$. The inverse-kinematics motion planner in RLBench has an average of 4.6 out of 10 success rate, with an average inference time of $12.32s$. Details on real-world experiments are provided in the supplementary.

Conclusion: In this paper we proposed a novel diffusion motion planner that is topologically and symmetry informed. It operates directly on the state space manifold during all stages of diffusion, denoising and guidance. To achieve that we introduced a novel diffusion model operating on the embedded hypertorus and guided via Riemannian gradient descent. Additionally, it encodes symmetric trajectories in an equivariant prior that accounts for symmetry-breaking effects via a novel *positive-negative embedding*. Our experimental results demonstrate that the proposed method outperforms existing approaches in terms of planning efficiency and generalization.

REFERENCES

Mayur J. Bency, Ahmed Hussain Qureshi, and Michael C. Yip. Neural path planning: Fixed time, near-optimal path generation via oracle imitation. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2019, Macau, SAR, China, November 3-8, 2019*, pp. 3965–3972. IEEE, 2019. doi: 10.1109/IROS40897.2019.8968089. URL `https://doi.org/10.1109/IROS40897.2019.8968089`.

Valentin De Bortoli, Emile Mathieu, Michael John Hutchinson, James Thornton, Yee Whye Teh, and Arnaud Doucet. Riemannian score-based generative modelling. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho (eds.), *Advances in Neural Information Processing Systems*, 2022. URL `https://openreview.net/forum?id=oDRQGo8I7P`.

Nicolas Boumal. *An Introduction to Optimization on Smooth Manifolds*. Cambridge University Press, 2023.

Johann Brehmer, Joey Bose, Pim de Haan, and Taco S Cohen. Edgi: Equivariant diffusion for planning with embodied agents. In *Advances in Neural Information Processing Systems*, volume 36, pp. 63818–63834. Curran Associates, Inc., 2023.

Michael M. Bronstein, Joan Bruna, Taco Cohen, and Petar Veličković. Geometric Deep Learning: Grids, Groups, Graphs, Geodesics, and Gauges. *arXiv preprint arXiv:2104.13478*, 2021.

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

João Carvalho, An T. Le, Mark Baierl, Dorothea Koert, and Jan Peters. Motion planning diffusion: Learning and planning of robot motions with diffusion models. In *IROS*, pp. 1916–1923, 2023b. doi: 10.1109/IROS55552.2023.10342382.

Junwoo Chang, Hyunwoo Ryu, Jiwoo Kim, Soochul Yoo, Joohwan Seo, Nikhil Prakash, Jongeun Choi, and Roberto Horowitz. Denoising heat-inspired diffusion with insulators for collision free motion planning. *arXiv preprint arXiv:2310.12609*, 2023.

Chang Chen, Fei Deng, Kenji Kawaguchi, Caglar Gulcehre, and Sungjin Ahn. Simple hierarchical planning with diffusion, 2024.

G.S. Chirikjian and A.B. Kyatkin. *Harmonic Analysis for Engineers and Applied Scientists: Updated and Expanded Edition*. Dover Books on Mathematics. Dover Publications, 2016. ISBN 9780486795645. URL `https://books.google.com/books?id=EO1lDAAAQBAJ`.

Taco Cohen and Max Welling. Group equivariant convolutional networks. In Maria Florina Balcan and Kilian Q. Weinberger (eds.), *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pp. 2990–2999, New York, New York, USA, 20–22 Jun 2016. PMLR. URL `https://proceedings.mlr.press/v48/cohenc16.html`.

Prafulla Dhariwal and Alex Nichol. Diffusion models beat gans on image synthesis, 2021a.

Prafulla Dhariwal and Alexander Quinn Nichol. Diffusion models beat gans on image synthesis. In *NeurIPS*, pp. 8780–8794, 2021b.

Jonathan D. Gammell, Siddhartha S. Srinivasa, and Timothy D. Barfoot. Informed rrt*: Optimal sampling-based path planning focused via direct sampling of an admissible ellipsoidal heuristic. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems, Chicago, IL, USA, September 14-18, 2014*, pp. 2997–3004. IEEE, 2014. doi: 10.1109/IROS.2014.6942976.

