GPD: Guided Polynomial Diffusion for Motion Planning

Ajit Srikanth*[1], Parth Mahajan*[1], Kallol Saha*[1,2], Vishal Mandadi*[1], Pranjal Paul[1], Pawan Wadhwani[1]
Brojeshwar Bhowmick[3], Arun Singh[4], and Madhava Krishna[1]

***Abstract*— Diffusion-based motion planners are becoming popular due to their well-established performance improvements, stemming from sample diversity and the ease of incorporating new constraints directly during inference. However, a primary limitation of the diffusion process is the requirement for a substantial number of denoising steps, especially when the denoising process is coupled with gradient-based guidance. In this paper, we introduce, for the first time, diffusion in the parametric space of trajectories, where the parameters are represented as Bernstein coefficients. We show that this representation greatly improves the effectiveness of the cost-function guidance and the inference speed. We also introduce a novel stitching algorithm that leverages the diversity in diffusion-generated trajectories to produce collision-free trajectories with just a single cost function-guided model. We demonstrate that our approaches outperform current SOTA diffusion-based motion planners for manipulators and provide an ablation study on key components.**

I. INTRODUCTION

Diffusion-based generative learning frameworks are quickly finding their ground in robotic settings. They have been explored across various domains including task planning [1]–[5], robot learning [6], [7], pose estimation [8], [9], grasping [10], [11], and so on. More recently, they have been adapted for motion planning and collision avoidance in manipulators [12]–[14], serving as a bridge between classical and learning-based motion planners.

Classical motion planning approaches [15]–[27] possess the capability to generalize across various scene types and are prevalent in industrial applications. However, optimization-based approaches rely heavily on the quality of initialization and are prone to getting trapped in local minima. On the contrary, sampling-based planners, which are known to be probabilistically complete, still require extensive hyperparameter tuning to improve their performance under a given time limit and for a given scene type [28]. Furthermore, the trajectories generated are not smooth and therefore require additional optimization. Recently, there has been an exploration of learning-based methods as well. In these approaches, a policy is trained using a dataset of scenes and corresponding optimal solutions. This policy is either employed to assist a classical planner [29] or directly predict the optimal trajectory end-to-end [30]. Such methods exhibit speed and high success rates within their training domain, without the need for extensive hyperparameter tuning. However, they fail to generalize to the out-of-distribution domains.

* Denotes equal contribution
[1] Robotic Research Center, IIIT Hyderabad, [2] Carnegie Mellon University, [3] TCS Research, [4] University of Tartu
Project website: https://guided-polynomial-diffusion.github.io

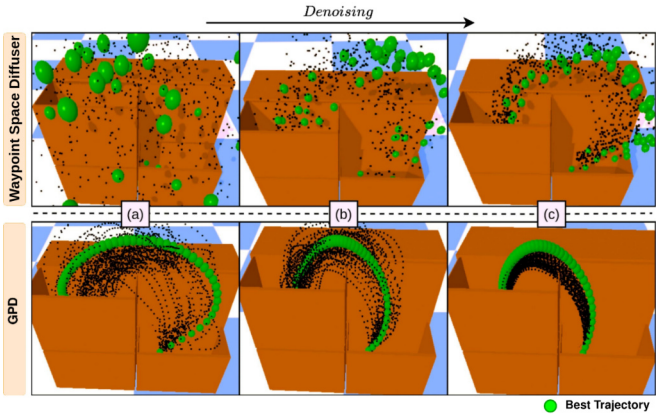

Fig. 1: **G**uided **P**olynomial **D**iffusion (GPD) denoises in the Bernstein space and produces smooth trajectories that converge rapidly to the prior (2nd row). In contrast, waypoint-diffusion models take a much longer time to produce trajectories that resemble the prior, as shown in the first row. (a) initial diffusion steps where GPD maintains smoothness while waypoint diffusers don't; (b) shows intermediate steps of diffusion where GPD already resembles the prior; (c) denotes the final diffusion steps.

Diffusion-based planners [12]–[14] were introduced to produce success rates comparable to learning-based approaches while maintaining the generalizability of classical approaches. [12] introduced cost function guided diffusion for motion planning. Building further, [13] showed that a single cost function is not sufficient to generalize across various settings. Thus, they introduce a framework with an ensemble of cost functions to improve performance. However, this requires a significant number of denoising steps, slowing down the inference speed. [31] adapts diffusion for motion planning in quadrupeds, while [32] expands the capabilities of diffusion-based planners by introducing gradient-free optimization techniques for black box constraints, such as language commands. [14] proposes to train diffusion models to capture and learn each motion planning constraint separately, and compose them directly during inference. While it is shown to be successful in simple environments, it is yet to be scaled up to complex scenes, where it necessitates the composition of a large number of such models during inference.

One of the fundamental bottlenecks of diffusion-based motion planning is the computation time required for the denoising process. [33], [34] attempt to increase the speed of the denoising process by skipping denoising steps and modifying the denoising process to directly estimate the target distributions. However, when denoising steps are coupled with guidance-based correction during inference, it may not

Manuscript 2910 submitted to 2025 IEEE International Conference on Robotics and Automation (ICRA). Received September 16, 2024.

be possible to arbitrarily reduce the denoising steps without compromising solution quality.

