Potential Based Diffusion Motion Planning

Yunhao Luo [1] **Chen Sun** [1] **Joshua B. Tenenbaum** [2] **Yilun Du** [2]

Abstract

Effective motion planning in high dimensional spaces is a long-standing open problem in robotics. One class of traditional motion planning algorithms corresponds to potential-based motion planning. An advantage of potential based motion planning is composability – different motion constraints can easily combined by adding corresponding potentials. However, constructing motion paths from potentials requires solving a global optimization across configuration space potential landscape, which is often prone to local minima. We propose a new approach towards learning potential based motion planning, where we train a neural network to capture and learn an easily optimizable potentials over motion planning trajectories. We illustrate the effectiveness of such approach, significantly outperforming both classical and recent learned motion planning approaches and avoiding issues with local minima. We further illustrate its inherent composability, enabling us to generalize to a multitude of different motion constraints. Project website at https://energy-based-model.github.io/potential-motion-plan.

1 Introduction

Motion planning is a fundamental problem in robotics and aims to find a smooth, collision free path between a start and goal state given a specified configuration space, and is heavily used across a variety of different robotics tasks such as manipulation or navigation (Laumond et al., 1998). A variety of approaches exist for motion planning, ranging from classical sampling based approaches (Kavraki et al., 1996; Kuffner & LaValle, 2000; Karaman & Frazzoli, 2011; Gammell et al., 2015) and optimization based methods (Ratliff et al., 2009; Mukadam et al., 2018; Kalakrishnan et al., 2011). A recent body of works have further explored how learned neural networks can be integrated with motion planning for accelerated performance (Ichter & Pavone, 2019; Qureshi et al., 2019; Fishman et al., 2023; Yamada et al., 2023; Le et al., 2023).

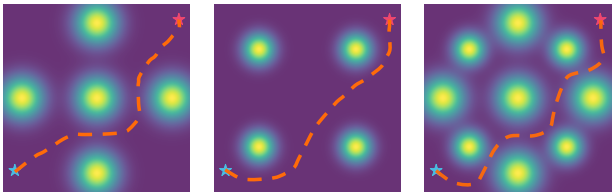


Figure 1: **Illustrative Example of Composing Diffusion Energy Potentials.** Our approach learns different potential functions over motion planning trajectories $q_{1:N}$ (orange dashed lines). Different potentials can be combined and optimized to construct new motion plans that avoid obstacles encoded in both potential functions.

A classical approach towards motion planning is potential based motion planning (Khatib, 1986; Koren et al., 1991; Ratliff et al., 2009; 2018; Xie et al., 2020), where both obstacles and goals define energy potentials through which trajectories are optimized to reach. An advantage of potential based motion planning is that different constraints to motion planning can be converted into equivalent energy potentials and directly combined to optimize for motion plans. However, potential based methods rely on gradient optimization with respect to local geometry, resulting in the long-standing local minima issues (LaValle, 2006). In addition, such motion planning techniques typically require implicit obstacle representations, which is hard to obtain in real-world settings.

We present a potential based motion planning approach leveraging diffusion models (Sohl-Dickstein et al., 2015; Ho et al., 2020) where diffusion models are used to parameterize and learn potential landscapes across configuration space trajectories between start and goal states. These potential functions can be directly inferred from raw perceptual inputs, removing the need for implicit object representations. Furthermore, the annealed optimization procedure across a sequence of potential energy landscapes in diffusion models (Du et al., 2023) can help avoid local minima in optimization. This optimization procedure is further stochastic, enabling the planner to generate a multitude of motion plans with diverse morphology for a specific problem, offering

[1]Brown University [2]MIT. Correspondence to: Yunhao Luo <yluo73@cs.brown.edu>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024.

various motion plan candidates for selection in test time. Finally, as our potential is defined globally over an environment, it is guided by both local and global environment geometry, and thus our method provides more accurate plans that require significantly less collision checking, compared with problem-independent sampling-based planners.

One major hurdle of learning-based motion planners is their generalizability to environments with unseen, more complex constraints. For example, a learned model trained on sparse obstacles will perform poorly in scenarios with cluttered obstacles, as such a setting is out of distribution. Our approach addresses this through *compositionality* – our learned potentials can be additively composed together to jointly solve motion planning problems with larger sets of constraints than seen at training time. As illustrated in Figure 1, combining two potentials from different diffusion models enables us to optimize for trajectories that satisfy both constraints, one to avoid obstacles in a cross, and a second to avoid obstacles in a square. Such flexibility to ad-hoc composition of constraints is especially useful in robotics where agents will often experience new sets of motion constraints in its environment over the course of execution.

Overall, in this paper, our contributions are three-fold. **(1)** We present an approach to learned potential based motion planning using diffusion models. **(2)** We illustrate the effectiveness of our approach, outperforming existing classical and learned motion planning algorithms in accuracy and collision checks. **(3)** We illustrate the compositionality of motion planner, enabling it to generalize to multiple sets of motion constraints as well as an increased number of objects.

2 Related Work

Motion Planning. Classic sampling-based motion planners (Kavraki et al., 1996; Kuffner & LaValle, 2000; Elbanhawi & Simic, 2014; Gammell et al., 2014; Janson et al., 2015; Choudhury et al., 2016; Strub & Gammell, 2020) have gained wide adoption due to their efficiency and generalizability. However, problem-independent nature of these methods can result in inefficiency particularly when planning for similar problems repetitively. Reactive methods, such as potential-based approaches (Khatib, 1986; Ratliff et al., 2018; Xie et al., 2020), velocity obstacles (Fiorini & Shiller, 1998; Van den Berg et al., 2008), and safety barrier certificates (Wang et al., 2017) can provide fast updates and have the guarantee for obstacle avoidance. However, their performance is typically constrained by local minima or numerical instability (LaValle, 2006), and they usually need to construct obstacle representations in the robot configuration space, which is hard to obtain from raw perception. To address these issues, recent works have proposed many deep-learning based motion planners (Ichter & Pavone, 2019; Bency et al., 2019; Fishman et al., 2023). Other works combine neural network with sampling-based methods (Qureshi et al., 2019; Johnson et al., 2021; Yu & Gao, 2021; Lawson & Qureshi, 2022), or combine trajectory level generative models with specifically designed cost functions (Saha et al., 2023; Carvalho et al., 2023) that requires privileged information. These methods can increase planning speed, expand the planning horizon, or reduce the access queries to the environment by leveraging learned knowledge. However, the performance of learned motion-planning approaches on out-of-distribution environments often sharply degenerates. In addition, many existing methods are only constrained to simple 2D environments (Yonetani et al., 2021; Chaplot et al., 2021; Toma et al., 2021; Chang et al., 2023; Carvalho et al., 2022). In contrast, we present a learning-based potential motion planning approach, requiring no ground truth knowledge of the optimized cost objective, which we illustrate can effectively generalize to new environments through composability of potentials.

Diffusion Models for Robotics. Many recent works have explored the application of diffusion model for robotics (Janner et al., 2022; Chen et al., 2022; Kapelyukh et al., 2023; Ha et al., 2023). Current research spans a variety of robotics problems, including action sequence generation (Liang et al., 2023; Fang et al., 2023; Li et al., 2023), policy (Wang et al., 2023; Kang et al., 2023), grasping (Urain et al., 2023; Huang et al., 2023), and visuomotor planning or control (Dalal et al., 2023; Yang et al., 2023a; Chi et al., 2023), with recent work also exploring their application in solving manipulation constraints (Yang et al., 2023b). In contrast to these works, we focus on how diffusion models can be used to explicitly parameterize and learn potentials in potential-based motion planning. We illustrate the efficacy of such an approach across motion-planning settings and its ability to generalize to new environments through composition with other learned potentials.

3 Method

We first introduce potential based motion planning in Section 3.1. We then discuss how potential based motion planning can be implemented with diffusion models in Section 3.2. We further discuss how such an approach enables us to combine multiple different potentials together in Section 3.3. Following this, we discuss how we can refine motion plans generated by diffusion models in cases of collision in Section 3.4. Finally, we show the probabilistic completeness of the proposed method in Section 3.5.

3.1 Potential Based Motion Planning

Given a specified start state q_{st} and end state q_{e} in a configuration space $\mathbb{R}^n$, motion planning is formulated as finding a collision-free trajectory $q_{1:N}$ which starts from q_{st} and

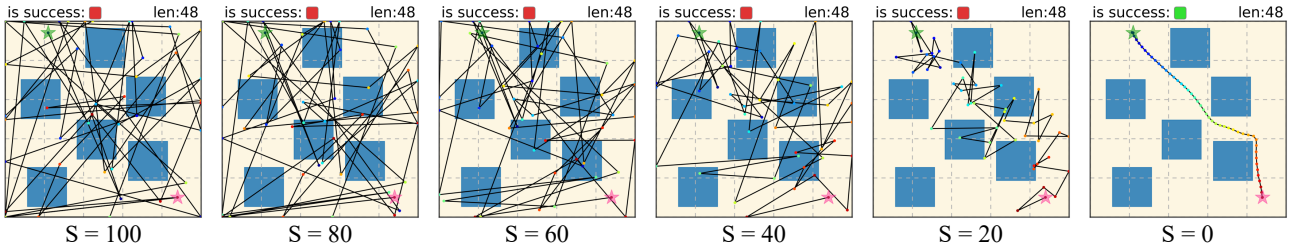


Figure 2: **Trajectory Denoising Process.** The trajectory is randomly initialized from Gaussian in timestep $S = 100$. Noise is iteratively removed via the gradient of the energy function as given in equation 5. A feasible trajectory can be obtained at timestep $S = 0$.

ends at q_e. To solve for such a collision-free trajectory $q_{1:N}$ in potential based motion planning (Khatib, 1986; Koren et al., 1991), a potential function $U(q) : \mathbb{R}^n \to \mathbb{R}$ on the configuration space composed of

$$U(q) = U_{\text{att}}(q) + U_{\text{repel}}(q), \tag{1}$$

is defined, where $U(q)$ assigns low potential value to the goal state q_e and high potential to all states which are in collision. In Equation 1, $U_{\text{att}}(q)$ represents a attraction potential that has low values at the the end state q_{end} and high values away from it, and $U_{\text{repel}}(q)$ represents a repulsion potential that has high values near obstacles and low values away them. The functional form of the potential function $U(q)$ provides an easy approach to integrate additional obstacles in motion planning by adding a new potential $U_{\text{new}}(q)$ representing obstacles to the existing potential in Equation 1.

To obtain a motion plan from a potential function $U(q)$, a collision-free trajectory $q_{1:N}$ from q_{st} to q_e is obtained by iteratively following the scaled gradient of the potential function

$$q_t = q_{t-1} - \gamma \nabla_q U(q), \tag{2}$$

with a successful motion plan constructed when the optimization procedure reaches the minimum of the potential function $U(q)$. A major limitation of above approach in Equation 2 is *local minima* – if the optimization procedure falls in such a minima, the motion plan will no longer successfully construct paths from q_{st} to q_e (Yun & Tan, 1997; Teli & Wani, 2021; LaValle, 2006).

3.2 Potential Based Diffusion Motion Planning

We next discuss how to learn potentials for potential motion planning that enables us to effectively optimize samples. Given a motion plan $q_{1:T}$ from start state q_{st} to end state q_e and a configuration space characterization C (*i.e.* the set of obstacles in the environment), we propose to learn a trajectory-level potential function U_θ so that

$$q^*_{1:T} = \arg\min_{q_{1:T}} U_\theta(q_{1:T}, q_{\text{st}}, q_e, C), \tag{3}$$

where $q^*_{1:T}$ is a successful motion plan from q_{st} to q_e. To learn the potential function in Equation 3, we propose to learn an EBM (LeCun et al., 2006; Du & Mordatch, 2019) across a dataset of solved motion planning $D = \{q^i_{\text{st}}, q^i_e, q^i_{1:T}, C^i\}$, where $e^{-E_\theta(q_{1:T}|q_{\text{st}}, q_e, C)} \propto p(q_{1:T}|q_{\text{st}}, q_e, C)$. Since the dataset D is of solved motion planning problems, the learned energy function E_θ will have minimal energy at successful motion plans $q^*_{1:T}$ and thus satisfy our potential function U_θ in Equation 3.

To learn the EBM landscape that enables us to effectively optimize and generate motion plans $q^*_{1:T}$, we propose to shape the energy landscape using denoising diffusion training objective (Sohl-Dickstein et al., 2015; Ho et al., 2020). In this objective, we explicitly train the energy landscape so gradient with respect to the energy function can denoise and recover a motion plan $q_{1:T}$ across many differing levels of noise corruption $\{1, \dots, S\}$ ranging from mostly correct motion paths to fully corrupted Gaussian noise trajectories. By shaping the gradient of the energy function to generate motion plans $q_{1:T}$ from arbitrary initialized trajectories, our learned energy landscape is able to effectively optimize for motion paths.

Formally, to train our potential, we use the energy based diffusion training objective in (Du et al., 2023), where the gradient of energy function is trained to denoise noise-corrupted motion plans $q^i_{1:T}$ from D

$$\mathcal{L}_{\text{MSE}} = \|\epsilon - \nabla_{q_{1:T}} E_\theta(\sqrt{1-\beta_s} q^i_{1:T} + \sqrt{\beta_s}\epsilon, s, q^i_{\text{st}}, q^i_e, C^i)\|^2 \tag{4}$$

where ϵ is sampled from Gaussian noise $\mathcal{N}(0, 1)$, $s \in \{1, 2, ..., S\}$ is the denoising diffusion step (we set $S = 100$), and β_s is the corresponding Gaussian noise corruption on a motion planning path $q^i_{1:T}$. We refer to E_θ as the *diffusion potential function*.