Mario Geiger, Tess Smidt, Alby M., Benjamin Kurt Miller, Wouter Boomsma, Bradley Dice, Kostiantyn Lapchevskyi, Maurice Weiler, Michał Tyszkiewicz, Simon Batzner, Dylan Madisetti, Martin Uhrin, Jes Frellsen, Nuri Jung, Sophia Sanborn, Mingjian Wen, Josh Rackers, Marcel Rød, and Michael Bailey. e3nn/e3nn: 2022-04-13, April 2022. URL `https://doi.org/10.5281/zenodo.6459381`.

Yutong He, Naoki Murata, Chieh-Hsin Lai, Yuhta Takida, Toshimitsu Uesaka, Dongjun Kim, Wei-Hsiang Liao, Yuki Mitsufuji, J Zico Kolter, Ruslan Salakhutdinov, and Stefano Ermon. Manifold preserving guided diffusion. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=o3BxOLoxm1.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *NeurIPS*. Curran Associates Inc., 2020. ISBN 9781713829546.

Chin-Wei Huang, Milad Aghajohari, Joey Bose, Prakash Panangaden, and Aaron Courville. Riemannian diffusion models. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho (eds.), *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=ecevn9kPm4.

Brian Ichter, James Harrison, and Marco Pavone. Learning sampling distributions for robot motion planning. In *IEEE ICRA*, 2018.

S.R. Jammalamadaka, A. Sengupta, and A. Sengupta. *Topics in Circular Statistics*. Series on multivariate analysis. World Scientific, 2001. ISBN 9789812779267. URL https://books.google.com/books?id=sKqWMGqQXQkC.

Michael Janner, Yilun Du, Joshua Tenenbaum, and Sergey Levine. Planning with diffusion for flexible behavior synthesis. In *ICML*, 2022a.

Michael Janner, Yilun Du, Joshua Tenenbaum, and Sergey Levine. Planning with diffusion for flexible behavior synthesis. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 9902–9915. PMLR, 17–23 Jul 2022b.

Bowen Jing, Gabriele Corso, Jeffrey Chang, Regina Barzilay, and Tommi Jaakkola. Torsional diffusion for molecular conformer generation. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems*, volume 35, pp. 24240–24253. Curran Associates, Inc., 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/994545b2308bbbbc97e3e687ea9e464f-Paper-Conference.pdf.

Sékou-Oumar Kaba, Arnab Kumar Mondal, Yan Zhang, Yoshua Bengio, and Siamak Ravanbakhsh. Equivariance with learned canonicalization functions. In *NeurIPS 2022 Workshop on Symmetry and Geometry in Neural Representations*, 2022. URL https://openreview.net/forum?id=pVD1k8ge25a.

M. Kalakrishnan, S. Chitta, E. Theodorou, P. Pastor, and S. Schaal. Stomp: Stochastic trajectory optimization for motion planning. In *IEEE International Conference on Robotics and Automation*, 2011.

L.E. Kavraki, P. Svestka, J.-C. Latombe, and M.H. Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Transactions on Robotics and Automation*, 1996. doi: 10.1109/70.508439.

J.J. Kuffner and S.M. LaValle. Rrt-connect: An efficient approach to single-query path planning. In *IEEE ICRA*, 2000. doi: 10.1109/ROBOT.2000.844730.

Steven M. Lavalle. Rapidly-exploring random trees: A new tool for path planning, 1998.

Adam Leach, Sebastian M Schmon, Matteo T. Degiacomi, and Chris G. Willcocks. Denoising diffusion probabilistic models on SO(3) for rotational alignment. In *ICLR 2022 Workshop on Geometrical and Topological Representation Learning*, 2022. URL https://openreview.net/forum?id=BY88eBbkpe5.

Zhixuan Liang, Yao Mu, Hengbo Ma, Masayoshi Tomizuka, Mingyu Ding, and Ping Luo. Skilldiffuser: Interpretable hierarchical planning via skill abstractions in diffusion-based task execution. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 16467–16476, June 2024.

Aaron Lou, Minkai Xu, Adam Farris, and Stefano Ermon. Scaling riemannian diffusion models. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL `https://openreview.net/forum?id=FLTg8uA5xI`.

K.V. Mardia and P.E. Jupp. *Directional Statistics*. Wiley Series in Probability and Statistics. Wiley, 2009. ISBN 9780470317815. URL `https://books.google.com/books?id=PTNiCm4Q-M0C`.

Alexander Quinn Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models. In *ICML*. PMLR, 2021.