In this paper, we argue that one of the core reasons for the slow denoising process is the choice of waypoint parameterization of the trajectory used in existing works [12], [13]. Such a representation produces non-smooth trajectories for most of the denoising process. Moreover, the vanilla gradient-based guidance updates are more likely to result in non-smooth updates with waypoint parameterization. Thus, as a potential workaround, we propose Guided Polynomial Diffusion (GPD), representing trajectories as Bernstein polynomials and designing a diffusion model in polynomial coefficient space along with a gradient update rule for cost-function guidance. As a result, the denoising process is also constrained to lie in the space of continuous and differentiable polynomials, producing highly smooth trajectories within a handful of denoising steps (see Fig.1). We also show that such representation enhances the gradient-based guidance by ensuring smooth updates. Furthermore, the representation of a long horizon trajectory with just a low dimensional vector of polynomial coefficients allows us to fit smaller diffusion models, which further improves the inference time.

In addition, we propose a novel stitching-based algorithm that leverages the diversity in diffusion-generated trajectories by assembling optimal segments from different trajectories directly during inference. Through empirical assessment, we demonstrate that this approach achieves superior success rates using just a single-cost function-guided diffusion model, outperforming prior methods that depend on multiple cost functions. In summary, our key contributions are:

1) We propose GPD, Guided Polynomial Diffusion for motion planning which learns a prior over trajectories parameterized as Bernstein polynomials. We show that cost function guidance in this space yields significantly higher success rates with fewer diffusion steps compared to prior approaches, as shown in Table I, II.
2) The Bernstein space requires a smaller diffusion model and a lower number of denoising steps to generate trajectories that resemble the prior (training dataset), leading to superior inference speeds Table I, II.
3) Additionally, we introduce a stitching algorithm, that leverages the diffusion model's ability to produce diverse trajectories. Employing just a single cost function guided diffusion model, this mechanism outperforms SOTA diffusion planners as shown in Table II.
4) In addition to robotic manipulation, we show that the enhanced inference speeds enable application in reactive scenarios like autonomous driving and mobile robot navigation, as shown in Sec IV-C

II. PROBLEM FORMULATION

Consider the problem of motion planning for a robotic manipulator with m joints placed in an environment E. We represent a trajectory $\boldsymbol{\tau}$ as a sequence of joint configurations $\boldsymbol{s}_i \in \mathbb{R}^m$ over a horizon h. Given an initial configuration $\boldsymbol{s}_0$ and a final configuration $\boldsymbol{s}_{h-1}$ for the trajectory $\boldsymbol{\tau} = [\boldsymbol{s}_0, \boldsymbol{s}_1, \cdots, \boldsymbol{s}_{h-1}] \in \mathbb{R}^{m \times h}$, we intend to minimize the cost function J:

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

The cost is a function of both the trajectory $\boldsymbol{\tau}$ and the environment E. It takes into account the kinematic feasibility of the trajectory, self-collisions within the manipulator, and the collisions between the manipulator and the scene.

III. DIFFUSING OVER POLYNOMIALS

We parameterize our trajectory $\boldsymbol{\tau}$ in terms of a Bernstein polynomial on the variable x. This constrains the trajectory $\boldsymbol{\tau}$ to be a linear combination of $(c+1)$ basis polynomials. The representation is formalized as:

$$\boldsymbol{\tau}(x) = \sum_{j=0}^{c} \boldsymbol{b}_j \binom{c}{j} x^j (1-x)^{c-j} \tag{2}$$

Each Bernstein coefficient denoted as $\boldsymbol{b}_j \in \mathbb{R}^m$ is a control point for the trajectory $\boldsymbol{\tau}$ and exists in the manipulator joint space. The variable $\tau(x)$ denotes a waypoint in the trajectory, where $x \in [0, 1]$. For the j^{th} waypoint in the horizon H, the value of x would be $\left(\frac{j}{H-1}\right)$. We compress a set of Bernstein coefficients into a vector representation denoted by $\boldsymbol{\alpha} = [\boldsymbol{b}_0, \boldsymbol{b}_1, \cdots, \boldsymbol{b}_c] \in \mathbb{R}^{m \times (c+1)}$.

For brevity, we use a Bernstein transform $\boldsymbol{B} \in \mathbb{R}^{(c+1) \times H}$ to convert a set of coefficients $\boldsymbol{\alpha}$ to a trajectory $\boldsymbol{\tau}$. Given the horizon H, we compress the operation shown in Equation 2 to the following matrix dot product:

$$\boldsymbol{\tau} = \boldsymbol{\alpha} \cdot \boldsymbol{B} \tag{3}$$

The transform $\boldsymbol{B}$ compresses the large state-space of a trajectory $\boldsymbol{\tau}$ to a smaller state space of control points $\boldsymbol{\alpha}$. Any set of control points within the robot's joint limits represents a smooth trajectory. Therefore, the transform maintains the temporal consistency and smoothness associated with a trajectory. Bernstein parameterization establishes limits on the trajectory by confining it within the convex hull defined by the Bernstein control points.