To optimize and sample from our diffusion potential function, we initialize a motion path $q^S_{1:T}$ at diffusion step S from Gaussian noise $\mathcal{N}(0, 1)$ and optimize for motion path following the gradient of the energy function E_θ. We iteratively refine the motion path $q^s_{1:T}$ across each diffusion step following

$$\begin{aligned} q^{s-1}_{1:T} &= q^s_{1:T} - \gamma\epsilon + \xi, \quad \xi \sim \mathcal{N}\big(0, \sigma_s^2 I\big), \\ \epsilon &= \nabla_{q_{1:T}} E_\theta(q_{1:T}, s, q_{\text{st}}, q_e, C) \end{aligned} \tag{5}$$

Algorithm 1 Compositional Potential Based Planning

Models: compositional set of N diffusion potential functions $E_\theta^i(q_{1:T}, t, q_{\text{st}}, q_{\text{e}}, C_i)$
Hyperparameters: trajectory horizon T, number of denoising diffusion steps S
Input: start position q_{st}, end position q_{e}, N motion planning constraints $C_{1:N}$
Initialize $q_{1:T}^S \sim \mathcal{N}(0, I)$
for $s = S \dots 1$ **do**
  # Combining Different Energy Potentials Together
  $\epsilon_{\text{comb}} = \sum_{i=1}^{N} \nabla_{q_{1:T}} E_\theta^i(q_{1:T}^s, s, q_{\text{st}}, q_{\text{e}}, C_i)$
  # Transit to Next Diffusion Time Step
  $q_{1:T}^{s-1} = q_{1:T}^s - \gamma\epsilon_{\text{comb}} + \xi, \quad \xi \sim \mathcal{N}\left(0, \sigma_s^2 I\right).$
end for
return $q_{1:T}^0$

To parameterize the energy function $E_\theta(q_{1:T}, s, q_{\text{st}}, q_{\text{e}}, C)$, we use classifier-free guidance scale of 2.0 (Ho & Salimans, 2022) to form a peaker composite energy function conditioned on C. γ and σ_s^2 are diffusion specific scaling constants[1]. The final predicted motion path $q_{1:T}^*$ corresponds to the output $q_{1:T}^0$ after running S steps of optimization from the diffusion potential function.

3.3 Composing Diffusion Potential Functions

Given two separate diffusion potential functions $E_\theta^1(\cdot)$ and $E_\theta^2(\cdot)$, encoding separate constraints for motion planning, we can likewise form a composite potential function $E^{\text{comb}}(\cdot) = E^1(\cdot) + E^2(\cdot)$ by directly summing the corresponding potentials. This potential function E^{comb} will have low energy precisely at motion planning paths $q_{1:T}$ which satisfy both constraints, with sampling corresponding to optimizing this potential function. To sample from the new diffusion potential function E^{comb}, we can follow

$$q_{1:T}^{s-1} = q_{1:T}^s - \gamma\epsilon^{\text{comb}} + \xi, \quad \xi \sim \mathcal{N}\left(0, \sigma_s^2 I\right), \tag{6}$$
$$\epsilon^{\text{comb}} = \nabla_{q_{1:T}}(E_\theta^1(q_{1:T}, s, q_{\text{st}}, q_{\text{e}}, C_1) + E_\theta^2(q_{1:T}, s, q_{\text{st}}, q_{\text{e}}, C_2))$$

This composite potential is the ground truth potential combining constraints if the constraints are independent, which is satisfied if the set of trajectories given constraints are uniformly distributed (see Appendix C), and otherwise serves as approximate proxy to optimize multiple constraints.

Applications of Composing Potential Functions. The ability to combine multiple separate potential functions for motion planning offers a variety of different ways to generalize and extend existing motion planning systems. First, in many motion planning problems, there is often a heterogenous set of different types of constraints or collisions that limit possible configuration space paths. For instance, in autonomous driving, constraints that can arise may include moving pedestrians, traffic lanes, road work, or incoming cars. Oftentimes, we cannot enumerate all potential combinations, but we wish motion planning systems to be able to handle all possible combinations of constraints. Jointly learning a single motion planning model for all constraints may be difficult, as at test time, we may see novel combinations that we do not have training data for. By learning separate diffusion potential fields for each constraint, we can combine them in an ad-hoc manner at test time to deal with arbitrary sets of constraints. We provide two concrete implementations of composing potentials together below and a detailed procedural in Algorithm 1.

[1]A rescaling term at each diffusion step is omitted above for clarity

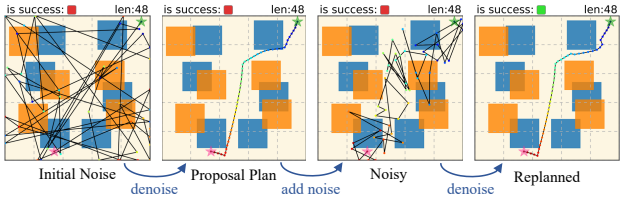


Figure 3: **Visualization of the Motion Refining Scheme**. A proposal plan is first generated by denoising an initial Gaussian noise. If collision is detected, a small noise is added to the proposal and the new plan is generated by denoising the partially noisy trajectory.

Generalization over More Obstacles. Suppose that the potential function E_θ is trained on environments with 4 obstacles $|C| = 4$. However, in test time, we want to generalize to a more complex environment that has 6 obstacles $C' = \{o_1, o_2, o_3, o_4, o_5, o_6\}$. This can be achieved by adding the potentials evaluated on two sets of obstacles, where $C_1 = \{o_1, o_2, o_3, o_4\}$ and $C_2 = \{o_3, o_4, o_5, o_6\}$. This formulation can be further extended to N sets of obstacles $C_{1:N}$ and the composite diffusion potential function is given by:

$$E_\theta^{\text{comb}}(q_{1:T}, s, q_{\text{st}}, q_{\text{e}}, C_{1:N}) = \sum_{i=1}^{N} E_\theta(q_{1:T}, s, q_{\text{st}}, q_{\text{e}}, C_i) \tag{7}$$

Generalization over Static and Dynamic Obstacles. Many real-life scenarios involve *dynamic* real-time interaction. For instance, to construct motion plans for an autonomous vehicle, we must both avoid static lane obstacles as well as moving cars. While static obstacles are often known a priori, the motion patterns of dynamics obstacles often change with time, making it advantageous to be able to combine different dynamic constraints with static ones. We can directly implement this by adding diffusion potential function $E_{\theta_s}^i$ that only trained on static obstacles C_i^s and diffusion potential function $E_{\theta_d}^j$ that only trained on dynamic obstacles C_j^d. In a more general form, to condition on a set of N_1 static obstacles $C_{1:N_1}^s$ with their diffusion potential functions $E_{\theta_s}^{1:N_1}$ and a set of N_2 dynamic $C_{1:N_2}^d$

Algorithm 2 Refining Motion Plans

Model: compositional potential denoiser $f_\theta(q_{1:T}, s, q_{\text{st}}, q_{\text{e}}, C_{1:N})$
Hyperparameters: number of refine attempts R, noise scale k
Input: trajectory $q_{1:T}$, start position q_{st}, end position q_{e}, N constraints $C_{1:N}$
Z = `Get_Collision_Sections`(q)
for $r = 1 \dots R$ **do**
  $q^k_{1:T} = \sqrt{\bar{\alpha}_k} q_{1:T} + (1 - \bar{\alpha}_k)\xi, \quad \xi \sim \mathcal{N}(0, \sigma_t^2 I)$
  $q' = f_\theta(q^k_{1:T}, k, q_{\text{st}}, q_{\text{e}}, C_{1:N})$,
  for all $z_i \in Z$ **do**
    if `is_section_good`$(q'[z_i])$ **then**
      $q[z_i] = q'[z_i]; \quad Z = Z \setminus z_i$
    end if
  end for
end for
return $q_{1:T}$

obstacles with their diffusion potential functions $E^{1:N_2}_{\theta_d}$, the composite diffusion potential function can be written as:

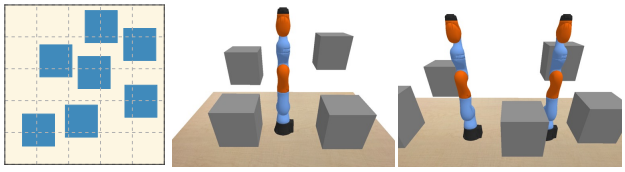
$$E^{\text{comb}}_\theta(q_{1:T}, s, q_{\text{st}}, q_{\text{e}}, [C^s_{1:N_1}, C^d_{1:N_2}]) = \sum_{i=1}^{N_1} E^i_{\theta_s}(q_{1:T}, s, q_{\text{st}}, q_{\text{e}}, C^s_i) + \sum_{j=1}^{N_2} E^j_{\theta_d}(q_{1:T}, t, q_{\text{st}}, q_{\text{e}}, C^d_j) \tag{8}$$

3.4 Refining Motion Plans

In practice, the predicted motion plan $q_{1:T}$ might occasionally contains sections that violate the constraints of the environment (i.e., collide with obstacles). To tackle this issue, both classical and learned motion planners (Kuffner & LaValle, 2000; Qureshi et al., 2019) provide mechanisms to refine trajectories subject to collisions in configuration space. With diffusion potential fields, we can likewise refine a trajectory, $q_{1:T}$ with collision, by locally perturbing it into a noisy trajectory $q^k_{1:T}$ defined by the k-th step of the diffusion forward process:

$$q^k_{1:T} = \sqrt{\bar{\alpha}_k} q_{1:T} + (1 - \bar{\alpha}_k)\xi, \quad \xi \sim \mathcal{N}(0, \sigma_t^2 I) \tag{9}$$

as in (Ho et al., 2020). A new motion plan $q'_{1:T}$ can be obtained by denoising the noisy trajectory following Equation 5, where $q'_{1:T} = f_\theta(q^k_{1:T}, k, q_{\text{st}}, q_{\text{e}}, C_{1:N})$ and $f_\theta(\cdot)$ is an iterative diffusion potential denoiser that outputs a clean trajectory. To fix the problematic trajectory $q_{1:T}$, the collision sections in $q_{1:T}$ will be replaced by the corresponding sections in $q'_{1:T}$ if these new sections are coherent and collision-free. This refining procedural can be repeated until a desired trajectory is found. The warm-start denoising scheme can enable faster re-planning and maintain the morphology of the original plan, supporting planning on energy-critical mobile devices or when the plan has been executed. Algorithm 2 displays the complete refining pipeline and Figure 3 provides a corresponding visualization.



Maze2D KUKA 7D Dual KUKA 14D

Figure 4: **Environment Demonstration.** Maze2D: a point robot moving in 2D workspace with the highlighted blocks as obstacles. KUKA: robotic arm with 7 DoF operating on a tabletop. The grey cuboids are obstacles. Dual KUKA 14D: Two side by side KUKA arms operate simultaneously.

3.5 Probabilistic Completeness

In this section, we will show the probabilistic completeness of our method. Let $f_\theta(q_{1:T})$ denote the probability density function of the output distribution $\mathcal{D}_o$ of our diffusion potential model. In such learned neural distribution, all data points $q_{1:T}$ are assigned positive density, that is,

$$\forall q_{1:T}, \; f_\theta(q_{1:T}) > 0 \tag{10}$$

Define $\mathcal{J}_c$ as the a set of all valid trajectories subject to constraint C. There always exists a small interval in the vicinity of a random trajectory $q^c_{1:T} \in \mathcal{J}_c$, such that

$$[q^c_{1:T} - \tau, q^c_{1:T} + \tau] \subseteq \mathcal{J}_c, \quad \tau > 0, \tag{11}$$

i.e., all trajectories in the interval satisfy the given constraint C. Let $\mathbb{P}_\tau$ denote the probability for our model to sample a trajectory from the interval, and according to equation 10 we have,

$$\mathbb{P}_\tau = \int_{q^c_{1:T}-\tau}^{q^c_{1:T}+\tau} f(x)dx > 0 \tag{12}$$

Let A_n denote the event that there is at least one trajectory $q_{1:T} \in \mathcal{J}_c$ among n sampled trajectories. Clearly, as the number of samples approaches infinity, event A will happen almost surely, i.e.,

$$\lim_{n\to\infty} \mathbb{P}\left(A_n\right) = 1 \tag{13}$$

Hence, our method is probabilistically complete.

4 Experiments

In this section, we first describe our environments and baselines in Section 4.1. Next, in Section 4.2, we discuss our experiments on the base environments and the motion refining algorithm. Following, in Section 4.3, we present the compositionality results by evaluating our motion planner on composite environments. Then, we describe the real world motion planning performance in Section 4.4.

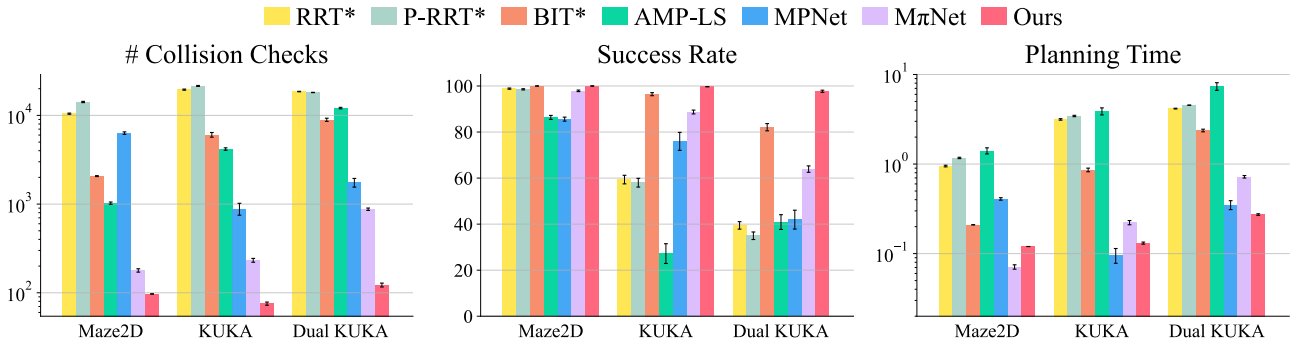


Figure 5: **Quantitative Comparisons in Motion Planning Environments.** Three metrics of three environments from 2D to 14D are reported. From left to right: a) number of collision checks, b) success rate, c) planning time.

4.1 Environments and Baselines

We first classify the environments that we evaluated on to 3 categories by the level of generalization capability:

- **Base Environment:** same number of constraints as in training phase, constraints sampled from the same distribution as in training.
- **Composite Same Environment:** more constraints than training phase, constraints sampled from the same distribution as in training.
- **Composite Different Environment:** more constraints than training phase, constraints sampled from different distributions.

Our simulated motion planning environments are listed below and shown in Figure 4. See Appendix A.1 for more details. In all environments, the motion planning task is to generate a feasible trajectory from the start state to the goal state. The agent trajectory is represented in C-space, while obstacles are represented in workspace.

