TRAINING-FREE DIFFUSION PLANNING WITH ANALYTICAL LOCAL SCORES

Anonymous authors
Paper under double-blind review

ABSTRACT

Motion planning requires trajectories that are smooth, goal-directed, and collision-free in environments with complex geometric constraints. Recent diffusion-based planners have shown that trajectory generation can be cast as iterative denoising, which has opened the doors to learning-based approaches that can handle multi-modal trajectory distributions and refine entire trajectories. However, such planners require training on large collections of feasible trajectories, rendering them map-specific, and difficult to deploy when high-quality demonstrations are unavailable. This paper introduces a training-free diffusion-based motion planner that replaces learned global trajectory scores with analytical local scores derived from obstacle, smoothness, velocity, and inter-agent feasibility terms. The proposed idea relies on a key observation: the trajectory score can be approximated from local interactions between neighboring waypoints and nearby constraints. This structure exploitation yields a decomposed denoising procedure that retains the optimization structure of classical optimization methods while inheriting the iterative refinement behavior of diffusion models. Experiments on a large collection of complex environments on multi-agent planning tasks show that the proposed analytical score produces smooth and feasible trajectories within very limited computational costs, for example in generating feasible paths for 300 agents in environments with 100 obstacles in under 6 seconds, while avoiding the data requirements of learned diffusion planners.

1 INTRODUCTION

Motion planning is a fundamental problem in robotics and autonomous systems, where the goal is to compute trajectories that move a system from an initial state to a target state while avoiding collisions with obstacles (LaValle, 2006). In multi-agent settings, this problem becomes substantially more challenging: each robot must satisfy its own motion and safety constraints while also avoiding time-dependent collisions with other agents (Sharon et al., 2015; Ma et al., 2018). The spatio-temporal coupling and nonlinearity of these constraints make multi-agent path planning difficult to solve efficiently, especially as the number of agents and obstacles increases (Liang et al., 2025).

Sampling-based planners provide strong completeness properties under suitable assumptions and have been widely used for high-dimensional planning problems. However, they often produce paths in a coarse configuration space that require additional smoothing and can become inefficient in narrow passages or crawded spaces (Kuffner & LaValle, 2000; LaValle, 2006). Optimization-based planners instead formulate motion planning as continuous trajectory optimization, making it possible to incorporate obstacle avoidance, smoothness, and kinematic constraints directly into the objective or feasible set (Kalakrishnan et al., 2011; Mukadam et al., 2018b). However, these methods are often sensitive to initialization and can become expensive in multi-agent settings because collision constraints couple many agents across time. More recently, diffusion models have shown promising results for motion planning by casting trajectory generation as an iterative denoising process that refines noise into trajectories (Liang et al., 2026b; Shaoul et al., 2025). This perspective is attractive because diffusion models refine entire trajectories rather than committing to one waypoint at a time, and can represent multimodal distributions over possible plans (Carvalho et al., 2023). The key challenge, however, is that existing diffusion planners require learning a neural score model over high-dimensional trajectory distributions. This makes them expensive to train, dependent on large collections of feasible demonstrations, and often map- or distribution-specific. Moreover, learned diffusion planners may still fail to enforce safety constraints, a critical limitation for robotic

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

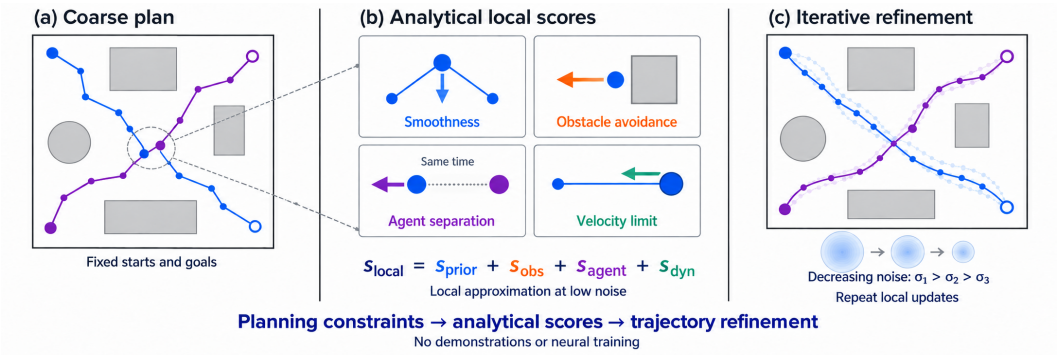


Figure 1: **Training-free diffusion planning.** (a) Coarse trajectories connect fixed starts and goals. (b) Analytical local scores combine (i) smoothness, (ii)obstacle avoidance, (iii) inter-agent separation, and (iv) velocity limits. (c) Repeated updates under decreasing noise refine the trajectories without neural training. [Note: the refined paths cross spatially but pass the intersection at different times].

planning (Liang et al., 2025). Projection-based refinement can improve constraint satisfaction, but it introduces additional computational cost (Liang et al., 2026b; 2025).