Nathan Ratliff, Matt Zucker, J. Andrew Bagnell, and Siddhartha Srinivasa. Chomp: Gradient optimization techniques for efficient motion planning. In *IEEE International Conference on Robotics and Automation*, 2009. doi: 10.1109/ROBOT.2009.5152817.

Anthony Simeonov, Yilun Du, Andrea Tagliasacchi, Joshua B. Tenenbaum, Alberto Rodriguez, Pulkit Agrawal, and Vincent Sitzmann. Neural descriptor fields: Se(3)-equivariant object representations for manipulation. In *2022 International Conference on Robotics and Automation, ICRA 2022, Philadelphia, PA, USA, May 23-27, 2022*, pp. 6394–6400. IEEE, 2022. doi: 10.1109/ICRA46639.2022.9812146. URL `https://doi.org/10.1109/ICRA46639.2022.9812146`.

Jascha Sohl-Dickstein, Eric A. Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *ICML*. JMLR.org, 2015.

Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. In *NeurIPS*, 2019.

Nathaniel Thomas, Tess Smidt, Steven Kearnes, Lusann Yang, Li Li, Kai Kohlhoff, and Patrick Riley. Tensor field networks: Rotation- and translation-equivariant neural networks for 3d point clouds, 2018. URL `https://arxiv.org/abs/1802.08219`.

Marc Toussaint. Robot trajectory optimization using approximate inference. In *ICML*. Association for Computing Machinery, 2009. ISBN 9781605585161. doi: 10.1145/1553374.1553508.

Toshihide Ubukata, Jialong Li, and Kenji Tei. Diffusion model for planning: A systematic literature review, 2024. URL `https://arxiv.org/abs/2408.10266`.

Julen Urain, Niklas Funk, Jan Peters, and Georgia Chalvatzaki. Se(3)-diffusionfields: Learning smooth cost functions for joint grasp and motion optimization through diffusion. In *IEEE ICRA*, 2023.

Julen Urain, Ajay Mandlekar, Yilun Du, Mahi Shafiullah, Danfei Xu, Katerina Fragkiadaki, Georgia Chalvatzaki, and Jan Peters. Deep generative models in robotics: A survey on learning from multimodal demonstrations, 08 2024.

Dian Wang, Stephen Hart, David Surovik, Tarik Kelestemur, Haojie Huang, Haibo Zhao, Mark Yeatman, Jiuguang Wang, Robin Walters, and Robert Platt. Equivariant diffusion policy. In *8th Annual Conference on Robot Learning*, 2024. URL `https://openreview.net/forum?id=wD2kUVLT1g`.

Jiankun Wang, Wenzheng Chi, Chenming Li, Chaoqun Wang, and Max Q.-H. Meng. Neural rrt*: Learning-based optimal path planning. *IEEE T-ASE*, 2020. doi: 10.1109/TASE.2020.2976560.

Cheng-Fu Yang, Haoyang Xu, Te-Lin Wu, Xiaofeng Gao, Kai-Wei Chang, and Feng Gao. Planning as in-painting: A diffusion-based embodied task planning framework for environments under uncertainty, 2023.

5 APPENDIX

5.1 LEMMAS AND PROOFS

Lemma 5.1. *If $R \sim \mathcal{IG}_{SO(2)}(R_\mu, \sigma^2)$ then $\theta := log(R) \sim \mathcal{WN}(\theta; \mu mod 2\pi, \sigma^2)$, where $\mathcal{WN}(\theta; \mu, \sigma^2)$ is the wrapped Gaussian with location,uncertainty parameters $\mu \in [-\pi, \pi)$, $\sigma > 0$ and density:*

$$\mathcal{WN}(\theta; \mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \sum_{k=-\infty}^{\infty} \exp\left(-\frac{(\theta - \mu - 2\pi k)^2}{2\sigma^2}\right),\ \theta \in [-\pi, \pi)$$

Proof. Let $\phi \sim \mathcal{N}(\mu, \sigma^2)$ be the value such that $R_\phi \sim \mathcal{IG}_{SO(2)}(R_\mu, \sigma^2)$. Then, $\theta = \log R_\phi = \log \exp \phi = \phi \mod 2\pi$. For $-\pi \leq a < b < \pi, \mathbb{P}[\theta \in (a, b)] = \mathbb{P}[\phi \in \cup_{k\in\mathbb{Z}}(a + 2\pi k, b + 2\pi k)] = \sum_{k\in\mathbb{Z}} \mathbb{P}[\phi \in (a + 2\pi k, b + 2\pi k)] = \sum_{k\in\mathbb{Z}} \int_{a+2\pi k}^{b+2\pi k} \mathcal{N}(\phi; \mu, \sigma^2) d\phi = \sum_{k\in\mathbb{Z}} \int_a^b \mathcal{N}(\phi; \mu + 2\pi k, \sigma^2) d\phi = \int_a^b \sum_k \mathcal{N}(\phi; \mu + 2\pi k, \sigma^2) d\phi$. Set $a, b = -\pi, x$. Differentiation gives the result. $\square$