A. Diffusion Models

Diffusion models [35], [36] fall within a group of generative models where Gaussian noise is sequentially added to a dataset sample through a scheduled forward diffusion process. This transforms the sample into an isotropic Gaussian. We denote the forward process on the control points $\boldsymbol{\alpha}$ as $q(\boldsymbol{\alpha}_t|\boldsymbol{\alpha}_{t-1}) = \mathcal{N}(\boldsymbol{\alpha}_t; \sqrt{1-\boldsymbol{\beta}_t}\boldsymbol{\alpha}_{t-1}, \boldsymbol{\beta}_t \mathrm{I})$ where $\boldsymbol{\beta}_t$ is the variance schedule and t is the diffusion timestep. A trainable reverse diffusion process $p_\theta(\boldsymbol{\alpha}_{t-1}/\boldsymbol{\alpha}_t) = \mathcal{N}(\boldsymbol{\alpha}_{t-1}; \mu_\theta(\boldsymbol{\alpha}_t, t), \boldsymbol{\Sigma}_t)$ parametrized by θ transforms a sample $\boldsymbol{\alpha}_T \sim \mathcal{N}(0, \mathrm{I})$ from a standard normal distribution into a valid sample from the prior $p_\theta(\boldsymbol{\alpha})$, by sequentially removing Gaussian noise.

We train a diffusion model to learn the prior $p_\theta(\boldsymbol{\alpha})$ over a set of Bernstein polynomial coefficients $\boldsymbol{\alpha}$. During inference, we repeatedly fix the first and the last control point at every denoising step, to condition the diffusion model to produce a trajectory for the given start and goal control points. These start and goal control points directly correspond to the start and goal configurations in the joint space.

Manuscript 2910 submitted to 2025 IEEE International Conference on Robotics and Automation (ICRA). Received September 16, 2024.

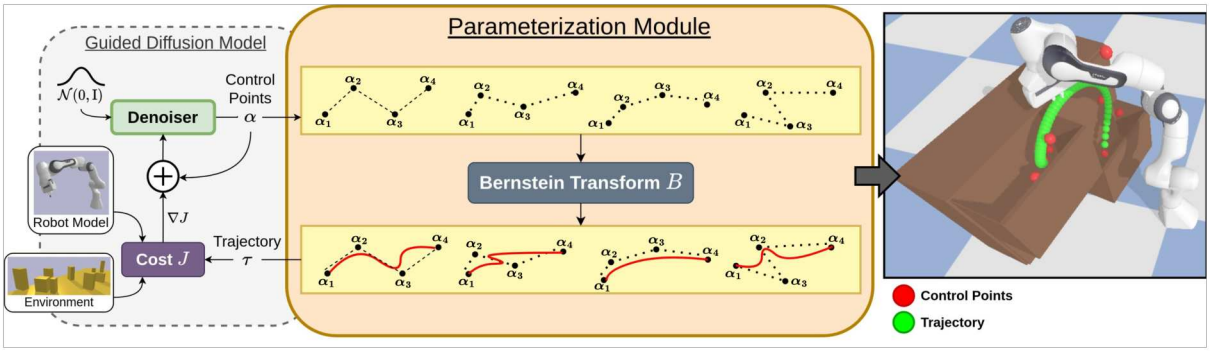

Fig. 2: **Architecture:** GPD uses a parameterization module alongside a guided diffusion model. The diffusion model denoises the control points $\boldsymbol{\alpha}$ sampled from a Gaussian distribution and is guided by a cost function J. The parameterization module uses a Bernstein transform to convert a set of polynomial coefficients or control points to a trajectory in the waypoint space to compute the cost function, which is used to guide the denoising process. The right-most image illustrates the control points and trajectory generated by GPD.

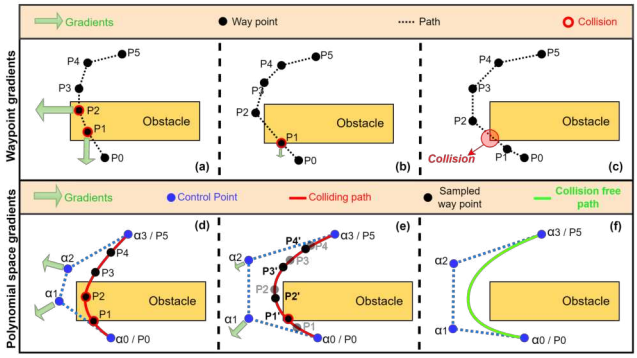

Fig. 3: **Guidance:** GPD exhibits more effective guidance as compared to waypoint space diffusion. Row 1 shows guidance applied directly to the waypoints, where we see that just moving the individual waypoints out of collision might not lead to a final collision-free trajectory. Row 2 shows the enhanced effect of guidance in GPD for the same cost function, where the entire "string" of trajectory is moved out of collision, even when a single waypoint is in collision.

Algorithm 1: Guided Polynomial Diffusion

Input: Learned reverse diffusion $\mu_\theta(\boldsymbol{\alpha}_t, t)$, Covariance schedule $\boldsymbol{\Sigma}_t$, n guide cost functions J, learning rate schedule $\boldsymbol{\lambda}$, hyper-parameter schedule $\mathbf{o}_t$

Initialization: $\boldsymbol{\alpha}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$

for $t = T, \ldots, \ldots 1$ **do**

  `# Compute gradients wrt coefficients for each guide in parallel`

  ${}^{1\ldots n}\boldsymbol{q}_t = \boldsymbol{B} \cdot {}^{1\ldots n}\boldsymbol{\alpha}_t$

  ${}^{1\ldots n}\boldsymbol{G} = \boldsymbol{B}^T \cdot \nabla_{{}^{1\ldots n}\boldsymbol{q}_t}(J({}^{1\ldots n}\boldsymbol{q}_t, {}^{1\ldots n}\boldsymbol{o}_t))$