To address these challenges, this paper develops a training-free diffusion-based planner with fully analytical local scores. The key idea is based on the observation that trajectory refinement has a strong locality of influence (Pan et al., 2012). Although the exact score of a full trajectory distribution can be globally coupled, the update of an individual waypoint is dominated by local temporal and spatial interactions: neighboring waypoints determine smoothness and velocity consistency, nearby obstacles determine collision avoidance, and nearby agents at the same time determine inter-robot separation (Mukadam et al., 2018a; Zucker et al., 2013), as illustrated in Figure 1(b). Motivated by this structure, we introduce a decomposed local score formulation that constructs an analytical score surrogate directly from the planning constraints, rather than learning a neural approximation to the score of a high-dimensional distribution over complete trajectories. The resulting planner starts from a coarse trajectory initialization with fixed endpoints (Figure 1(a)) and repeatedly applies analytical denoising updates under a decreasing noise schedule (Figure 1(c)). Because each update depends only on local trajectory neighborhoods, nearby obstacles, and nearby agents, the method avoids dataset collection and neural-network training while remaining scalable to large multi-agent scenes.

Contributions. The paper makes the following contributions: **(1)** It introduces a training-free diffusion planning framework that replaces learned global trajectory scores with analytical score surrogates derived from the structure of the motion planning problem. **(2)** It develops a decomposed local score formulation in which each waypoint is refined using smoothness, obstacle-avoidance, velocity-feasibility, and inter-agent collision-avoidance terms. **(3)** It demonstrates across multi-agent path planning benchmarks that the proposed planner achieves high feasibility and computational efficiency, obtaining high success on standard test configurations and scaling to scenes with 300 agents and 100 obstacles in just a few seconds.

2 Related Work

Multi-agent path planning has a long history in robotics, with early methods largely built around sampling-based and optimization-based approaches. *Sampling-based planners*, approximate the feasible configuration space by drawing samples and connecting them through local steering procedures (Kavraki et al., 1996; Kuffner & LaValle, 2000; LaValle, 2006). These can scale to high-dimensional spaces better than exhaustive search methods, however, they often produce piecewise or fragmented paths that require post-processing, and their performance can degrade in narrow passages or settings where continuous kinodynamic constraints must be enforced along the entire trajectory. A complementary line of work formulates motion planning as *trajectory optimization* problems (Augugliaro et al., 2012; Park et al., 2020; Mellinger et al., 2012). This gives these methods a direct mechanism for incorporating feasibility requirements, but the resulting optimization problems become increasingly difficult in multi-agent settings, where collision constraints couple agents across both space and time. To improve computational tractability, several optimization-based planners

replace hard feasibility constraints with differentiable costs or penalty terms (Ratliff et al., 2009; Kalakrishnan et al., 2011; Mukadam et al., 2018b), but they are often sensitive to initialization and can struggle in environments with many obstacles or agents. In contrast, our method formulates planning as an iterative denoising process driven by analytical local scores, avoiding dense state-space exploration while also avoiding repeated global trajectory optimization.

Score-based Generative Models for Motion Planning. Recent advancements in generative models offer have opened new avenues for solving motion planning problems by treating feasible trajectories as samples from a distribution and generating them through iterative denoising (Song et al., 2020). Diffusion planners are attractive because they refine whole trajectories at once and can represent multimodal plans in dense environments (Carvalho et al., 2023; Shaoul et al., 2025; Luo et al., 2024). However, they typically require learning a neural score over high-dimensional trajectory distributions, which depends on large collections of feasible demonstrations and does not guarantee constraint satisfaction. To improve feasibility, recent work has added gradient-based guidance during sampling, control barrier and Lyapunov functions, or projection-based refinement after denoising (Carvalho et al., 2023; Mizuta & Leung, 2024; Shaoul et al., 2025; Luo et al., 2024; Liang et al., 2026b; 2025; Christopher et al., 2024; Liang et al., 2026a). These Both retain learned generative models; TFDP constructs its refinement field directly from smoothness and feasibility factors.