Lemma 5.2. *The RGD update on $SO(2)^n$ can be written: $R_{k+1} = R_k \exp\left(-\alpha_k \nabla_\theta J(R_k \exp(\theta))|_{\theta=0}\right)$*

Proof. We need to compute the gradient $\nabla J(R) \in T_R SO(2)^n$ in a form that we can use automatic differentiation. We note that $T_R SO(2)^n = \{RX | X \in \mathfrak{so}(2)^n\}$ i.e. for Lie groups all tangent spaces can be computed by a left-action of the group on the lie algebra which is the tangent space at identity. The gradient $\nabla J : SO(2)^n \to TSO(2)^n$ is a vector field on the manifold. At $R \in SO(2)^n$ it is defined as the unique vector satisfying $dJ_R(V) = g_R(\nabla J(R), V), \forall V \in T_R SO(2)^n$, where $dJ_R : T_R SO(2)^n \to \mathbb{R}$ is the differential $dJ_R(V) = \frac{d}{dt}\big|_{t=0} J(R \exp(tR^{-1}V))$ and $g_R : T_R SO(2)^n \times T_R SO(2)^n \to \mathbb{R}$ is a Riemannian metric which is our setting will be $g_R(U, V) = \sum_{i=1}^n \frac{1}{2} Tr(U_i^T V_i), U, V \in T_R SO(2)^n$ which is the metric induced by the inner product in $\mathfrak{so}(2)^n$ i.e., $\langle U, V\rangle = \sum_{i=1}^n \frac{1}{2} Tr(U_i^T V_i), U, V \in \mathfrak{so}(2)^n$. Since $\nabla J(R) \in T_R SO(2)^n$ we set $\nabla J(R) = R\text{grad}J(R)$ where $\text{grad}J(R) \in \mathfrak{so}(2)^n$ and by substituting $dJ_R(RX) = \langle \text{grad}J(R), X\rangle, \forall X \in \mathfrak{so}(2)^n$. Thus, $R^{-1}\nabla J(R) = \text{grad}J(R)$ and $R_{k+1} = R_k \exp(-\alpha_k \text{grad}J(R))$. The problem is simplified to computing $\text{grad}J(R)$. Since $SO(2)$ has a single generator $g = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}$, $SO(2)^n$ has n generators $G = (G_1, \cdots, G_n)$ where G_i is the generator of $SO(2)$ lifted to $SO(2)^n$. Then $\text{grad}J(R) = \sum_{i=1}^n \beta(R)_i G_i$. Substituting G_i into the differential gives we get $dJ_R(RG_i) = \langle gradJ(R), G_i\rangle = \beta(R)_i$, since $G_i^2 = I$. Then, $\beta(R)_i = \frac{d}{d\theta_i}\Big|_{\theta_i=0} J(R \exp(\theta_i G_i))$.

$$R_{k+1} = R_k \exp\left(-\alpha_k \sum_{i=1}^n \beta(R)_i G_i\right)$$

$$R_{k+1} = R_k \exp\left(-\alpha_k \sum_{i=1}^n \left(\frac{d}{d\theta_i}\Big|_{\theta_i=0} J(R_k \exp(\theta_i G_i))\right) G_i\right)$$

Observe that every G_i embeds g to a different dimension, so we can also write $\text{grad}J(R) = (\beta(R)_1 g, \ldots, \beta(R)_n g)_{i=1}^n$. Now if we gather $\theta = (\theta_1, \ldots, \theta_n)$ we get:

$$R_{k+1} = R_k \exp\left(-\alpha_k \nabla_\theta J(R_k \exp(\theta))|_{\theta=0}\right) = R_k \exp\left(-\alpha_k \nabla_\theta (J \circ \exp)(\theta_k)\right)$$