  `# Denoise for n guides in parallel`

  ${}^{1\ldots n}\boldsymbol{\alpha}_{\boldsymbol{t-1}} \sim \mathcal{N}(\boldsymbol{\mu}_\theta({}^{1\ldots n}\boldsymbol{\alpha}_t, t) + {}^{1\ldots n}\lambda_t \cdot {}^{1\ldots n}\boldsymbol{G}, {}^{1\ldots n}\boldsymbol{\Sigma}_t)$

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

B. Guidance on Coefficients

The denoising process can generate trajectory samples from a distribution that models the dataset but does not account for scene-specific collision costs. To produce trajectories that comply with the scene-specific collision constraints, we guide the diffusion process with the gradients of the cost function J with respect to the control points i.e. Bernstein coefficients $\boldsymbol{\alpha}$, following

$$\boldsymbol{\alpha}_{t-1} = \boldsymbol{\alpha}_t - \gamma\epsilon - \gamma_2 \nabla_{\boldsymbol{\alpha}_t} J(\boldsymbol{\alpha}_t, \boldsymbol{o}_t) + \boldsymbol{\zeta} \tag{4}$$

where $\boldsymbol{o}_t$ are the cost function parameters, and $\boldsymbol{\zeta} \sim N(0, \boldsymbol{\Sigma}_t)$. To compute $\nabla_{\boldsymbol{\alpha}_t} J(\boldsymbol{\alpha}_t, \boldsymbol{o}_t)$, we first discretize the polynomial using the equation $\boldsymbol{q}_t = \boldsymbol{\alpha}_t \cdot \boldsymbol{B}$, where the sampled waypoints of the $\boldsymbol{q}_t$ are equidistant from each other. We then compute $J(\boldsymbol{q}_t, \boldsymbol{o}_t)$ using the differentiable collision cost functions, similar to [13]. Finally, using chain rule of differentiation, we compute $\nabla_{\boldsymbol{\alpha}_t} J(\boldsymbol{\alpha}_t, \boldsymbol{o}_t)$ as:

$$\begin{aligned} \nabla_{\boldsymbol{\alpha}_t} J(\boldsymbol{q}_t, \boldsymbol{o}_t) &= \nabla_{\boldsymbol{\alpha}_t}(\boldsymbol{q}_t) \nabla_{\boldsymbol{q}_t}(J(\boldsymbol{q}_t, \boldsymbol{o}_t)) \\ &= \boldsymbol{B}^T \cdot \nabla_{\boldsymbol{q}_t}(J(\boldsymbol{q}_t, \boldsymbol{o}_t)) \end{aligned} \tag{5}$$

The R.H.S of (5) can also be understood as pre-conditioning the gradient obtained by waypoint parameterization with the matrix $\boldsymbol{B}^T$. This has connections to the idea presented in [15], which uses a finite-differences matrix to pre-condition cost gradients obtained for waypoint parameterized trajectories. However, $\boldsymbol{B}^T$ is a better pre-conditioner since it ensures higher continuity and differentiability in the updates. This is similar to pulling a string, where gradients from each waypoint influence all coefficients, shifting the entire trajectory while maintaining smoothness. This contrasts with updating the waypoints directly with their corresponding gradients, where each waypoint moves independently of each other (Fig. 3). We believe this is the first use of such preconditioning in diffusion model guidance updates.

Furthermore, we argue that cost function guidance in GPD is more effective as it applies gradients to smoother, prior-resembling trajectories earlier in denoising, as seen in Fig. 1. This contrasts with prior guided diffusion methods, which apply gradients to non-smooth waypoints in most of the intermediate denosing steps. Such methods are less effective as guidance on noisy trajectories is often dominated by the denoising process pushing the trajectories towards the prior,

Manuscript 2910 submitted to 2025 IEEE International Conference on Robotics and Automation (ICRA). Received September 16, 2024.

Algorithm 2: GPD with Stitching

Input: Trajectories generated by GPD $\boldsymbol{T}_{gpd}$, Goal G, Cost function J, Collision window size w

Initialization: $i = arg\,min \;\; J(\boldsymbol{T}_{gpd})$, $\boldsymbol{\tau}_{sol} = [\,]$, $j = 0$

while $\tau_i^j \notin G$ **do**
  $\boldsymbol{C}_{free} = collision\_fn(\boldsymbol{\tau}_i^{j:j+w}, E)$
  if $!\,C_{free}$ **then**
    $i_{stitched}$, $j_{stitched}$, $\boldsymbol{\tau}_{stitch}$ = **get_valid_stitch**($\boldsymbol{T}_{gpd}$, τ_i^j)
    $\boldsymbol{\tau}_{sol}.add(\boldsymbol{\tau}_{stitch})$
    i, j = $i_{stitched}$, $j_{stitched}$
  $\boldsymbol{\tau}_{sol}.add(\boldsymbol{\tau}_i^j)$
Return $\boldsymbol{\tau}_{sol}$

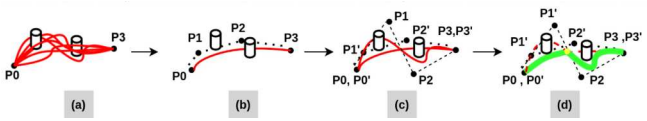

Fig. 4: **Stitching:** (a) Diverse batch of generated trajectories. (b) trajectory with the lowest cost. (c) pair of trajectories that can be stitched. (d) shows the final collision-free trajectory by stitching two different trajectories. The stitch is shown in yellow.

rather than towards optimising the cost.