The method proposed in this work, instead, replaces the learned global score with analytical local score components derived directly from smoothness, velocity, obstacle-avoidance, and collision constraints. This yields a training-free planner that preserves the iterative refinement behavior of diffusion models while making denoising updates fast and tied to the geometry of the environment.

3 PRELIMINARIES

Multi-agent path planning. Multi-agent path planning involves computing collision-free trajectories for multiple agents navigating a shared environment from designated start positions to goal states. Let $\mathcal{A} = \{a_1, \dots, a_{N_a}\}$ be a set of N_a robots that move in a bounded, planar workspace $\mathcal{W} \subset \mathbb{R}^2$ containing a set of static obstacles $\mathcal{O} = \{o_1, \dots, o_{N_o}\}$. The *free configuration space* is defined as the set of all points in the workspace that are not occupied by obstacles: $\mathcal{C}_{\text{f}} := \mathcal{W} \setminus \left(\bigcup_{o \in \mathcal{O}} o\right)$. Each robot a_i is modeled as a disk of radius $r_i > 0$ and is characterized at discrete time steps $h \in \{0, \dots, H\}$ by a 2-D position $\pi_i^h = (x_i^h, y_i^h) \in \mathcal{C}_{\text{f}}$. A *trajectory* for robot a_i is the sequence $\boldsymbol{\pi}_i := (\pi_i^0, \pi_i^1, \dots, \pi_i^H)$.

The goal of multi-agent path planning is to compute a set of trajectories $\boldsymbol{\Pi} = \{\boldsymbol{\pi}_1, \boldsymbol{\pi}_2, \dots, \boldsymbol{\pi}_{N_a}\}$ such that each robot starts at its designated start position and reaches its target position while avoiding collisions with obstacles and other agents. For every time index h we require

$$\pi_i^h \in \mathcal{C}_{\text{f}}, \qquad \forall i, \tag{1a}$$

$$\|\pi_i^h - \pi_j^h\|_2 \geq r_i + r_j, \qquad \forall i < j, \tag{1b}$$

$$\|\pi_i^{h+1} - \pi_i^h\|_2 \leq v_{\max}\, \Delta t, \qquad \forall i, h < H, \tag{1c}$$

where Eq. (1a) enforces obstacle avoidance, Eq. (1b) inter-robot separation, and Eq. (1c) a first-order kinematic bound with time step Δt and speed limit $v_{\max}$.

Diffusion models for motion planning. Score-based diffusion models define a forward noising process and a reverse sampling process (Song et al., 2020). In the forward process, samples from a data distribution are gradually perturbed with Gaussian noise. Let π denote a trajectory, and let $p_0(\pi \mid \mathcal{O}, s, g)$ denote the distribution of feasible trajectories conditioned on the obstacle set $\mathcal{O}$, start state s, and goal state g. At noise level σ, the forward process induces a noise-smoothed distribution p_σ by $\pi_\sigma = \pi + \sigma\epsilon$, with $\epsilon \sim \mathcal{N}(0, I)$. The reverse process then generates samples by moving noisy trajectories back toward the feasible trajectory distribution. This reverse process is driven by the score function $\nabla_\pi \log p_\sigma(\pi \mid \mathcal{O}, s, g)$, which points in the direction of increasing probability under the noise-smoothed distribution. In learned diffusion planners, feasible trajectories are treated as data samples and used to train a neural network s_θ to approximate this score function (Carvalho et al., 2023; Shaoul et al., 2025; Liang et al., 2025):

$$s_\theta(\pi; \mathcal{O}, s, g). \approx \nabla_\pi \log p_\sigma(\pi \mid \mathcal{O}, s, g) \tag{2}$$

At sampling time, the learned score is used in Langevin dynamics to iteratively refine an initial noisy trajectory, for example through updates of the form

$$\pi \leftarrow \pi + \epsilon s_\theta(\pi; \mathcal{O}, s, g) + \sqrt{2\epsilon\tau}\,\xi, \qquad \xi \sim \mathcal{N}(0, I),$$

with a decreasing noise schedule, step size ϵ (typically chosen proportional to σ^2), and optional temperature parameter τ (Xu et al., 2018). Thus, the learned score determines the direction in which the full trajectory is refined at each denoising step.