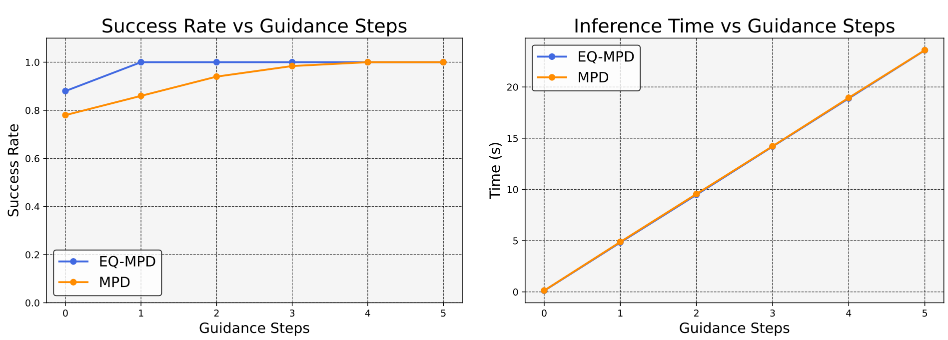
$\square$

5.2 PRELIMINARIES ON DIFFUSION MODELS FOR MOTION PLANNING

Diffusion models transform a trajectory from the data distribution $\tau^0 \sim q(\tau^0|\mathcal{E})$ by iteratively applying a time-dependent Markov diffusion process (typically parameter-free e.g. Gaussian with fixed variance). I.e. for all $i \in [N]$, $q(\tau^i|\tau^{i-1}, i) = \mathcal{N}(\tau^i; \sqrt{1-\beta_i}\tau^{i-1}, \beta_i I)$ where β_i is the noise schedule (Nichol & Dhariwal, 2021). If τ^0 live in Euclidean space, the distribution at time step i is again Gaussian and permits sampling in one step without running the forward diffusion process i.e.,

$q(\tau^i|\tau^0, i) = \mathcal{N}(\tau^i; \sqrt{\bar{\alpha}_i}\tau^0, (1-\bar{\alpha}_i)I),\ \alpha_i = 1-\beta_i, \bar{\alpha}_i = \prod_{j=1}^{i} \alpha_j$. Diffusion models approximate the inverse (denoising) process, $q(\tau^{i-1}|\tau^i, i), i \in [N]$, that transforms noisy trajectories back to the data distribution with a parametrized Gaussian (usually with parameter-free variance for stability): $p_w(\tau^{i-1}|\tau^i, i) = \mathcal{N}(\tau^{i-1}; \mu_i = \mu_w(\tau^i, i), \Sigma_i = \tilde{\beta}_i I),\ \tilde{\beta}_i = \beta_i(1-\bar{\alpha}_{i-1})/(1-\bar{\alpha}_i), i \in [N]$. The terminal condition is set to $p(\tau^N) = \mathcal{N}(0, I)$. Since the posterior mean has a closed form for this parametrization, DDPM (Ho et al., 2020) proposed to parametrize the noise term directly i.e., $\mu_w(\tau^i, i) = \frac{1}{\sqrt{\alpha_i}}\left(\tau^i - \frac{1-\alpha_i}{\sqrt{1-\bar{\alpha}_i}}\epsilon_w(\tau^i, i)\right)$. During training we do not have access to $q(\tau^0|\mathcal{E})$ but only to an empirical distribution $\tilde{q}(\tau^0|\mathcal{E})$ of samples. Typically, there is an expert planner P that provides these samples as $\tau^0 = P(s_{start}, s_{goal}, T, \mathcal{E})$. The parameters θ are optimized by injecting noise of different scales to the original trajectory and predicting the noise level directly via the loss: $\mathcal{L}(w) = \mathbb{E}_{(i,\epsilon,\tau^0)\sim\mathcal{U}(1,N)\times\mathcal{N}(0,I)\times\tilde{q}(\tau^0|\mathcal{E})}[\|\epsilon - \epsilon_w(\tau^i = \sqrt{\bar{\alpha}_i}\tau^0 + \sqrt{1-\bar{\alpha}_i}\epsilon, i)\|_2^2]$ This loss is an upper bound to the negative log likelihood of the data $\mathbb{E}_{\tau^0\sim\tilde{q}(\tau^0|\mathcal{E})}[-\log p_w(\tau^0)]$. Planning-as-inference poses motion planning as posterior sampling. If O is random variable describing our task desiderata $p(\tau|O) \propto p(O|\tau)p_w(\tau) \propto \exp(-\mathcal{J}(\tau))p_w(\tau)$, where the likelihood depends on user defined costs and constraints via $\mathcal{J}(\tau) = \sum_j \lambda_j c_j(\tau)$ with $\lambda_j > 0$. These include smoothness constraints, joint limits, end-effector costs etc. In principle the cost could also depend on the diffusion step aka $J^i(\tau), i \in [N]$. Akin to classifier-based guidance (Dhariwal & Nichol, 2021b), the authors in (Carvalho et al., 2023b) show that (under some assumptions) sampling this posterior can be done directly during sampling the prior trajectory by modifying the denoised trajectory in the diffusion model at each step $i \in [N]$. First the sample $\tau^N \sim \mathcal{N}(0, I)$ is drawn then N denoising steps are applied sequentially where the diffusion neural network at each step predicts $\epsilon_w(\tau^i, i)$ and the trajectory is updated as: $\tau^{i-1} = \mu_w(\tau^i, i) - \sum_j \lambda_j \nabla_\tau c_j^{i-1}(\tau)|_{\tau=\mu_w(\tau^i,i)} + \tilde{\beta}_i \mathbf{z},\ \mathbf{z} \sim \mathcal{N}(0, I)$. Thus, by a simple modification to the prior sampling procedure, the generated trajectories have both a high prior i.e. they are kinematically feasible and a high likelihood i.e. they are cost minimizing. To ensure that the final trajectory also starts and ends at the right states the authors inpaint as $\tau_0^i = s_{start}, \tau_T^i = s_{goal}\ \forall i \in [N]$. During planning many trajectories are sampled and the one that minimizes the cost is selected. Guidance is the main computational bottleneck in this planning process.