C. Stitching

Prior work [13] shows the need for an ensemble of cost-guided models, however, this would require fine hyper-parameter tuning of multiple cost functions which can be a tedious process. We circumvent this problem by leveraging the diversity of generated trajectories. Specifically, we observe that various trajectories traverse through distinct collision-free segments as shown in Fig. 4, which when stitched together can produce fully collision-free trajectories.

Let E represent the scene as a collection of obstacles. Let $\boldsymbol{T}_{gpd} = [\boldsymbol{\tau}_1, \boldsymbol{\tau}_2, ..., \boldsymbol{\tau}_n]$ be a batch of n trajectories generated by GPD, and τ_i^j be the j^{th} waypoint in trajectory $\boldsymbol{\tau}_i$. Let $E_i^{free} \subseteq E$ denote segments in the scene that do not collide with the robot in trajectory $\boldsymbol{\tau}_i$. We suggest that when a set of trajectories $\boldsymbol{T}$ is highly diverse, $\cup_i E_i^{free} = E$, even in cases where $E_i^{free} \neq E \;\forall\; i \in \{1, 2, ..., n\}$. This implies that as we increase the diversity, we are more likely to find a set of collision-free segments that can be stitched together to obtain a fully collision-free trajectory. To leverage this property, we propose a stitching framework, that merges trajectories during inference.

Initially, we rollout the lowest-cost trajectory $\boldsymbol{\tau}_i$ from $\boldsymbol{T}_{gpd}$ using a sliding window approach. We denote e_i^{free} as the segments in the window that do not collide with the robot. During the rollout, we sequentially evaluate if $e_i^{free} \neq E$, in which case we look for a valid stitch target $\boldsymbol{\tau}_s$ in $\boldsymbol{T}_{gpd}$. We denote a stitch target τ_s as valid, if the closest waypoint τ_s^p in τ_s (in the direction of goal) is collision-free in its respective sliding window. Given a valid stitch target, we use RRT-C [37] as a local planner to obtain a stitch τ_{stitch}. The process is continued until the goal configuration is reached. Algorithm 2 summarizes our stitching framework.

Method	Per Dataset Success (%, ↑) Global	Hybrid	Both	Planning Time (s, ↓)
PD	32.11	40.38	41.54	**0.22**
MPD [12]	51.33	62.27	67.12	7.32
EDMP-1G [13]	53.11	66.50	68.88	7.95
GPD-1G	**65.50**	**72.89**	**73.00**	0.8

TABLE I: Comparison of GPD-1G against PD and guidance with a single cost function in the waypoint space (EDMP-1G, MPD), in terms of success rates and inference time.

In polynomial diffusion, the intermediate diffusion steps yield smooth trajectories that converge rapidly to our kinematically valid prior. By the final diffusion steps, the trajectories are both smooth and kinematically valid, closely resembling the prior (as shown in Fig. 1) while also incorporating some applied guidance. Hence including the trajectories obtained from the last few diffusion steps results in a batch of more diverse trajectories, due to the combination of trajectories at various stages of guidance and denoising. Our results demonstrate that even with a single guide, we can achieve higher success rates. To test the diversity of our prior, we benchmark the performance of our stitching algorithm in Section IV-B.

IV. EXPERIMENTAL SETUP

We conduct our experiments using the Franka Panda robot in the Pybullet simulator [38].

Training: We train our model on a dataset of 6.54 million Bernstein polynomials. The polynomials are generated by fitting their coefficients using least squares to the MπNets [30] training dataset, comprising of 6.54 million trajectories .We use 8th-order Bernstein polynomials with 8 control points to parameterize the train set trajectories, as they provide a sufficient fit. The MπNets dataset consists of tabletops, dressers, and cupboards out of which 3.27 million trajectories generated using a global planner, AIT* [39], and another 3.27 million trajectories using a hybrid planner, that uses AIT* [39] to generate plans in the end-effector space, and Geometric Fabrics [40] to produce geometrically consistent motion in joint space. We model our polynomial denoiser as a temporal UNet similar to [6], and train it for 20k epochs on a RTX A4000 GPU. We set the number of diffusion timesteps $T = 64$, in comparison to EDMP [13] which uses $T = 256$.

Testing: We benchmark our performance on 3 different test datasets from MπNets [30]. (1) A *Global Solvable Dataset* consists of 1800 environments on which the *global planner* was able to generate valid collision-free trajectories. (2) A *Hybrid Solvable Dataset* consists of 1800 environments on which only the *hybrid planner* was able to generate valid collision-free trajectories. (3) A *Both Solvable Dataset* consists of 1800 environments where both planners (1 & 2) were able to generate valid collision-free trajectories. Each scene is defined by an obstacle configuration, an initial joint configuration and a goal pose. We perform our benchmarks

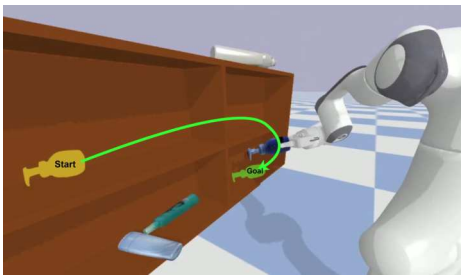

Fig. 5: **Out-of-distribution scenes:** GPD generalises to unseen object-in-hand scene **without any additional training** just by modifying our cost function directly at inference