- **Maze2D**. A point-robot moving in a 2D workspace. The configuration space is the x-y coordinate of the agent. We offer two variants: *Static Maze2D* where obstacles stay in the same locations and *Dynamic Maze2D* where obstacles are moving in randomly generated linear trajectories.
- **KUKA**. A KUKA arm of 7 DoF operating on a tabletop in a 3D workspace. The start/goal is given as the 7D joint state of the KUKA arm.
- **Dual KUKA**. Two KUKA arms are placed side by side on a tabletop and operating simultaneously. The start/goal is given as the joint states of two KUKA arms (14 DoF).

Baselines. For non-learning motion planning methods, our baselines include classic sampling-based planning baseline RRT* (Karaman & Frazzoli, 2011), sampling-based method with potential functions based heuristic P-RRT* (Qureshi & Ayaz, 2016), advanced sampling-based method BIT* (Gammell et al., 2015), sampling-based method for dynamic environments SIPP (Phillips & Likhachev, 2011), and traditional potential-based method RMP (Ratliff et al., 2018).

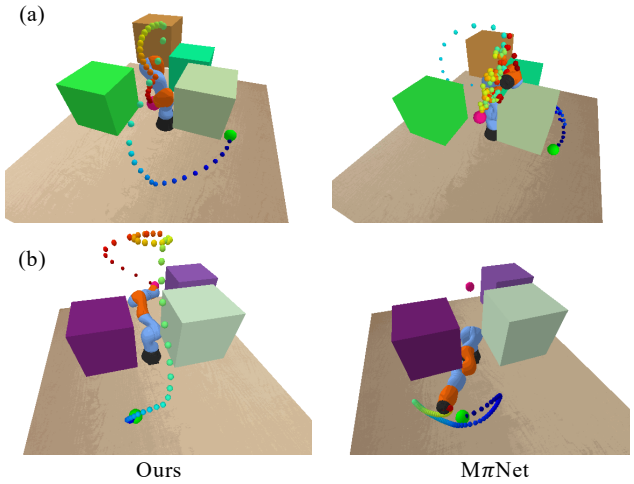


Figure 6: **Qualitative Motion Plans in KUKA Environment.** Results of two motion planning problem are shown in two rows. The large green/pink ball indicates the start/goal state. Our method in the left column generates smooth and near-optimal trajectories, while in row (a), the trajectory of MπNet chooses a longer route from behind and gets stuck in some local regions, and in row (b), MπNet cannot pass through the narrow passage and keeps hovering near the start state.

For learned motion planners, we compare our method with: MPNet (Qureshi et al., 2019), MπNet (Fishman et al., 2023), and AMP-LS (Yamada et al., 2023). MPNet is trained on trajectories with sparse waypoints and use MLPs to encode environment configuration and predict the next robot state. MπNet is trained on dense trajectory waypoints and predicts the movement vector instead of the next robot state. AMP-LS encodes the robot state into a latent feature and approaches the goal state by iteratively using the gradient of several planning losses to update the latent. A sequence of latents are then decoded and form a trajectory. In evaluation, all the start/goal states and environment configurations are unseen to the models. For each experiment, we evaluate on 100 different environments with 20 randomly sampled starts and goals in each environment. No re-training is performed in test time.

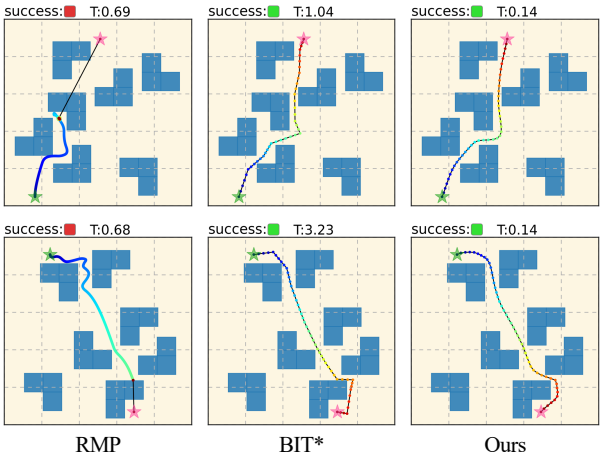


Figure 7: **Qualitative Performance on Environments with Concave Obstacles.** Potential-based planning method RMP is prone to local minima. BIT* is able to find a feasible solution, but the trajectories are rough and require much longer planning time, especially for the harder problem in the second row. Trajectories generated by our planner are both smooth and short in length.

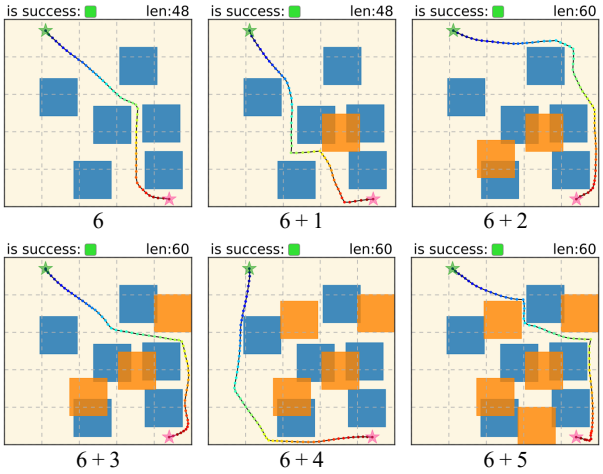


Figure 8: **Qualitative Compositional Generalization over More Obstacles.** Two models that trained on only 6 obstacles are composed and tested on out-of-distribution environments with 7, 8, 9, 10, 11 obstacles, respectively.

4.2 Performance on Base Environments

We first evaluate motion planning performance in each base environment: randomly generated environments that follow the same procedural generation pipeline as the training environments. Quantitative results are shown in Figure 5 and Appendix B.1. We include the full details of evaluation setup in Appendix A.2.3.

Comparison to Sampling-based Planners. We set the planning time limit to 5 seconds for all sampling-based methods. Both RRT*, P-RRT*, and BIT* can succeed in the base Maze2D environment. However, their success rates suffer from a significant degradation when the dimension of the configuration space increases. The planning time of the sampling-based planners also rises dramatically as the dimension of the space increases. By contrast, the success rates of our method surpass all baselines and are consistent across different environments, achieving this in less than 11% of BIT*'s planning time and requiring up to two orders of magnitude fewer collision checks.

Comparison to Learning-based Planners. We also compare to three learning-based motion planning baselines: MPNet, MπNet, and AMP-LS. Our method outperforms all baselines in both success rate and number of collision check. In Dual KUKA, our method leads the state-of-the-art learning-based planner MπNet by nearly 35% in success rate while with less than 40% of its planning time and 7 times less of its collision checks. We also observe that the performance of other planners drop quickly as the difficulty increases, while our planner performs steadily across all environments. Though the planning time of MπNet in Maze2D is slightly shorter than ours, the gap is closing as the dimension of the configuration space increases, and in Dual KUKA, our planner requires less planning time than all baselines, probably due to the expensive cost of collision checks and higher task complexity. Qualitative comparisons are shown in Figure 6 and 18.

	$R=3$		$R=5$		$R=10$	
Env	Before	After	Before	After	Before	After
Maze2D	96.3	99.8	95.3	99.0	95.8	100.0
KUKA	71.3	90.0	69.5	94.3	69.8	94.8
Dual KUKA	45.5	69.8	47.3	77.3	47.0	80.8

Table 1: **Quantitative Results of Motion Plan Refining**. Success rates before and after motion refining are shown. R denotes the number of refining attempts. The proposed refining method can consistently boost success rates on three base environments.

	Maze2D – Convex		Maze2D – Concave	
Method	Success	Time	Success	Time
RRT*	98.8	0.95	92.1	2.14
P-RRT*	98.5	1.17	85.9	2.69
BIT*	100.0	0.21	100.0	0.45
MPNet	88.4	0.21	84.3	0.38
MπNet	97.9	**0.07**	96.9	**0.10**
MPD	77.9	2.99	44.4	3.93
RMP	64.9	0.13	28.0	0.34
Ours	**100.0**	**0.12**	**100.0**	**0.15**

Table 2: **Quantitative Performance on Convex and Concave Obstacles.** Motion planning performance on Maze2D environments with 6 convex obstacles (left) and 7 concave obstacles (right). The proposed diffusion potential planner outperforms the traditional potential-based method RMP and diffusion motion planning counterpart MPD by a margin. While both our method and the advanced sampling-based method BIT* can solve all the problems, ours requires significantly less planning time than BIT*.

Environments with Concave Obstacles. We construct additional Maze2D environments where obstacles are con-

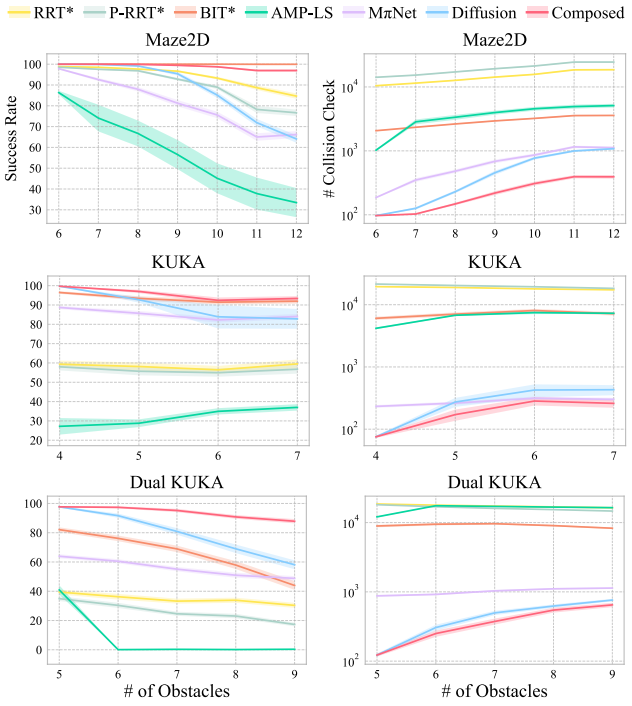


Figure 9: **Compositional Generalization.** Quantitative comparisons of different planner on composite environments. The shaded areas represent standard errors. Each graph's leftmost column displays results for environments that contain the same number of obstacles as encountered during training. By composing potentials at test time, our method (red line) can generalize to environments with much more obstacles than training time.

cave to demonstrate the capability of our learned potential motion planner to avoid local minima. As shown in Figure 7, potential-based method RMP tends to get trapped in local minima in environments filled with concave obstacles. In addition, we observe that BIT* is able to find a feasible motion plan, but the morphology of the proposed trajectories are sharp and irregular. We further present the quantitative results on base Maze2D environments in Table 2, where we include an additional diffusion-based motion planning approach (Carvalho et al., 2023). Notably, our diffusion potential planner is the only learning-based approach that successfully solves every motion planning problems in our evaluation set. Although the advanced sampling-based method BIT* can also find all the solutions, its average planning time is significantly longer, taking three times longer than our method in Maze2D – Concave. More results are provided in Appendix B.4.

Motion Refining. We present quantitative and qualitative results of refining motion plans in Table 1 and Figure 3. The gain of refining motion plans increases as the dimension of the environment increases. As shown in Table 1, the success rate generally increases as we increase the number of refining attempts R, but the gain gradually saturates in 10 attempts. In this case, the proposed trajectory probably suffers from a catastrophic collision, and to obtain a successful

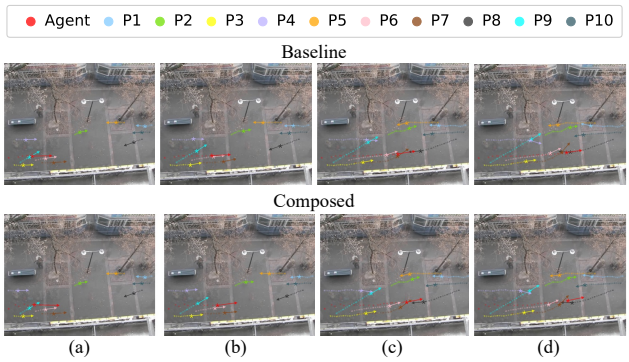


Figure 10: **Qualitative Real World Motion Plans, *Hotel* Scene.** The composed model provides long-horizon motion plan that avoid 10 pedestrians, while only trained on 5 pedestrians. In column (a) and (b), the composed plan is aware of P1 (cyan) and P6 (pink) and overtakes them from above, while the baseline model runs into them. In column (c), the composed motion plan chooses to move faster so as to pass through the intersection with P7 (brown) before P7 arrives, but the baseline motion plan results in a collision due to its slower speed. In column (d), the composed plan choose to go upward to avoid the oncoming P8 (black).

	Base Dynamic			Static 1 + Dynamic			Static 2 + Dynamic		
M	Suc	Time	Col	Suc	Time	Col	Suc	Time	Col
SIPP	69.9	32.2	1M+	70.4	185.5	1.7M+	74.0	98.7	1.3M+
Ours	**99.5**	**0.13**	**168.8**	**96.6**	**4.31**	**828.6**	**97.5**	**4.23**	**646.4**

Table 3: **Quantitative Results on Base Dynamic and Static + Dynamic on Maze2D.** Static 1 and Static 2 refer to two different static Maze2D environments. Our planner can generalize to environments with both static and dynamic obstacles while only separately trained on static environments or dynamic environments.

motion plan, the diffusion potential model might need to resample a trajectory from pure noise.

4.3 Performance on Composite Environments

Composing Obstacles. We first evaluate the compositionality by adding obstacles to the environments. Qualitative results of composite Maze2D environments are given in Figure 8, where we train our model on 6 obstacles and evaluate on environments with up to 11 obstacles. Each orange block represents an extra obstacle added to the test time environments. As we can see, the composed model effectively proposes different trajectories according to the presented obstacles by sampling from the composite potential. We report the quantitative performance on three environments with different configuration space dimensions in Figure 9. More results are shown in Appendix B.2. In addition, we further include a baseline which directly learns demonstrations of different numbers and types of obstacles in a single diffusion potential function in Appendix B.6.