5.3 Ablation - Guidance steps



The optimization-based guidance is crucial in guiding the denoised trajectories to low-cost regions while being robust to small changes in the environment. The per-step cost is same between EQ-MMPD and the MPD Canonical model, as depicted in the plot However, from the plot we see that EqMMPD obtains the best results with 2 guidance steps, beyond which there is no improvement while MPD needs 5 guidance steps to achieve the same success rate which more than doubles the inference time. We chose to have 2 guidance steps throughout our experiments.

5.4 Dataset Generation and Training

The training data is obtained from *apriori* chosen canonical environment only. While MPD is trained with 200 contexts (i.e., 200 different initial and final states), each consisting of 50 trajectories over 64 waypoints or horizon length, we train EQ-MMPD with 200 contexts, but with only 25 trajectories for the positive and negatives. The trajectory generation happens by querying the RRT-connect first, with the specified initial and final configurations (For Eq-MMPD, we do as described in Eq. 1), and then smoothening out the *sharp trajectories* using the GPMP optimizer. The optimizer ensures that the velocities are zero at the start and end configurations, and based on the initial constant-velocity guess, generates smoothened velocities and hence, smoothened trajectories.

5.5 REAL-WORLD EXPERIMENTS

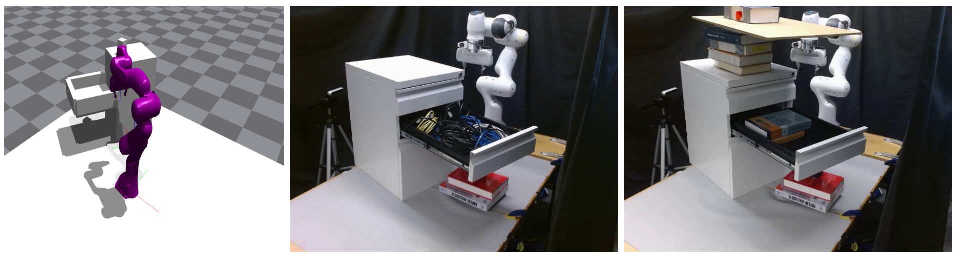


Figure 4: Left figure depicts the canonical environment in the simulation (random $SO(2)$ rotation about the base by $148.6°$. The middle figure is a real-world replication, with a rotation of $17°$, with the place goal being on top of the cabinet, with the drawer full. The figure on the right is the same but with additional obstruction added from the top during test time, with the place goal being inside the drawer (partially free). In all three environments, the pick goal is under the opened drawer.