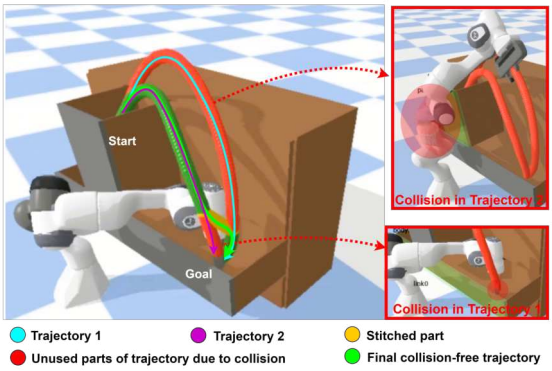

Fig. 6: **Qualitative Results of Stitching.** Collision is detected in the red portion of the initial selected trajectory. The stitching algorithm then generates a stitch (shown in yellow) to another trajectory (which is in collision at a different segment). The final executed trajectory is shown in green.

on a workstation with a Ryzen 9 5950X processor and an RTX 3060 GPU.

Metrics: We use Success Rate (SR) as the main metric to compare our performance against baselines. The success rate is evaluated to be the percentage(%) of successful runs in the test dataset. If the robot end-effector successfully reaches the goal pose, without colliding with the environment or itself, we report a score of 1, otherwise 0. Collisions are checked using Pybullet's in-built collision checker, by executing the planned trajectory in the bullet physics engine. We also benchmark the planning time of our algorithms, and stitching time for algorithms that use the stitching method.

Models and Baselines: We benchmark the performances of six different models that use polynomial parameterization. (1) GPD-1G - GPD with a single cost function-based guidance, (2) GPD-7G - GPD with 7 cost functions with varying configurations similar to [13] (3) PD - Polynomial diffusion prior, (4) GS - Stitching applied to Bernstein polynomials sampled from a Gaussian distribution, (5) PDS - Stitching applied to trajectories generated by PD, (6) GPDS - Stitching applied to trajectories generated by GPD-1G. We compare our model against stochastic optimization-based planners (G.Fabrics [40], STORM [16]), optimization-based planner CHOMP [15] with a batch of quintic-spline initializations, and behavior-cloning based deep-learning planners (MPNets [29], MπNets [30]) trained on the same dataset.The scores for G. Fabrics, STORM and M*pi*nets, have been reported as per the ones mentioned in the M*pi*nets paper. We also compare against guided-diffusion based models MPD [12] and EDMP [13]. We adapted MPD to the more diverse MπNets environment and for a fair comparison MPD is trained with different number diffusion steps and UNet sizes including the ones mentioned in their paper, and the best scores have been reported. Similarly, EDMP scores are reported with the best 12 Guides provided by them.

A. Effect of Parameterization on Guidance

We analyse the effect of parameterizing trajectories with Bernstein polynomial coefficients on Guidance by benchmarking the GPD-1G against PD, and also with SOTA waypoint diffusion models EDMP [13] and MPD [12]. Table I shows that while unguided polynomial prior (PD) is able to achieve good success rates, scene-specific cost guided frameworks (MPD, EDMP, GPD) significantly improve the success rates. We further find that our GPD-1G clearly outperformes both EDMP-1G and MPD in terms of both success rate, and speed. This increase in success rate is attributed to the increased effect of guidance as mentioned in Section III-B. Furthermore, GPD is about 10 times faster than MPD and EDMP-1G, taking only 0.8s to produce collision-free trajectories. This is due to the polynomial parameterization, which enhances convergence speed, reduces the size of the state space, and decreases the complexity of the diffusion model. Note that MPD used on these scenes has a higher number of diffusion steps as compared to the one mentioned in [12]; this is because of the increase in diversity and the complexity of the prior trajectories in the MπNets scenes.

***Comparision with baselines**:* We first evaluate our GPD-7G model, with 3 cost functions based on the intersection volume and 4 cost functions based on the swept volume costs similar to [13] on the test datasets. We show SOTA peformance when compared to prior diffusion based methods [12], [13] and classical methods [15], [16], [40], while using lower number of cost functions than the previous SOTA [13]. We also compare with the learning based approaches MπNets and MPNets, which are expected to show high success rates as they are trained specifically on these datasets. Our method achieves SOTA performance in the global dataset and demonstrates comparative performance to the SOTA planner (MπNets) in Hybrid and Both datasets, despite being trained in a scene-agnostic manner. Additionally, similar to [12], [13], we generalize to out-of-distribution settings, such as Object-in-hand (Fig. 5) by incorporating the object as another link of our manipulator with fixed joints and adding an object-environment collision-cost to our cost function directly during inference. Thus, we generalize to new settings without needing any additional training, unlike MπNets and MPNets.