Composing Multiple Constraints. We then investigate the compositionality to combine two different diffusion potential functions together, i.e., models trained on completely

different environments. Specifically, we separately train a model on 6 small obstacles (*Static 1*) and a model on 3 large obstacles (*Static 2*) and evaluate the composed model on environments where both small and large obstacles are presented. Quantitative and qualitative results are shown in Table 13 and Figure 12. Moreover, we compose the models trained on static environments with another model trained on dynamic environments, by which the composed model can generalize to environments where both static and dynamic obstacles are presented, namely *Static 1 + Dynamic* and *Static 2 + Dynamic*. The corresponding quantitative results are shown in Table 3. The planning time limit for SIPP is set to 60s/300s for base/composite dynamic environments. Please refer to Figure 14 and Figure 15 in Appendix B.3 for more qualitative results.

4.4 Performance on Real World Datasets

Finally, we evaluate the effectiveness of our method on the real world ETH/UCY (Pellegrini et al., 2010; Lerner et al., 2007) dataset. The dataset we used consists of 6 scenes (*ETH, Hotel, Zara01, Zara02, Students01, Students03*), where each scene contains human trajectories in world-coordinates collected by manual annotation from a bird-eye-view camera. Our focus is to investigate if our model can propose successful trajectories given the start and goal locations of an agent in a random, cluttered street-level real-world interaction. Specifically, the planner is trained to predict the trajectory of the agent (highlighted in red), conditioned on the trajectories of 5 other pedestrians. The training data contains 5 scenes and the held-out scene is used for evaluation. In Figure 17, we present the qualitative results where 5 other pedestrians are presented. We also evaluate on 10 presented pedestrians by composing two potential functions each constrained by 5 pedestrians, as illustrated in Figure 10. Details settings and qualitative results are presented in Appendix B.5.

5 Discussion

Limitations. Our existing formulation of potential based diffusion motion planner has several limitations. First, although our motion trajectory is accurate, it is often sub-optimal, e.g., there exists a shorter path from start to goal. This may be addressed by adding an additional potential to reach the goal as soon as possible. Second, our approach to composing potentials scales linearly with the number of composed models, requiring significantly more computation power with additional models. This can be remedied by having different potential operate on shared features in a network.

Conclusion. In this work, we have introduced a potential based diffusion motion planner. We first formulate our diffusion potential motion planner and describe its connections and advantages over traditional potential based planner. We illustrate the motion planning performance of our approach in terms of success rate, planning time, and the number of collision checks over motion planning problems with dimension of 2D, 7D, 14D. We further illustrate the compositionality of the approach, enabling generalization to both new objects and new combinations of motion constraints. Finally, we illustrate the potential of our work on real world scenes with multi-agent interaction.

Acknowledgements

We acknowledge support from NSF grant 2214177; from AFOSR grant FA9550-22-1-0249; from ONR MURI grant N00014-22-1-2740; from ARO grant W911NF-23-1-0034; and from the Samsung Global Research Outreach Program. Yilun Du is supported by a NSF Graduate Fellowship. We thank the anonymous reviewers for their careful review. Our research was conducted using computational resources at the Center for Computation and Visualization at Brown University.

Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, which we believe should be similar to a generic machine learning paper. For example, if the motion pattern in the dataset is biased, the motion plans generated by our model might have similar biases as the training dataset. No other issues we feel must be specifically highlighted here.

References

Ajay, A., Du, Y., Gupta, A., Tenenbaum, J. B., Jaakkola, T. S., and Agrawal, P. Is conditional generative modeling all you need for decision making? In *The Eleventh International Conference on Learning Representations*, 2023.

Bency, M. J., Qureshi, A. H., and Yip, M. C. Neural path planning: Fixed time, near-optimal path generation via oracle imitation. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 3965–3972. IEEE, 2019.

Carvalho, J., Baierl, M., Urain, J., and Peters, J. Conditioned score-based models for learning collision-free trajectory generation. In *NeurIPS 2022 Workshop on Score-Based Methods*, 2022.

Carvalho, J., Le, A. T., Baierl, M., Koert, D., and Peters, J. Motion planning diffusion: Learning and planning

of robot motions with diffusion models. *arXiv preprint arXiv:2308.01557*, 2023.

Chang, J., Ryu, H., Kim, J., Yoo, S., Seo, J., Prakash, N., Choi, J., and Horowitz, R. Denoising heat-inspired diffusion with insulators for collision free motion planning. *arXiv preprint arXiv:2310.12609*, 2023.

Chaplot, D. S., Pathak, D., and Malik, J. Differentiable spatial planning using transformers. In *International Conference on Machine Learning*, pp. 1484–1495. PMLR, 2021.

Chen, H., Lu, C., Ying, C., Su, H., and Zhu, J. Offline reinforcement learning via high-fidelity generative behavior modeling. *arXiv preprint arXiv:2209.14548*, 2022.

Chi, C., Feng, S., Du, Y., Xu, Z., Cousineau, E., Burchfiel, B., and Song, S. Diffusion policy: Visuomotor policy learning via action diffusion. *arXiv preprint arXiv:2303.04137*, 2023.

Choudhury, S., Gammell, J. D., Barfoot, T. D., Srinivasa, S. S., and Scherer, S. Regionally accelerated batch informed trees (rabit*): A framework to integrate local information into optimal path planning. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 4207–4214. IEEE, 2016.

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

Dalal, M., Mandlekar, A., Garrett, C., Handa, A., Salakhutdinov, R., and Fox, D. Imitating task and motion planning with visuomotor transformers. *arXiv preprint arXiv:2305.16309*, 2023.

Du, Y. and Mordatch, I. Implicit generation and modeling with energy based models. *Advances in Neural Information Processing Systems*, 32, 2019.

Du, Y., Durkan, C., Strudel, R., Tenenbaum, J. B., Dieleman, S., Fergus, R., Sohl-Dickstein, J., Doucet, A., and Grathwohl, W. S. Reduce, reuse, recycle: Compositional generation with energy-based diffusion models and mcmc. In *International Conference on Machine Learning*, pp. 8489–8510. PMLR, 2023.

Elbanhawi, M. and Simic, M. Sampling-based robot motion planning: A review. *Ieee access*, 2:56–77, 2014.

Fang, X., Garrett, C. R., Eppner, C., Lozano-Pérez, T., Kaelbling, L. P., and Fox, D. Dimsam: Diffusion models as samplers for task and motion planning under partial observability. *arXiv preprint arXiv:2306.13196*, 2023.

Fiorini, P. and Shiller, Z. Motion planning in dynamic environments using velocity obstacles. *The international journal of robotics research*, 17(7):760–772, 1998.

Fishman, A., Murali, A., Eppner, C., Peele, B., Boots, B., and Fox, D. Motion policy networks. In *Conference on Robot Learning*, pp. 967–977. PMLR, 2023.

Gammell, J. D., Srinivasa, S. S., and Barfoot, T. D. 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*, pp. 2997–3004. IEEE, 2014.

Gammell, J. D., Srinivasa, S. S., and Barfoot, T. D. Batch informed trees (bit*): Sampling-based optimal planning via the heuristically guided search of implicit random geometric graphs. In *2015 IEEE international conference on robotics and automation (ICRA)*, pp. 3067–3074. IEEE, 2015.

Ha, H., Florence, P., and Song, S. Scaling up and distilling down: Language-guided robot skill acquisition. *arXiv preprint arXiv:2307.14535*, 2023.

Hendrycks, D. and Gimpel, K. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016.

Ho, J. and Salimans, T. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*, 2022.

Ho, J., Jain, A., and Abbeel, P. Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems*, volume 33, pp. 6840–6851, 2020.

Huang, S., Wang, Z., Li, P., Jia, B., Liu, T., Zhu, Y., Liang, W., and Zhu, S.-C. Diffusion-based generation, optimization, and planning in 3d scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 16750–16761, June 2023.

Ichter, B. and Pavone, M. Robot motion planning in learned latent spaces. *IEEE Robotics and Automation Letters*, 4 (3):2407–2414, 2019.

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

Janson, L., Schmerling, E., Clark, A., and Pavone, M. Fast marching tree: A fast marching sampling-based method for optimal motion planning in many dimensions. *The International journal of robotics research*, 34(7):883–921, 2015.

Johnson, J. J., Kalra, U. S., Bhatia, A., Li, L., Qureshi, A. H., and Yip, M. C. Motion planning transformers: A motion planning framework for mobile robots. *arXiv preprint arXiv:2106.02791*, 2021.

Kalakrishnan, M., Chitta, S., Theodorou, E., Pastor, P., and Schaal, S. Stomp: Stochastic trajectory optimization for motion planning. In *2011 IEEE international conference on robotics and automation*, pp. 4569–4574. IEEE, 2011.

Kang, B., Ma, X., Du, C., Pang, T., and Yan, S. Efficient diffusion policies for offline reinforcement learning. *arXiv preprint arXiv:2305.20081*, 2023.

Kapelyukh, I., Vosylius, V., and Johns, E. Dall-e-bot: Introducing web-scale diffusion models to robotics. *IEEE Robotics and Automation Letters*, 2023.

Karaman, S. and Frazzoli, E. Sampling-based algorithms for optimal motion planning. *The international journal of robotics research*, 30(7):846–894, 2011.

Kavraki, L. E., Svestka, P., Latombe, J.-C., and Overmars, M. H. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE transactions on Robotics and Automation*, 12(4):566–580, 1996.

Khatib, O. Real-time obstacle avoidance for manipulators and mobile robots. *The international journal of robotics research*, 5(1):90–98, 1986.

Kingma, D. P. and Welling, M. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Koren, Y., Borenstein, J., et al. Potential field methods and their inherent limitations for mobile robot navigation. In *Icra*, volume 2, pp. 1398–1404, 1991.

Kuffner, J. J. and LaValle, S. M. 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)*, volume 2, pp. 995–1001. IEEE, 2000.

Laumond, J.-P. et al. *Robot motion planning and control*, volume 229. Springer, 1998.

LaValle, S. M. *Planning algorithms*. Cambridge university press, 2006.

Lawson, D. and Qureshi, A. H. Control transformer: Robot navigation in unknown environments through prm-guided return-conditioned sequence modeling. *arXiv preprint arXiv:2211.06407*, 2022.

Le, A. T., Chalvatzaki, G., Biess, A., and Peters, J. Accelerating motion planning via optimal transport. In *IROS 2023 Workshop on Differentiable Probabilistic Robotics: Emerging Perspectives on Robot Learning*, 2023.

LeCun, Y., Chopra, S., Hadsell, R., Ranzato, M., and Huang, F. A tutorial on energy-based learning. *Predicting structured data*, 1(0), 2006.

Lerner, A., Chrysanthou, Y., and Lischinski, D. Crowds by example. In *Computer graphics forum*, volume 26, pp. 655–664. Wiley Online Library, 2007.

Li, W., Wang, X., Jin, B., and Zha, H. Hierarchical diffusion for offline decision making. 2023.

Liang, Z., Mu, Y., Ding, M., Ni, F., Tomizuka, M., and Luo, P. Adaptdiffuser: Diffusion models as adaptive self-evolving planners. In *International Conference on Machine Learning*, 2023.

Liu, N., Li, S., Du, Y., Torralba, A., and Tenenbaum, J. B. Compositional visual generation with composable diffusion models. *arXiv preprint arXiv:2206.01714*, 2022.

Mukadam, M., Dong, J., Yan, X., Dellaert, F., and Boots, B. Continuous-time gaussian process motion planning via probabilistic inference. *The International Journal of Robotics Research*, 37(11):1319–1340, 2018.

Pellegrini, S., Ess, A., and Van Gool, L. Improving data association by joint modeling of pedestrian trajectories and groupings. In *Computer Vision–ECCV 2010: 11th European Conference on Computer Vision, Heraklion, Crete, Greece, September 5-11, 2010, Proceedings, Part I 11*, pp. 452–465. Springer, 2010.

Phillips, M. and Likhachev, M. Sipp: Safe interval path planning for dynamic environments. In *2011 IEEE international conference on robotics and automation*, pp. 5628–5635. IEEE, 2011.

Qureshi, A. H. and Ayaz, Y. Potential functions based sampling heuristic for optimal path planning. *Autonomous Robots*, 40:1079–1093, 2016.

Qureshi, A. H., Simeonov, A., Bency, M. J., and Yip, M. C. Motion planning networks. In *2019 International Conference on Robotics and Automation (ICRA)*, pp. 2118–2124. IEEE, 2019.

Ratliff, N., Zucker, M., Bagnell, J. A., and Srinivasa, S. Chomp: Gradient optimization techniques for efficient motion planning. In *2009 IEEE international conference on robotics and automation*, pp. 489–494. IEEE, 2009.

Ratliff, N. D., Issac, J., Kappler, D., Birchfield, S., and Fox, D. Riemannian motion policies. *arXiv preprint arXiv:1801.02854*, 2018.

Rezende, D. J. and Viola, F. Taming vaes. *arXiv preprint arXiv:1810.00597*, 2018.

Saha, K., Mandadi, V., Reddy, J., Srikanth, A., Agarwal, A., Sen, B., Singh, A., and Krishna, M. Edmp: Ensemble-of-costs-guided diffusion for motion planning. *arXiv preprint arXiv:2309.11414*, 2023.

Sohl-Dickstein, J., Weiss, E., Maheswaranathan, N., and Ganguli, S. Deep unsupervised learning using nonequilibrium thermodynamics. In *International Conference on Machine Learning*, pp. 2256–2265. PMLR, 2015.

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

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

Teli, T. A. and Wani, M. A. A fuzzy based local minima avoidance path planning in autonomous robots. *International Journal of Information Technology*, 13:33–40, 2021.

Toma, A.-I., Jaafar, H. A., Hsueh, H.-Y., James, S., Lenton, D., Clark, R., and Saeedi, S. Waypoint planning networks. *arXiv preprint arXiv:2105.00312*, 2021.

Urain, J., Funk, N., Peters, J., and Chalvatzaki, G. 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)*, pp. 5923–5930. IEEE, 2023.

Van den Berg, J., Lin, M., and Manocha, D. Reciprocal velocity obstacles for real-time multi-agent navigation. In *2008 IEEE international conference on robotics and automation*, pp. 1928–1935. Ieee, 2008.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Wang, L., Ames, A. D., and Egerstedt, M. Safety barrier certificates for collisions-free multirobot systems. *IEEE Transactions on Robotics*, 33(3):661–674, 2017.

Wang, Z., Hunt, J. J., and Zhou, M. Diffusion policies as an expressive policy class for offline reinforcement learning. In *The Eleventh International Conference on Learning Representations*, 2023.