To test our motion planner on real-world tasks, we consider a pick-place experiment in a relatively cluttered environment, as shown in Fig. 4. The goal is to pick an object from underneath the middle drawer of a cabinet and place it on top of the cabinet, or inside the drawer. The environment is obstructed from the top during test time (right figure in Fig. 4, with reduced workspace. We train the diffusion model on a simpler canonical environment with only the cabinet present, using only 50 contexts with 25 trajectories each. The rotation for the canonical environment is randomly chosen to be $148.6°$ about the base joint, while during testing, the rotation is randomly chosen to be $\sim 17°$, with local changes to the cabinet itself with test-time obstructions from the top. The `Emika Franka Panda` arm is run using the *effort trajectory controller*, to which we provide a subsampled set of state waypoints. We find that our motion planner is able to generalize well to these transformed versions of the environment, producing feasible (i.e., collision-free and smooth) trajectories to successfully execute the task. The video recordings for the hardware experiments are provided here[1].

5.6 TRAINING DETAILS

Our model architecture is implemented in pytorch, and is based on a temporal UNet, consistent with Carvalho et al. (2023a) and Janner et al. (2022a), but we lift the number of inputs to 21 instead of 14. (14-dim angle representation, with 7 joint velocities). Exact network architectural details can be found in Appendix C of Janner et al. (2022a). We train the network with a learning rate $\alpha = 1e^{-4}$, with a batch size of 32, and total training steps to be 500,000, using the `Adanm` optimizer.

5.7 NOTATION

It is important to distinguish the group $\mathrm{SO}(2) = \{R_\theta \in \mathbb{R}^{2\times 2} | R_\theta^T R_\theta = I, \det(R_\theta) = 1, \theta \in [-\pi, \pi)\}$ of 2d rotations from its parametrization $\theta \in [-\pi, \pi) \subset \mathbb{R}$ and from the space $S^1 = \{(\cos\theta, \sin\theta) | \theta \in [-\pi, \pi)\} \subset \mathbb{R}^2$ that is the embedding of unit complex numbers in $\mathbb{R}^2$ and which inherits the complex multiplication that makes it a group. In particular, $S^1 \simeq SO(2)$ as topological groups by $(x, y) \mapsto \begin{pmatrix} x & -y \\ y & x \end{pmatrix}$, $R \mapsto (R_{11}, R_{21})$. $\mathrm{SO}(2)$ is a Lie group with Lie algebra $\mathfrak{so}(2) := \left\{ \begin{pmatrix} 0 & -\theta \\ \theta & 0 \end{pmatrix}, \theta \in \mathbb{R} \right\} \simeq \mathbb{R}$. To simplify the notation we take directly the real values as elements of the lie algebra unless otherwise defined. The group and algebra relate via the exponential map

[1] https://drive.google.com/drive/folders/1G-f2X0aSm14Q2knOqF3CGxvnb6AdUNQX?usp=sharing

$\exp : \mathfrak{so}(2) \to \mathrm{SO}(2)$, which we overload to take real values $\exp(\theta) = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix}$. The exponential map is surjective. It is also injective if we limit the domain to $[-\pi, \pi)$ and the inverse is $log : \mathrm{SO}(2) \to \mathfrak{so}(2)$ with $log(R) = atan2(R_{21}, R_{11})$ with $atan2(y, x) = 2arctan\frac{y}{\sqrt{x^2+y^2}+x} \in (-\pi, \pi)$ for $(x, y) \in S^1 - \{(-1, 0), (1, 0), (0, -1)\}$ and $(\pm 1, 0) \mapsto \pm\pi/2$, $(0, -1) \mapsto -\pi$. We will find it useful to use the aforementioned isomorphisms to define: $Exp : \mathbb{R} \to S^1$, $Exp(\theta) = (\cos\theta, \sin\theta)$ which restricted to $[-\pi, \pi)$ has inverse $Log : S^1 \to \mathbb{R}$, $Log(x, y) = atan2(y, x)$. Also, we define the operator $\mathrm{mod}2\pi$ to map real values to the interval $[-\pi, \pi)$ as $\theta mod 2\pi = \theta - 2\pi\lfloor\frac{\theta+\pi}{2\pi}\rfloor$. We distinguish 1) the planning problem time $t \in T$, which we use as subscript and 2) the diffusion process time $i \in N$ which we use as superscript i.e. $\tau^i = (s_t^i)_{t\in T}$ is the i-times diffused trajectory from $\tau = \tau^0$; the denoised trajectory. We denote by $\tau_q, \tau_{\dot{q}}$ the configuration and velocity parts of the state space trajectory.