B. Leveraging diversity to increase performance

Using an ensemble of cost functions greatly improves the success rates as shown in [13]. However, each cost function needs to be carefully tuned, which makes it a tedious process. We show that our stitching algorithm, applied on a single

Method	Per Dataset Success (%, ↑)			Time (s, ↓)
	Global	Hybrid	Both	
G. Fabrics [40]	38.44	59.33	60.06	0.15
STORM [16]	50.22	74.50	76.00	4.03
CHOMP [15]	26.67	31.61	32.2	2.39
MPNets [29]	41.33	65.28	67.67	4.95
MπNets [30]	75.78	**95.33**	**95.06**	0.33
MPD [12]	46.33	57.27	62.12	7.32
EDMP [13]	75.93	86.13	85.06	8.22
GPD 7G	81.94	90.88	90.24	0.85
GS	43.31	57.92	54.37	7.48 (0 D + 7.48 S)
PDS	68.81	80.19	75.49	4.17 (0.22 D + 3.95 S)
GPDS	**87**	92.83	92.61	1.9 (0.8 D + 1.1 S)

TABLE II: Comparison of our models in terms of success rate, and planning time (in seconds) on the MπNets [30] dataset. D refers to denoising time, and S is the time taken by the stitching algorithm

cost function guided polynomial diffusion (GPDS), outperforms both single and multi-cost function-guided diffusion baselines [12], [13] (as shown in Table II). The stitching algorithm leverages the diversity of the generated trajectories as explained in Section III-C and illustrated in Fig. 6. We use RRT-Connect [21] as the local planner to stitch between the generated trajectories.

We show all the results related to this section on Table II where we compare stitching on GPDS, GS, and PDS. We use a batch size of 1000 trajectories for stitching on GS. For PDS and GPDS we use a batch size of 32 during the denoising process and include trajectories from the last 5 diffusion steps. This increases the effective batch size for stitching by a factor of 5 by directly leveraging the fact that the trajectories generated by polynomial diffusion in the last few timesteps closely resemble the prior distribution, as illustrated in Fig. 1 and explained in Sec III-C.

The table shows that GPDS, which utilizes a single cost function, outperforms GPD 7G and EDMP, both of which use an ensemble of cost functions to boost their overall success rates. Next, we show that stitching requires good sample quality of generated trajectories. For this, in the same table, we first compare PDS against GS, highlighting the point that stitching on kinematically valid trajectories (PDS) far outperforms stitching on Gaussian-sampled trajectories (GS). In addition, since GPD ensures that most segments of the trajectories are collision-free through guidance, we see that stitching on such trajectories (GPDS) outperforms stitching on just kinematically feasible prior (PDS).

We also benchmark GPDS against an optimized version of RRT-Connect, to show that GPDS's superior performance is a result of combining the local planner used for stitching (RRT-Connect) and the GPD prior. RRT-connect achieves a success rate of 83.33, 90.83, and 90.60 on the global, hybrid, and both datasets respectively. As evident from the Section II, GPDS outperforms both RRT-Connect and GPD 1G. Note that RRT-Connect takes an average time of 7.14s when benchmarked on the MπNets dataset, whereas, we only take an average of 1.9s.

C. GPD for Navigation

The non-limiting nature of our model is also demonstrated for time-critical requirements of navigation tasks across distinct environments. Fig. 7 and 8 showcase the model's ability to generate multi-modal trajectories in complex indoor and outdoor conditions respectively, where the denoising process is guided by a cost function that penalizes curvature and acceleration magnitudes. In the urban driving scenario, shown in Fig. 8, the presence of dynamic obstacles necessitates the inclusion of collision cost to plan collision-free trajectories at reactive rates. In both contexts, GPD was able to generate smooth, feasible trajectories at a frequency of 16 Hz, demonstrating near real-time planning capabilities due to the model's enhanced computational speed.

Fig. 7: **Multi-modal trajectory generation in an indoor space.** The minimum-cost trajectory is shown in Red, along with diverse navigable paths projected onto a perspective view.

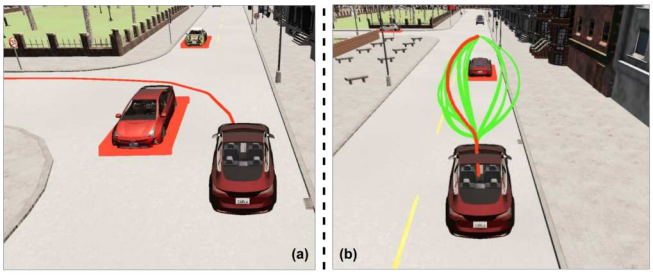

Fig. 8: **GPD performance in a simulated urban driving scene.** (a) shows a collision-free trajectory for a turning scenario. (b) shows a diverse trajectory set generated by GPD with the selected plausible path highlighted in Red. The relevant obstacles considered for collision-checking are shown with a red patch beneath them.

V. CONCLUSION & FUTURE WORK

In this paper, we enhanced previous diffusion-based motion planners by learning a prior over Bernstein polynomial coefficients to represent valid trajectories, in contrast to waypoint representation. We show that cost function guidance during inference is more effective when denoising in the polynomial space rather than the waypoint space. We also introduce a stitching algorithm to leverage the diversity of generated trajectories directly during inference. Through these, we show SOTA performance at both speed and success rates. Finally, with the enhanced inference speed, we demonstrate our ability to adapt to reactive scenes such as in an autonomous driving setting.

Future research could explore implicit modelling of open-vocabulary cost functions from data, potentially enabling language-directed motion planning without explicit cost functions.

REFERENCES

[1] C.-F. Yang, H. Xu, T.-L. Wu, X. Gao, K.-W. Chang, and F. Gao, *Planning as in-painting: A diffusion-based embodied task planning framework for environments under uncertainty*, 2023. arXiv: 2312.01097 [cs.CV].