Xie, M., Van Wyk, K., Li, A., Rana, M. A., Wan, Q., Fox, D., Boots, B., and Ratliff, N. Geometric fabrics for the acceleration-based design of robotic motion. *arXiv preprint arXiv:2010.14750*, 2020.

Yamada, J., Hung, C.-M., Collins, J., Havoutis, I., and Posner, I. Leveraging scene embeddings for gradient-based motion planning in latent space. *arXiv preprint arXiv:2303.03364*, 2023.

Yang, M., Du, Y., Dai, B., Schuurmans, D., Tenenbaum, J. B., and Abbeel, P. Probabilistic adaptation of text-to-video models. *arXiv preprint arXiv:2306.01872*, 2023a.

Yang, Z., Mao, J., Du, Y., Wu, J., Tenenbaum, J. B., Lozano-Pérez, T., and Kaelbling, L. P. Compositional diffusion-based continuous constraint solvers. *arXiv preprint arXiv:2309.00966*, 2023b.

Yonetani, R., Taniai, T., Barekatain, M., Nishimura, M., and Kanezaki, A. Path planning using neural a* search. In *International conference on machine learning*, pp. 12029–12039. PMLR, 2021.

Yu, C. and Gao, S. Reducing collision checking for sampling-based motion planning using graph neural networks. *Advances in Neural Information Processing Systems*, 34:4274–4289, 2021.

Yun, X. and Tan, K.-C. A wall-following method for escaping local minima in potential field based motion planning. In *1997 8th International Conference on Advanced Robotics. Proceedings. ICAR'97*, pp. 421–426. IEEE, 1997.

A Appendix

In this appendix, we first present our dataset details in Section A.1. Next, we provide implementation details in Section A.2, such as model architecture, training and evaluation setups, hyperparameters. Further, We show additional quantitative and qualitative results in Section B, including base environments, composite environments, and real-world datasets. Lastly, in Section C, we give a proof on the optimality of composing potentials of sets of constraints.

A.1 Dataset Details

In this section, we present details of the three base environments. Each dataset consists of feasible trajectories of randomly sampled start and goal states. All datasets are collected by BIT*(Gammell et al., 2015). All environments except Maze2D are simulated via PyBullet (Coumans & Bai, 2016–2021).

Maze2D. The workspace is a 5×5 square and the agent is a point-robot whose state is its x-y coordinate in the workspace. In the base Maze2D environment (*Static 1*), the obstacles are 6 square blocks of size 1×1. We construct another Maze2D environment (*Static 2*) where obstacles are 3 square blocks of size 1.4×1.4. In addition, we construct a Maze2D environment with 7 concave obstacles. For simplicity, the volume of the robot agent is ignored – collision happens when the location is inside the region of an obstacle. The training data contains 3,000 different environment configurations and 25,000 states for each environments (approximately 500 trajectories).

KUKA. One KUKA LBR iiwa robotic arm is placed at the center $(0, 0, 0)$ in world coordinate and obstacles are given as cubic of length 0.40 meter and are randomly placed in the surrounding of the robot. The training data contains 2,000 different environment configurations and 25,000 states for each environments.

Dual KUKA. Two KUKA LBR iiwa robotic arms are placed side by side on a tabletop. One is at world coordinate $(-0.5, 0, 0)$ and the other is at $(0.5, 0, 0)$. Obstacles are given as cubic of length 0.44 meters and are randomly placed in the surrounding of the robots. The training data contains 2,500 different environment configurations and 25,000 states for each environment.

ETH/UCY. The dataset we used contains 6 scenes, where in each scene there are hundreds of street-level pedestrians trajectories. In each scene, a video is recorded using a fixed bird-eye-view camera facing the street, and the coordinates of each pedestrian are manually annotated according to the recorded video. We use real-world pedestrian trajectories of 5 scenes to train our model and evaluate on the one held-out scene.

A.2 Implementation details

Software:. The computation platform is installed with Red Hat 7.9, Python 3.8, PyTorch 1.10.1, and Cuda 11.1

Hardware:. For each of our experiments, we used 1 RTX 3090 GPU.

A.2.1 Energy-based Diffusion Model

Model Architecture. The diffusion potential model f_θ consists of a CNN trajectory denoiser based on U-Net similar to (Ajay et al., 2023) and a constraint (i.e., environment configuration) encoder. The U-Net contains repeated residual blocks where each block consists of two temporal convolutions followed by GroupNorm and SiLU nonlinearity (Hendrycks & Gimpel, 2016). The constraint encoder uses the Transformer encoder structure (Vaswani et al., 2017), whose input is a set of obstacle locations for static environments or obstacle trajectories for dynamic environments. We remove the positional embedding, since the obstacles information should be permutation invariance with each other. We concatenate the learned class token from transformer with the time embedding and feed the concatenated tensor to temporal convolution blocks in U-Net for denoising. More details of the models are shown in Table A.2.1 and Table A.2.1. Note that we do not further explore the selection of the concrete model architecture, but we believe that some more advanced architectures could further improve our performance.

Hyperparameters	Value
Base Feature Channels	64
Feature Dimension Scale	(64, 256, 512)
Groups in GroupNorm	8
Nonlinearity	SiLU

Table 4: Hyperparameter of U-Net.

Hyperparameters	Value
Base Embedding Channel	64
Transformer Layers	3
Attention Heads	1
Nonlinearity	SiLU

Table 5: Hyperparameter of Constraint Encoder.

Energy Parameterization. To encode the energy $E(\cdot)$ of as in equation 4, we use L2 energy-parameterization as given in equation 14. For more details on Energy-based Diffusion Model, please refer to (Du et al., 2023).

$$E_\theta^{L2}(x,t) = \frac{1}{2}||f_\theta(x,t)||^2 \tag{14}$$

A.2.2 Training Details

Training Pipeline. In training, the dataloader randomly samples trajectories of length equal to the training horizon from the whole dataset. We provide detailed hyperparameters for training our model in Table 6. We do not apply any hyperparameter search nor learning rate scheduler. The training time of our model is approximately two days, but we observe that the performance is close to saturation within one day.

Hyperparameters	Value
Horizon	48
Diffusion Time Step	100
Probability of Condition Dropout	0.2
Iterations	2M
Batch Size	512
Optimizer	Adam
Learning Rate	2e-4

Table 6: Hyperparameters of Diffusion Potential Motion Planner (Training)

A.2.3 Evaluation Details

Our Evaluation Pipeline. The input of the planner is the start state, goal state, and the environment constraints (e.g., obstacle locations represented in workspace), and the output is the proposed trajectory. In test time, the number of denoising timestep is set to 10 by using DDIM (Song et al., 2020). The intermediate noise scale eta in DDIM is set to 0 for base environments and 1 for composite environments. The evaluation pipeline of our model consists of three phases: Propose Motion Plan Candidates, Candidate Selection, and Motion Refining. The planner first generates multiple candidate trajectories using Algorithm 1. It then accesses the environment and performs collision check to select a successful trajectory from the candidates. Finally, if no desired candidate is found, it will execute the motion refining as in Algorithm 2. Hyperparameters used in evaluation is detailed in Table 7 and Table 8. We observe that our planner can directly solve all the problems in Maze2D, hence we do not use replanning during the evaluation of Maze2D environments.

Baselines Evaluation Pipeline. We try our best to re-implement every baseline and follow their original settings. For MPNet (Qureshi et al., 2019), we follow their implementation and use bi-directional path generation in test time. As for AMP-LS, we implement a Variational Auto-Encoder (VAE) (Kingma & Welling, 2013) to encode the robot pose state and leverage GECO loss (Rezende & Viola, 2018) in path optimization. MπNet does not provide a replan scheme in their design. For fair comparison, we boost MπNet with replan by backtracing to previous timestep and adding a small random noise for restart when any collisions are detected.

B Additional Results

In this section, we provide more quantitative and qualitative motion planning results. In Section B.1, we present additional planning results on the base environments. In Section B.2, we show planning performance on composite same environments.

Hyperparameters	Value
Horizon	48
DDIM Time Step	8
DDIM eta	0.0
Guidance Scale	2.0
# of Trajectory Candidate	20
# of Refine Attempts R	0
Refine Noise Scale k	–

Table 7: Hyperparameters of Diffusion Potential Motion Planner (Evaluation on Maze2D)

Hyperparameters	Value
Horizon	52
DDIM Time Step	10
DDIM eta	0.0
Guidance Scale	2.0
# of Trajectory Candidate	20
# of Refine Attempts R	5
Refine Noise Scale k	3

Table 8: Hyperparameters of Diffusion Potential Motion Planner (Evaluation on KUKA and Dual KUKA)

In Section B.3, we provide planning performance on composite different environments, including composing two different static models and composing a static model and a dynamic model. In Section B.4, we show motion planning performance on a more challenging Maze2D environment with concave obstacles. In Section B.5, we present more quantitative and qualitative evaluation results on the real-world dataset.

B.1 Performance on Base Environment

We provide detailed numerical motion planning results on three base environments in Table 9. The corresponding bar plot visualization is given in Figure 5. Besides, we provide an additional GNN-based motion planning baseline (Yu & Gao, 2021) in the table below. We report the mean and standard error on three main motion planning metrics: success rate, planning time and number of collision checks. Our method consistently outperforms all other baselines on success rate. Especially, in the hardest Dual KUKA environment, our planner outperforms other learning-based baseline by 40-50% on success rate, and even outperforms the advance sampling-based method BIT* by 15% while with one order of magnitude less planning time and collision checks.

	Maze2D			KUKA			Dual KUKA		
Method	Success	Time	Check	Success	Time	Check	Success	Time	Check
RRT*	98.8±0.3	0.95±0.02	10453.4±156.8	59.4±1.9	3.15±0.06	19508.6±278.6	39.5±1.6	4.16±0.04	18560.5±112.6
P-RRT*	98.5±0.3	1.17±0.02	14158.9±225.8	58.0±1.9	3.44±0.06	21471.4±269.1	35.0±1.6	4.54±0.02	18117.7±78.8
BIT*	100.0±0.0	0.21±0.00	2067.4±24.4	96.5±0.7	0.86±0.04	6047.6±364.7	82.1±1.6	2.37±0.08	8925.4±365.0
AMP-LS	86.3±0.9	1.41±0.11	1025.4±30.2	27.2±4.3	3.89±0.35	4176.4±131.0	40.9±3.2	7.38±0.72	12091.3±230.0
MPNet	85.6±0.9	0.41±0.01	6315.7±203.0	76.0±3.9	0.10±0.02	885.4±135.5	42.0±4.1	0.35±0.04	1750.2±195.6
MπNet	97.9±0.3	0.07±0.00	178.7±7.8	88.7±0.8	0.22±0.01	232.5±11.6	63.9±1.4	0.72±0.02	875.2±26.7
GNN	100.0±0.0	0.11±0.00	1043.3±5.5	98.9±0.4	0.38±0.01	2207.4±153.6	94.6±0.6	1.22±0.03	6269.1±344.7
Ours	100.0±0.0	0.12±0.00	97.0±0.3	99.7±0.0	0.13±0.00	75.5±3.1	97.7±0.5	0.27±0.01	122.4±6.1

Table 9: **Quantitative Motion Planning Performance.** Evaluated on 100 unseen environments with 20 motion planning problems in each environment. We report the mean over all environments and the standard error across different environments. The table is the numerical results corresponding to the bar plot visualization in Figure 5.

In addition, we present 4 qualitative comparisons with the state-of-the-art learning-based motion planner MπNet on the KUKA environments in Figure 18 at the end of this section. In Figure 19, we demonstrate the versatile morphology of our motion trajectories on the same motion planning problem.

B.2 Performance on Composite Same Environment

We construct *composite same environments* with more obstacles of the same type than training phase environments. Since more obstacles are presented, these evaluation environments are out of training distribution and hence more challenging. The locations of each obstacle in the evaluation environments are random and unseen to the model. We further provide a baseline *Diffusion* in which we do not compose potentials. The quantitative results are shown in Table 10, 11, 12, and qualitative results are shown in Figure 11. Our composed model demonstrates superior performance over all benchmarks, which is consistent with the expectation that composition enables more effective generalization to more obstacles.

As presented in Section 3.3, when generalizing to more obstacles, we might split the obstacles into several groups before composition. In our experiments, the splitting of obstacles is random, and we do not observe correlations between performance and the ways of splitting.

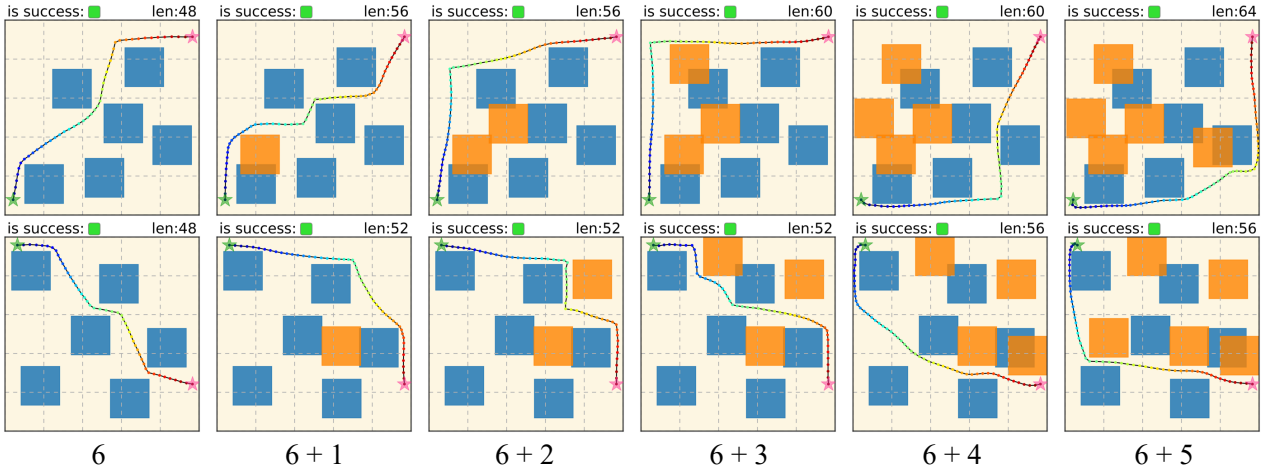


Figure 11: **Compositional Generalization over Increasing Obstacles.** The green/pink star indicates the start/goal state. Our planner is trained with a dataset with only 6 obstacles and can generalize to scenarios with more obstacles by directly composing potentials in test time (without any re-training). The generated trajectories demonstrate various morphology and reach the goals with near optimal paths.

		6 + 1		6 + 2		6 + 3	
	Method	Success	Check	Success	Check	Success	Check
Maze2D	RRT*	98.5±0.3	11467.3±176.8	97.7±0.4	12675.6±238.9	96.7±0.5	14247.8±302.1
	P-RRT*	97.5±0.3	15308.9±257.3	96.8±0.5	17176.1±324.3	92.8±0.7	19280.9±429.9
	BIT*	100.0±0.0	2340.0±35.4	100.0±0.0	2635.7±40.6	100.0±0.0	2942.3±53.7
	AMP-LS	74.1±6.4	2845.2±247.0	66.7±6.2	3357.6±298.0	56.5±7.0	3973.5±290.0
	MπNet	92.6±0.8	349.7±23.3	88.0±1.1	481.2±30.1	81.2±1.5	685.3±45.5
	Diffusion	99.9±0.1	126.1±5.6	99.2±0.2	229.7±11.8	95.4±0.9	454.8±34.2
	Composed	100.0±0.0	102.8±2.6	99.9±0.1	147.9±6.4	99.5±0.3	218.8±15.1

		6 + 4		6 + 5		6 + 6	
	Method	Success	Check	Success	Check	Success	Check
Maze2D	RRT*	93.3±0.6	15729.8±307.4	88.7±1.1	18508.4±449.0	84.7±1.3	18598.5±416.9
	P-RRT*	89.0±0.9	21280.1±398.9	78.2±1.7	24471.1±526.0	76.7±1.5	24491.8±501.8
	BIT*	100.0±0.0	3220.2±59.7	100.0±0.1	3563.4±79.2	100.0±0.1	3580.7±61.1
	AMP-LS	45.1±7.2	4560.4±310.0	37.8±7.7	4928.6±361.0	33.5±7.0	5107.2±341.0
	MπNet	75.6±1.5	860.7±47.3	65.0±1.8	1154.3±50.8	66.2±1.6	1120.9±48.0
	Diffusion	85.2±1.4	769.7±41.4	71.9±1.8	997.0±38.1	64.0±1.8	1080.1±31.9
	Composed	98.8±0.3	308.2±22.1	97.0±0.5	393.9±23.0	97.0±0.5	392.7±25.7

Table 10: **Compositional Generalization over Increased Obstacles in Maze2D.** In the top row, the *left digit* represents the number of obstacles that the model is trained on; the *right digit* denotes the number of extra obstacles added to the evaluation environments. Unlike other learning-based planners, whose performance significantly declines as the number of obstacles increases, our method sustains a near-optimal success rate even when the quantity of obstacles doubles.

		4 + 1		4 + 2		4 + 3	
	Method	Success	Check	Success	Check	Success	Check
KUKA	RRT*	58.1±2.1	18920.7±281.3	56.5±2.0	17964.2±230.7	59.5±2.3	17339.7±261.3
	P-RRT*	55.7±2.1	20356.2±254.2	55.0±2.1	19420.8±216.0	56.8±2.3	18287.8±203.3
	BIT*	93.4±1.1	7045.6±483.9	91.5±1.2	8055.9±505.9	92.0±1.2	7176.8±477.9
	AMP-LS	28.8±2.0	6767.0±154.0	35.0±1.6	7461.5±401.0	37.0±1.7	7277.4±150.0
	MπNet	85.7±1.2	262.6±16.0	82.2±1.4	314.7±19.3	84.0±1.3	301.7±18.6
	Diffusion	92.7±1.5	273.3±45.1	83.9±6.1	425.5±99.6	82.8±5.1	430.8±79.8
	Composed	97.0±0.8	171.5±35.3	92.4±1.5	283.4±43.2	93.4±1.2	259.9±39.4

Table 11: **Compositional Generalization over Increased Obstacles in KUKA.** In the top row, the *left digit* represents the number of obstacles that the model is trained on; the *right digit* denotes the number of additional obstacles. By composing potentials, our planner surpasses all the baselines by a margin.

		5 + 1		5 + 2		5 + 3		5 + 4	
	Method	Success	Check	Success	Check	Success	Check	Success	Check
Dual KUKA	RRT*	36.2±1.9	17843.8±110.8	33.2±1.7	17311.8±93.9	33.9±1.8	16847.9±81.1	30.4±1.7	16355.9±72.6
	P-RRT*	30.4±1.8	16921.2±75.2	24.6±1.4	16029.4±74.3	23.1±1.5	15387.9±69.0	17.3±1.3	14667.8±81.9
	BIT*	76.2±1.9	9470.5±376.1	69.0±2.2	9631.7±284.6	57.9±2.7	9034.9±256.0	44.0±3.0	8290.1±176.1
	AMP-LS	0.1±0.0	17512.0±180.0	0.3±0.1	17107.4±219.0	0.1±0.1	16725.1±233.0	0.3±0.1	16419.1±210.0
	MπNet	60.5±1.4	919.4±27.5	55.1±1.5	1034.9±27.6	51.0±1.7	1106.4±30.3	48.8±1.5	1134.0±27.2
	Diffusion	91.7±1.5	305.6±29.8	80.9±2.2	497.1±35.5	69.0±2.9	624.0±35.8	58.1±3.0	761.0±31.0
	Composed	97.3±0.7	250.4±26.2	95.2±1.1	373.9±37.1	90.8±1.3	545.7±43.7	87.8±1.5	648.4±47.2

Table 12: **Compositional Generalization over Increased Obstacles in Dual KUKA.** In the top row, the *left digit* represents the number of obstacles that the model trained on; the *right digit* denotes the number of additional obstacles. In this most difficult Dual KUKA environment, our planner outperforms other baselines by a larger margin. From 5 + 1 to 5 + 4, the advanced sampling-based method BIT* drops by 32%, with a success rate of 44%, while our method only drops by less than 10%, achieving 87.8% in success rate.

B.3 Performance on Composite Different Environment

We present extra experiment results on composing two different models. Each model is separately trained on one single obstacle type and we evaluate them on more complex environments with multiple obstacle types.

Static 1 + Static 2. In Table 13, we present quantitative generalization results of composing two different static Maze2D models. Static 1 is a model only trained on obstacles of size 1×1 and Static 2 is a model only trained on obstacles of size 1.4×1.4. The composite environment *Static 1 + Static 2* contains six 1×1 obstacles and three 1.4×1.4. As the setting in the evaluation of the base environments, we evaluate on 100 different environments with 20 randomly generated start and goal in each environment. Obstacles in all environments are randomly placed. No extra training is required. We also present qualitative results over increasing obstacles in the composite different environment (Static 1 + Static 2) in Figure 12.

	Static 1 + Static 2	
Method	Success	Check
RRT*	90.3	18495.1
P-RRT*	82.8	24959.9
BIT*	100.0	3486.9
Ours	98.9	304.7

Table 13: **Quantitative Results on Maze2D Composite Different Environments.** Static 1 is a model only trained on obstacles of size 1×1 and Static 2 is a model only trained on obstacles of size 1.4×1.4. The testing environments contain six 1×1 obstacles and three 1.4×1.4 obstacles. Though trained on different environments separately, the composed model reaches near optimal success rate while requiring one order of magnitude less collision checks than BIT*.

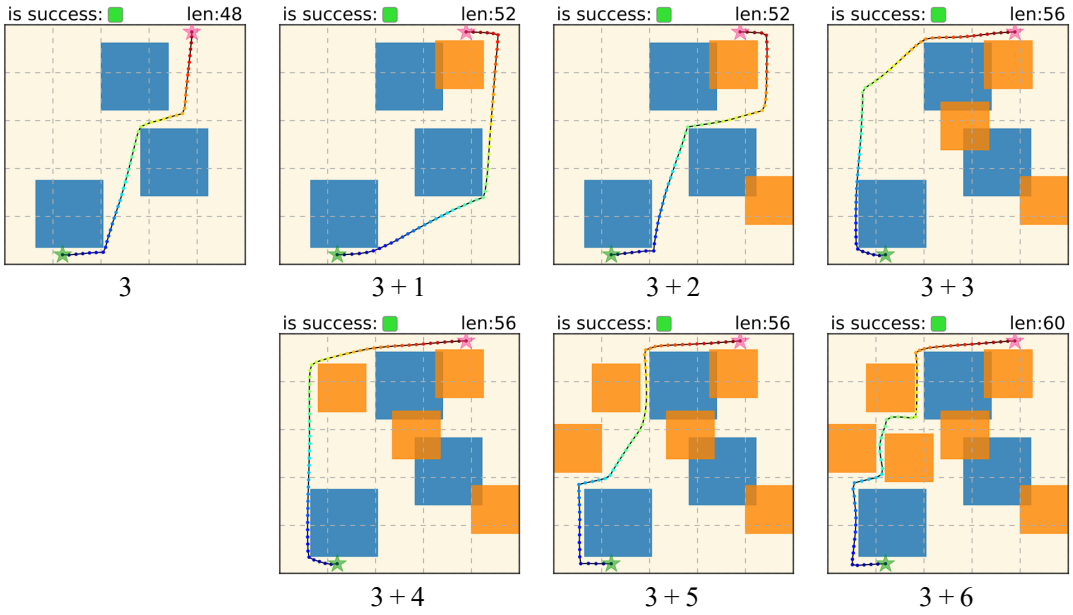


Figure 12: **Compositional Generalization over Different Obstacles.** The green/pink star indicates the start/goal state. Two separately trained models are composed: a model *only* trained on large blocks (blue) of size 1.4×1.4 and a model *only* trained on smaller blocks (orange) 1×1. By composing two models, our planner can generalize to more complex scenarios with various obstacles in test time (without any re-training). The generated trajectories demonstrate various morphology and reach the goal with smooth and short paths.

Static 1 + Concave. In Figure 13, we provide qualitative results on composite environments of square obstacles and concave obstacles. In the leftmost column, we show a generated motion trajectory on a base environment with 6 square obstacles only. Following this, we gradually add concave obstacles to the base environment, from 6 + 1 to 6 + 6. Our planner can plan smooth and coherent trajectories accordingly by composing the corresponding potentials.

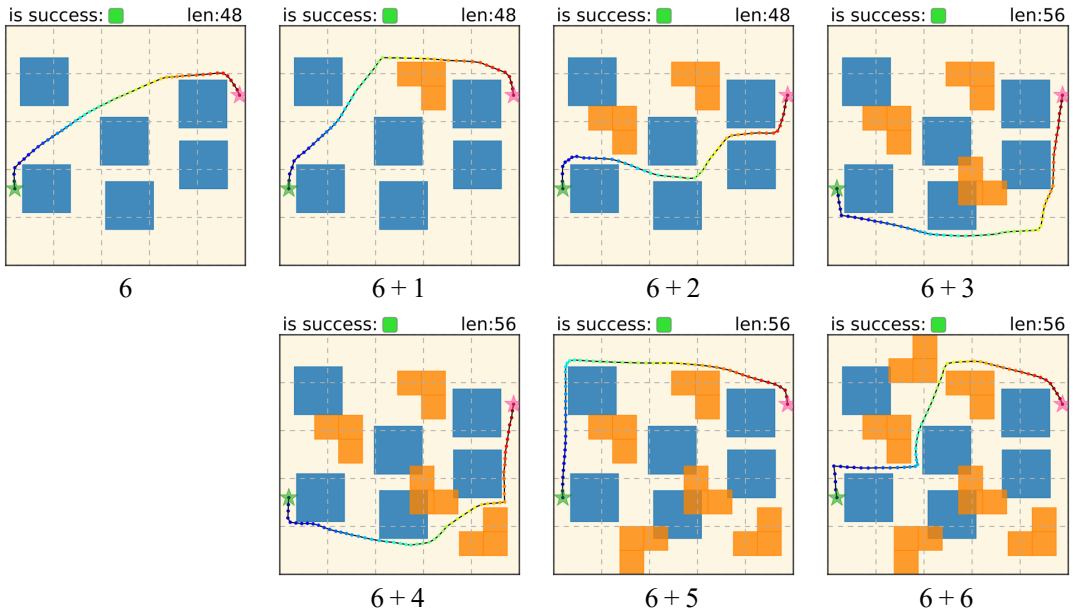


Figure 13: **Compositional Generalization over Square and Concave Obstacles.** The green/pink star indicates the start/goal state. Two separately trained models are composed: a model *only* trained on square blocks (blue) of size 1.0×1.0 and a model *only* trained on concave blocks (orange). Our planner can adaptively propose motion trajectories according to the environments by composing the potentials of square obstacles and concave obstacles.

Static 1 + Dynamic Environment. We evaluate our method on composite different environments which contain both static and dynamic obstacles. Note that our method is only trained on environments that purely consist of static obstacles or dynamic obstacles. *Static 1* is a model that is only trained on six 1×1 static obstacles and *Dynamic* is a model that only trained on one moving 2×2 obstacle. The composite environments contain six 1×1 static obstacles together with one moving 2×2 obstacle. The quantitative results are shown in Table 3 and two qualitative results are shown in Figure 14.