[2] Z. Yang, J. Mao, Y. Du, *et al.*, "Compositional Diffusion-Based Continuous Constraint Solvers," in *Conference on Robot Learning*, 2023.

[3] J. Chang, H. Ryu, J. Kim, *et al.*, "Denoising heat-inspired diffusion with insulators for collision free motion planning," *arXiv preprint arXiv:2310.12609*, 2023.

[4] X. Fang, C. Garrett, C. Eppner, T. Lozano-Pérez, L. Kaelbling, and D. Fox, "DiMSam: Diffusion models as samplers for task and motion planning under partial observability," in *CoRL 2023 Workshop on Learning Effective Abstractions for Planning (LEAP)*, 2023. [Online]. Available: https://openreview.net/forum?id=a14qioqpel.

[5] W. Liu, Y. Du, T. Hermans, S. Chernova, and C. Paxton, "Structdiffusion: Language-guided creation of physically-valid structures using unseen objects," in *RSS 2023*, 2023.

[6] M. Janner, Y. Du, J. Tenenbaum, and S. Levine, "Planning with diffusion for flexible behavior synthesis," in *International Conference on Machine Learning*, 2022.

[7] X. Li, V. Belagali, J. Shang, and M. S. Ryoo, "Crossway diffusion: Improving diffusion-based visuomotor policy via self-supervised learning," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*, 2024, pp. 16 841–16 849. DOI: 10.1109/ICRA57147.2024.10610175.

[8] A. Simeonov, A. Goyal, L. Manuelli, *et al.*, "Shelving, stacking, hanging: Relational pose diffusion for multi-modal rearrangement," *Conference on Robot Learning*, 2023.

[9] S. Suresh, Z. Si, S. Anderson, M. Kaess, and M. Mukadam, "MidasTouch: Monte-Carlo inference over distributions across sliding touch," in *Proc. Conf. on Robot Learning, CoRL*, Auckland, NZ, Dec. 2022.

[10] J. Urain, N. Funk, J. Peters, and G. Chalvatzaki, "Se (3)-diffusionfields: Learning smooth cost functions for joint grasp and motion optimization through diffusion," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2023, pp. 5923–5930.

[11] Y. Xu, W. Wan, J. Zhang, *et al.*, "Unidexgrasp: Universal robotic dexterous grasping via learning diverse proposal generation and goal-conditioned policy," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2023, pp. 4737–4746.

[12] J. Carvalho, A. T. Le, M. Baierl, D. Koert, and J. Peters, "Motion planning diffusion: Learning and planning of robot motions with diffusion models," in *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2023, pp. 1916–1923. DOI: 10.1109/IROS55552.2023.10342382.

[13] K. Saha, V. Mandadi, J. Reddy, *et al.*, "Edmp: Ensemble-of-costs-guided diffusion for motion planning," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*, 2024, pp. 10 351–10 358. DOI: 10.1109/ICRA57147.2024.10610519.

[14] Y. Luo, C. Sun, J. B. Tenenbaum, and Y. Du, "Potential based diffusion motion planning," in *Forty-first International Conference on Machine Learning*, 2024.

[15] 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.

[16] 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.

[17] 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.

[18] 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.

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

[20] 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.

[21] 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.

[22] 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.

[23] 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.

[24] 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.

[25] 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.

[26] 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.

[27] W. Thomason, Z. Kingston, and L. E. Kavraki, "Motions in microseconds via vectorized sampling-based planning," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*, 2024, pp. 8749–8756. DOI: 10.1109/ICRA57147.2024.10611190.

[28] M. Moll, C. Chamzas, Z. Kingston, and L. E. Kavraki, "Hyperplan: A framework for motion planning algorithm selection and parameter optimization," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2021, pp. 2511–2518. DOI: 10.1109/IROS51168.2021.9636651.

[29] 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.

[30] 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.

[31] J. Liu, M. Stamatopoulou, and D. Kanoulas, "Dipper: Diffusion-based 2d path planner applied on legged robots," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*, 2024, pp. 9264–9270. DOI: 10.1109/ICRA57147.2024.10610013.

[32] B. Yang, H. Su, N. Gkanatsios, *et al.*, "Diffusion-es: Gradient-free planning with diffusion for autonomous and instruction-guided driving," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2024, pp. 15 342–15 353.

[33] C. Lu, Y. Zhou, F. Bao, J. Chen, C. Li, and J. Zhu, "Dpm-solver: A fast ode solver for diffusion probabilistic model sampling in around 10 steps," *Advances in Neural Information Processing Systems*, vol. 35, pp. 5775–5787, 2022.

[34] J. Song, C. Meng, and S. Ermon, "Denoising diffusion implicit models," *arXiv preprint arXiv:2010.02502*, 2020.

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

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

[37] J. Kuffner and S. LaValle, "Rrt-connect: An efficient approach to single-query path planning," in *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065)*, vol. 2, 2000, 995–1001 vol.2. DOI: 10.1109/ROBOT.2000.844730.

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

[39] 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.

[40] M. Xie, K. V. Wyk, A. Li, *et al.*, "Geometric fabrics for the acceleration-based design of robotic motion," *ArXiv*, vol. abs/2010.14750, 2020. [Online]. Available: https://api.semanticscholar.org/CorpusID:225094405.