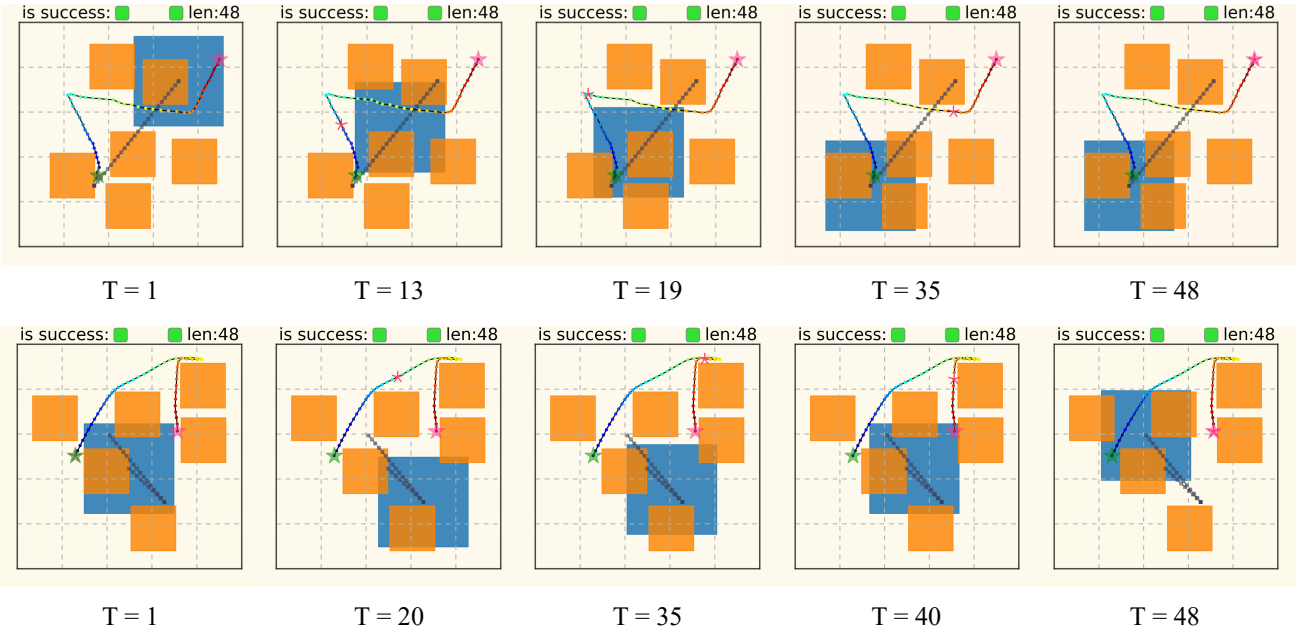


Figure 14: **Qualitative Compositionality Generalization over Static 1 + Dynamic Obstacles.** The current position of the agent is shown in the pink asterisk. The grey line indicates the moving trajectory of the dynamic obstacle (the large blue block). In the first row, the planned trajectory goes further left in order to avoid the moving obstacle from the upper right. At T = 19, the trajectory is traveling right while can still avoid the other orange static obstacles. In the second row, the dynamic obstacle is first moving toward the bottom right corner and then going back to the upper left. At around T = 35, the trajectory takes several extra moves, waiting for the blue block passing through the goal position.

Static 2 + Dynamic Environment. *Static 2* is a model that is only trained on 1.4×1.4 static obstacles and *Dynamic* is a model that only trained on one large moving 2×2 obstacle. The composite environments contain three 1.4×1.4 static obstacles and one moving 2×2 obstacle. The quantitative results are shown in Table 3 and two qualitative results are shown in Figure 15.

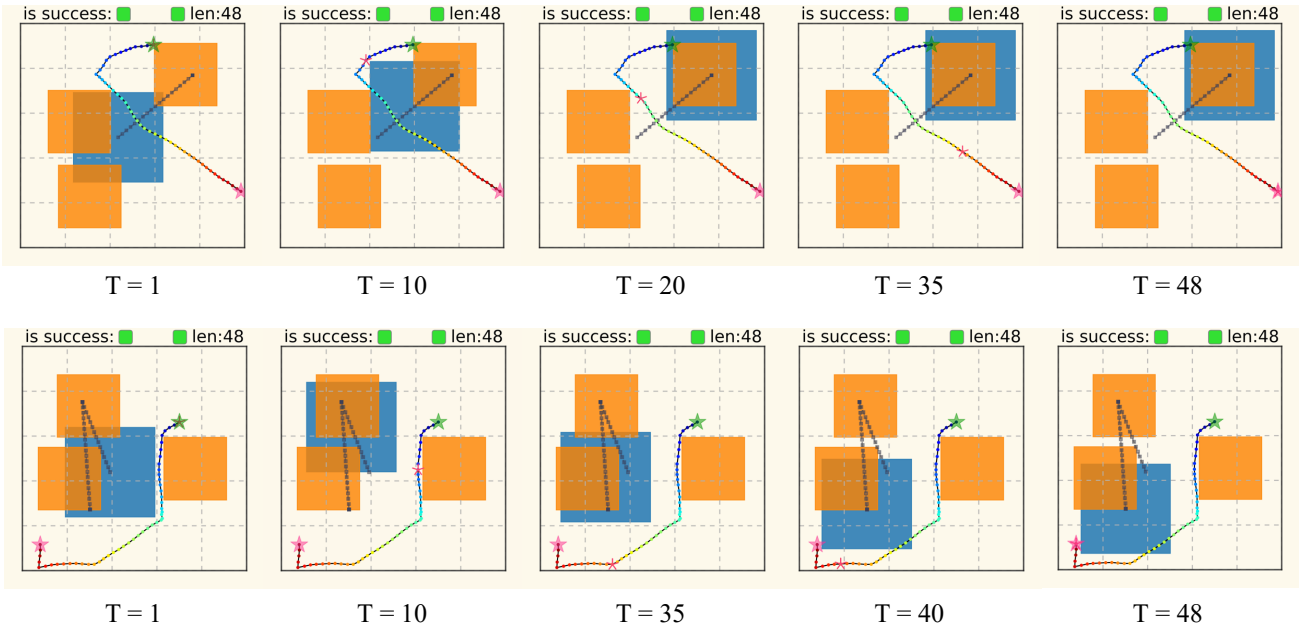


Figure 15: **Qualitative Compositionality Generalization over Static 2 + Dynamic Obstacles.** The current position of the agent is shown in the pink asterisk. The grey line indicates the moving trajectory of the dynamic obstacle (the large blue block). Two motion plans are shown. In the first row, the planned trajectory veers further left to circumvent the approaching moving obstacle while still being aware of the other orange static obstacles. In the second row, the dynamic obstacle first moves upwards and then goes downwards. Both phrases of the obstacle's motion can potential block the agent's path, especially when it is going downwards. The planned trajectory first navigates through the gap between the blue block and the orange block, and then reserves enough space and travel near the bottom of the environment to prevent any collisions.

B.4 Performance on Concave Obstacles

We construct Maze2D environments with 7 concave obstacles and have shown the qualitative and quantitative results in Figure 7 and Table 2 in the main paper.

Specficially, in Table 2, we show the motion planning performance on Convex and Concave obstacles side by side, where RMP (Ratliff et al., 2018) is a traditional potential-based motion planner and MPD is a recent diffusion-based motion planner (Carvalho et al., 2023). Similar to other experiments, all obstacles are randomly placed and no post-training/fine-tuning is required. We can see that all methods subject to a certain decline in environments with concave obstacles. Notably, our learned diffusion potential motion planner can still solve all the problems, surpassing the pure potential-based method by a margin while requiring significantly less planning time than the state-of-the-art sampling-based planners BIT*. In Figure 16, we present more qualitative results of RMP, RRT*, BIT*, MPNet, and our method on environments with concave obstacles.

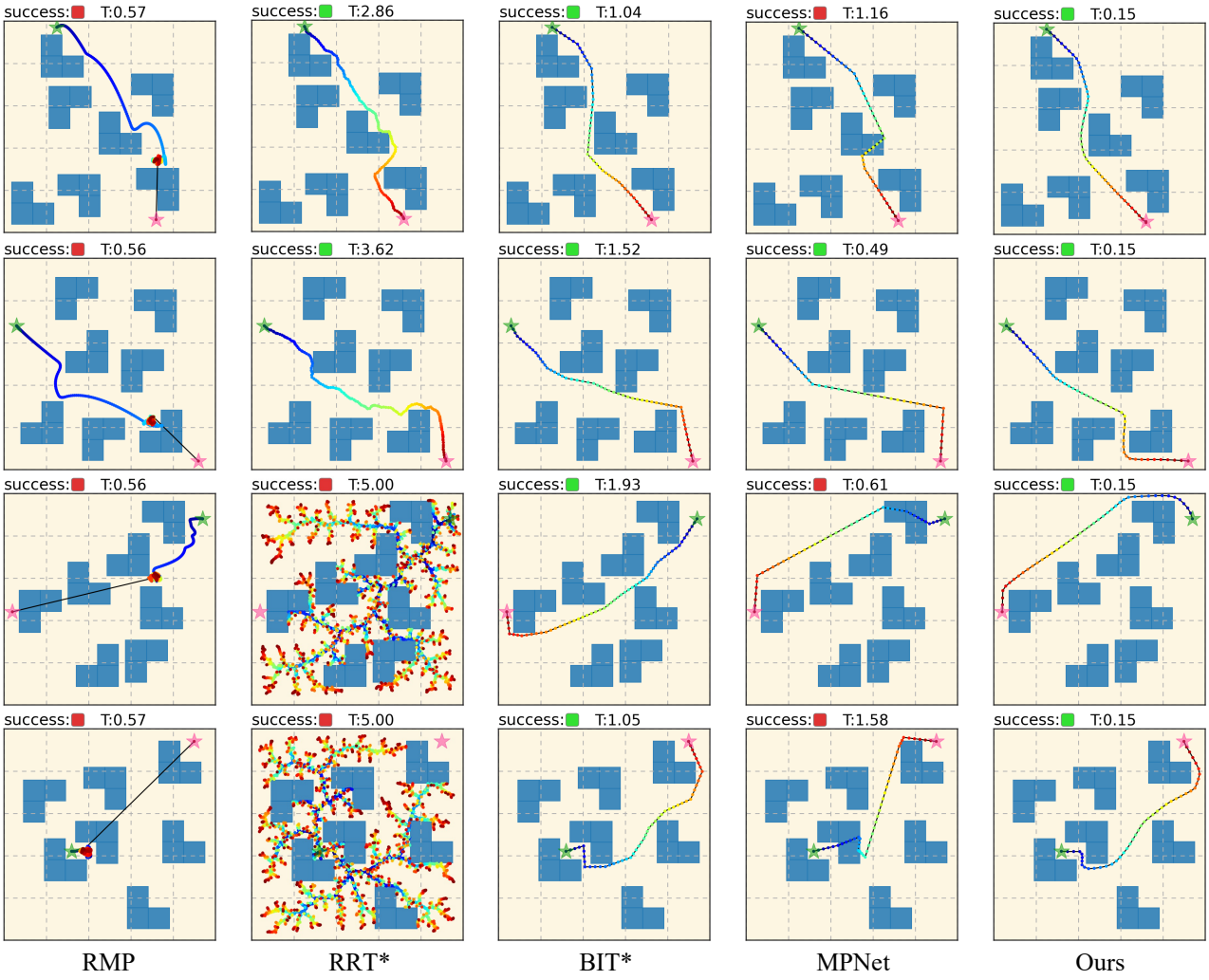


Figure 16: **Qualitative Performance on Environments with Concave Obstacles.** The green star indicates the start pose and the red star indicates the goal pose. RMP tends to stuck in local minima and fails to reach the goals; RRT* is slow in speed and may reach the planning time limit in some difficult problems; trajectories by MPNet may occasionally go through obstacles in the environments; our method is able to generate smooth and low-cost trajectories without collision while being up to 10 times faster than BIT*.

B.5 Performance on Real-World Dataset

Current robotic path planning problem is usually limited to simulation environments and there is no widely-used real-world benchmark. Since pedestrian trajectory is naturally a kind of demonstrations of human planning, we leverage the ETH/UCY Dataset to evaluate real-world motion planning capability. Specifically, the entire dataset is comprised of 6 scenes: *ETH, Hotel, Zara01, Zara02, Students01, Students03*, where the models are trained on 5 scenes and tested on the held-out scene. Similar to our simulation environments, the model takes as input the position of start, goal, and other pedestrians, and outputs a predicted motion plan. The predicted motion plan is desired to be as close as possible to the real human trajectory and thus we report the Average Displacement Error (ADE) as defined in equation 15:

$$\text{ADE} = \frac{\sum_{i \in N} \sum_{t \in T} ||q_t^i - \hat{q}_t^i||_2}{N \times T} \tag{15}$$

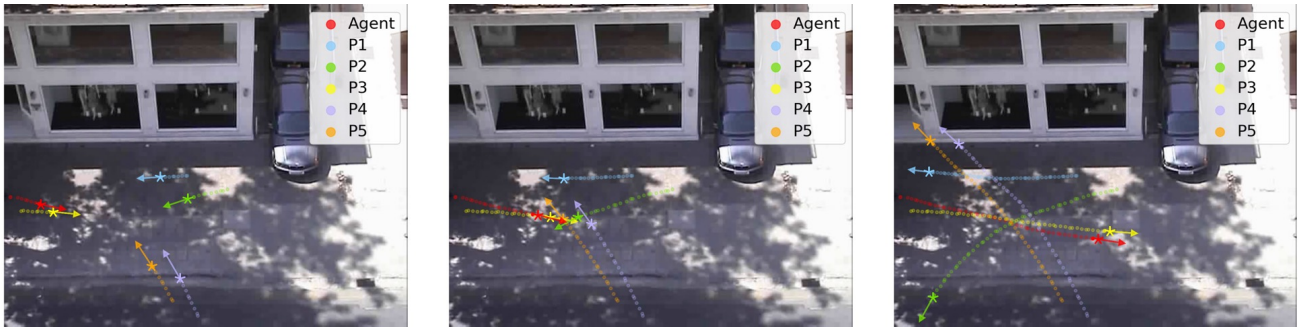
where $\hat{q}_t^i \in \mathbb{R}^2$ is the step t in the ith predicted path and q_t^i is the corresponding ground truth. ADE measure the similarity between the predicted trajectories and human trajectories. A smaller ADE value indicates the predictions are closer to the real human behavior.

	ETH		*Hotel*		*Zara01*		*Zara02*		*Students01*		*Students03*	
Method	ADE ↓	Time	ADE	Time	ADE	Time	ADE	Time	ADE	Time	ADE	Time
MPNet	18.11	0.12	28.60	0.11	11.06	0.12	17.22	0.11	10.37	0.12	8.93	0.11
MπNet	37.70	0.26	44.49	0.29	1.14	0.22	13.66	0.23	12.76	0.18	1.54	0.22
Ours	**0.94**	**0.17**	**5.20**	**0.17**	**0.35**	**0.17**	**0.38**	**0.17**	**0.52**	**0.17**	**0.89**	**0.17**

Table 14: **Quantitative Results on real-world ETH/UCY Dataset.** We adopt the ADE metric to show the similarity between the predicted motion plans and real human motion trajectories on unseen scenarios. All motion plans are in the world coordinate as given in the dataset. Our method can mimic human motion behaviors more precisely in most scenes, while both MPNet and MπNet cause drastic error compared to real human trajectories.

The per-scene quantitative performance is shown in Table 14. Each scene is captured at different time of a day or different location, resulting in different data distribution. The ADE is relatively smaller in *Zara* and *Students* because the model can see a similar counterpart scene at training, e.g., train set includes *Zara01* when tested on *Zara02*. By contrast, *ETH* or *Hotel* are more unique to other scenes (e.g., contain different scene layout and human behavior patterns) and thus causing higher evaluation error. As shown in Table 14, MPNet consistently falls short of the ADE, though with slightly faster speed. We observe that MπNet can produce reasonable motion plans in many cases, but it occasionally predicts random values, which causes significant deviation from the target and leads to the notbaly larger ADE. Our method demonstrates better generalizability and stability by precisely mimicing the human trajectories in most held-out scenes. We also notice that in *Hotel*, all methods suffer from a severe surge in ADE. Our speculation is that the *Hotel* has different walkable world-coordination and in addition, it contains various unseen types of pedestrians motion pattern, such as slowly pacing people that are chatting or waiting, people stepping on or off the train.

Base Real World Motion Planning. We visualize the motion trajectory planned by our model in *Zara02* scene where five other pedestrians are presented in Figure 17. The planner is trained on the other 5 scenes and evaluated on this held-out scene. In the given scenario, our agent (highlighted in red) is heading towards the bottom right corner where P2 (yellow) is on its way. Also, the agent is about to enter an intersection with multiple oncoming pedestrians. In this complex dynamic scene, the trajectory planned by our model first chooses to follow P2 (shown in T = 10) and adjust its pace to let other pedestrians pass through first. As shown in in T = 22, our agent successfully crosses the intersection without interrupting any other pedestrians.



T = 10 T = 22 T = 50

Figure 17: **Qualitative Real World Motion Plan, *Zara02* Scene.** Each star represents a pedestrian on the street. Red indicates the trajectory planned by our model, while other colors represent 5 pedestrians in the surrounding. Our motion plan smoothly passes through the intersection without any collision or discontinuity.

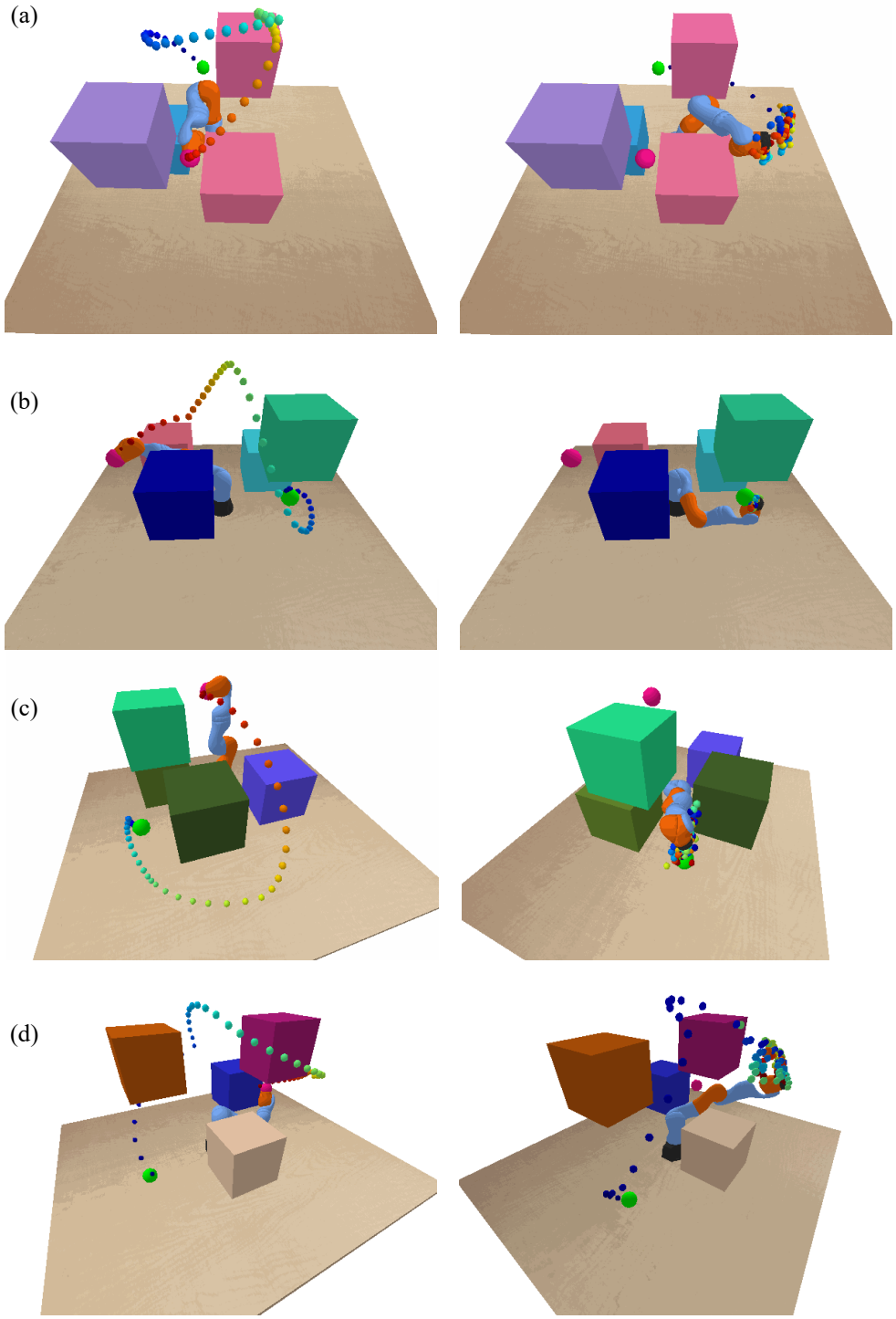
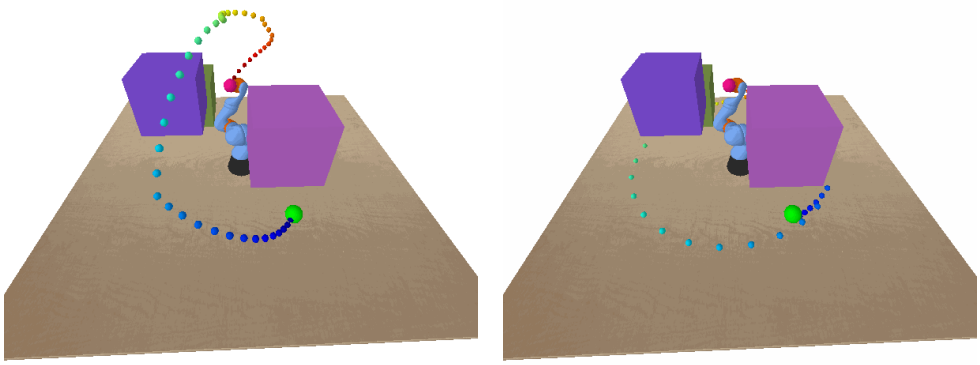
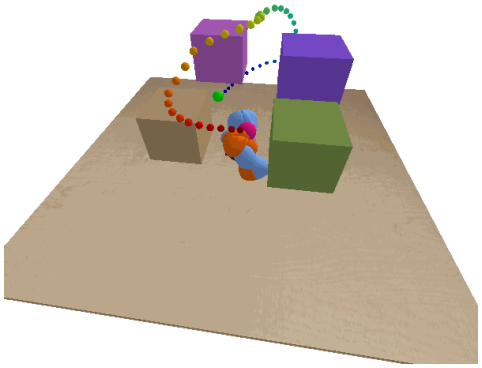


Figure 18: **Comparison with MπNet on KUKA.** Trajectories of 4 motion planning problems. The large green/pink ball indicates the location of the end effector of the start/goal state. Our planner can generate smooth long-horizon motion trajectories and avoid being stuck in local geometry. For example, in (a), our planner is able to select a feasible and shorter path passing through the center of the workspace to reach the goal state; and in (c), the trajectory by our planner navigates through narrow passage ways without collision, e.g. travel between the green and purple blocks.

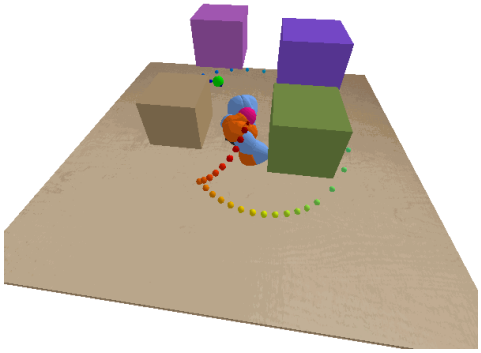


Trajectory 1, View a

Trajectory 2, View a



Trajectory 1, View b



Trajectory 2, View b

Figure 19: **Flexible Trajectory Morphology.** Our method can generate trajectories with various morphological shapes. The large green/pink ball indicates the location of the end effector of the start/goal state. View a and b represent two different viewing angles of the same trajectory. In the figure, Trajectory 1 and Trajectory 2 are generated under the same constraints (e.g., state state, goal state, obstacles locations) but with different route selection. Trajectory 1 routes through the central narrow passage between the two purple blocks and arrives the pink ball from above. Trajectory 2 avoids the obstacles by going under the blocks and arrives the same goal state from below.

B.6 Comparison to Training on Multiple Types and Numbers of Obstacles

In this section, we train an additional baseline which learns all data with different numbers and types of obstacles in a single diffusion potential function. Thus, training this baseline requires additional motion planning demonstrations of various obstacle combinations. Note that for our proposed compositional motion planning method, models are only trained on a limited number or type of obstacles, while can effectively generalize to various combinations through composition.

We use Maze2D environment as the testbed. Let s denote obstacle of type 1 and l denote obstacle of type 2, so that $s6$ indicates 6 obstacles of type 1. We keep all the training and evaluation setups the same as in previous experiments in B.2. In Table 15, *Direct Training* refers to a model directly trained on various numbers/types of obstacles, and *Composed* refers to the model trained on a fixed number of obstacles and generalizes to novel obstacle combinations via composing potentials in test-time. For further comparison, we include another baseline *Process All Obstacles* in the table, which is also trained on a fixed number of obstacles, but is directly passed in different numbers of obstacles at test-time (This baseline is equivalent to *Diffusion* shown in Table 10).

Shown in Table 15, we observe that composing models obtains a comparable performance to the *Direct Training* and substantially outperforms *Process All Obstacles*. Note that the training and evaluating distribution is similar for Direct Training, while very different for *Composed*, for example, the training data in *Direct Training* includes the scenarios with 12 obstacles, while *Composed* is only trained on 6 obstacles, highlighting the effectiveness of composition for generalization.

	$s6$		$s7$		$s8$		$s9$		$s10$	
Method	Success	Check	Success	Check	Success	Check	Success	Check	Success	Check
Process All Obstacles	100.0	97.0	99.9	126.1	99.2	229.7	95.4	454.8	85.2	769.7
Direct Training	100.0	99.5	100.0	86.3	100.0	95.7	100.0	118.4	99.9	140.5
Composed	100.0	97.0	100.0	102.8	99.9	147.9	99.5	218.8	98.8	308.2

	$s11$		$s12$		$s6+l1$		$s6+l2$		$s6+l3$	
Method	Success	Check	Success	Check	Success	Check	Success	Check	Success	Check
Process All Obstacles	71.9	997.0	64.0	1080.1	–		–		–	
Direct Training	98.8	227.4	98.2	264.1	100.0	90.4	99.9	133.8	99.8	172.4
Composed	97.0	393.9	97.0	392.7	99.9	184.2	99.2	304.1	98.9	304.7

Table 15: **Quantitative Comparison with Model Directly Trained on various Types/Number of Obstacles.** Motion planning performance on the Maze2D environments with combinations of different types and numbers of obstacles, e.g., $s6 + l2$ denotes an environment with six obstacles of type 1 and two obstacles of type 2. In columns with –, *Processing All Obstacles* baseline is not applicable as different models are composed. Note that in *Direct Training*, the model is trained on all 10 obstacle combinations shown above; while for *Composed*, we only train a model on environments with six obstacles of type 1 and a model on environments with three obstacles of type 2.

C Proof of Conditional Independence for Composing Potentials

In Section 3.3, we present a way to generalize to multiple unseen out-of-training-distribution constraints by composing corresponding potentials. Specifically, in Equation 6, we assume that constraint C_1 and C_2 are conditional independent. In this section, we will show how our assumption holds in a general case, and thus the compositionality of our planner can be achieved as in (Liu et al., 2022).

Assume that two set of constraints are given, $C_1 = \{o_1, o_2, o_3, o_4\}$ and $C_2 = \{o_3, o_4, o_5, o_6\}$. Let $f_{C_i}(q_{1:T})$ denote a probability density function over trajectories, where positive likelihood is uniformly assigned to the trajectory $q_{1:T}$ if it satisfies the constraint C_i; otherwise, the likelihood is set to 0. Let $\mathcal{J}_{c_i}$ denote the set of trajectories that satisfies constraint C_i. Then, we have

$$f_{C_i}(q_{1:T}) = \begin{cases} \rho_i & \text{if } q_{1:T} \in \mathcal{J}_{c_i} \\ 0 & \text{if } q_{1:T} \notin \mathcal{J}_{c_i} \end{cases} \tag{16}$$

where ρ_i is a small constant. Similarly, we can define $f_{C_1 \cup C_2}$ as the probability density function of trajectories that satisfy both C_1 and C_2. Clearly, given any trajectory $q_{1:T}$, the probability density of $q_{1:T}$ is positive if and only if both $f_{C_1}(q_{1:T})$ and $f_{C_2}(q_{1:T})$ are positive. More specifically, we have

$$f_{C_1 \cup C_2}(q_{1:T}) = \gamma f_{C_1}(q_{1:T}) f_{C_2}(q_{1:T}) \tag{17}$$

where γ is a constant (note the proportionality constant in the previous equation). We can see that the joint probability density function equals to the scaled product of f_{C_1} and f_{C_2}. While the above equation doesn't indicate independence, sampling using the score function for the left side of the equation is the same as sampling from the summed score function for the right side of the equation, because the score function is the gradient of the log probability and is invariant to the constant multiplier. Therefore, the constant γ here will not affect the test-time sampling process and the proposed procedure of composing potentials for generalization can be achieved in theory.