Crucially, the score network s_θ is a function of the *entire noisy trajectory*, since feasibility depends on global trajectory structure, obstacle geometry, and interactions among agents across time. Prior diffusion planners often add guidance terms during sampling to penalize obstacle penetration, inter-agent collisions, or trajectory roughness (Carvalho et al., 2023; Liang et al., 2025; 2026b). However, the underlying model still requires learning a global score over high-dimensional trajectory distributions, which is expensive and depends on access to feasible training trajectories. Next, the paper introduces a method that avoids learning such a global score model altogether.

4 Training-Free Diffusion-based Planning (TFDP)

This section presents *Training-Free Diffusion-based Planning* (TFDP), a method that performs iterative denoising without a learned score model. The key idea is to replace the unknown global score $\nabla_\pi \log p_\sigma(\pi \mid \mathcal{O}, s, g)$ of the data distribution with the score of an explicitly *constructed* surrogate target, whose factors are dictated by the locality of the planning problem itself.

4.1 A Surrogate Target Distribution over Trajectories

Rather than learning to approximate the data distribution of feasible trajectories, TFDP constructs a surrogate that exploits the spatio-temporal structure of the path planning problem. The surrogate is a product of four factors, colored in Figure 1(b), that encode smoothness, obstacle avoidance, inter-agent separation, and kinematic feasibility:

$$p(\pi) \;\propto\; \phi_{\text{prior}}(\pi)\,\chi_{\text{obs}}(\pi)\,\chi_{\text{agent}}(\pi)\,\chi_{\text{dyn}}(\pi), \tag{3}$$

Specifically, the Gaussian factor

$$\phi_{\text{prior}}(\pi) \;=\; \exp\left(-\tfrac{\lambda_{\text{sm}}}{2}\sum_i \sum_{h=1}^{H-1} \left\|\pi_i^{h+1} - 2\pi_i^h + \pi_i^{h-1}\right\|^2\right), \tag{4}$$

penalizes sharp bends through a Gaussian finite-difference smoothness prior, derived from locality of motion and from the desire of producing smooth trajectories. The other three indicator factors are

$$\chi_{\text{obs}}(\pi) = \prod_i \prod_{h=1}^{H} \mathbb{I}\left[\,\pi_i^h \in \mathcal{C}_f\,\right], \tag{5a}$$

$$\chi_{\text{agent}}(\pi) = \prod_{i<j} \prod_{h=1}^{H} \mathbb{I}\left[\,\left\|\pi_i^h - \pi_j^h\right\| \geq r_i + r_j\,\right], \tag{5b}$$

$$\chi_{\text{dyn}}(\pi) = \prod_i \prod_{h=1}^{H-1} \mathbb{I}\left[\,\left\|\pi_i^{h+1} - \pi_i^h\right\| \leq v_{\max}\Delta t\,\right]. \tag{5c}$$

The three indicator factors are ordered to match the feasibility constraints: Eq. (5a) corresponds to obstacle avoidance in Eq. (1a), Eq. (5b) corresponds to inter-robot separation in Eq. (1b), and Eq. (5c) corresponds to the kinematic bound in Eq. (1c). The surrogate (3) deliberately discards information about which feasible trajectories are *likely* under demonstration data, retaining only which trajectories are *feasible and smooth*. This is the price paid for being training-free, but it is also a virtue: the surrogate is entirely determined by the problem geometry and the smoothness weight, so it can be evaluated without any learned component. The next step is to show how to compute its score in closed form.

Then, for each waypoint π_i^h of agent a, we approximate the full score at noise level σ by

$$\begin{aligned}\nabla_{\pi_i^h} \log p_\sigma(\pi \mid \mathcal{O}) \;\approx\; & s_{\text{prior}}(\pi_i^h;\, \pi_i^{h-1}, \pi_i^{h+1}) + s_{\text{obs}}(\pi_i^h;\, \mathcal{O}) \\ & + s_{\text{agent}}(\pi_i^h;\, \pi_{j\neq i}^h) + s_{\text{dyn}}(\pi_i^h;\, \pi_i^{h-1}, \pi_i^{h+1}).\end{aligned} \tag{6}$$

The four terms define the local score (s_{local} in Figure 1(b)) combining smoothness, agent-obstacle collision avoidance, agent-agent collision avoidance, and dynamic feasibility (i.e., velocity constraints). The approximation removes long-range score dependencies while preserving the terms that dominate local feasibility. Note this decomposition also makes the update interpretable as each waypoint is moved according to the direction described by each multi-agent path planning constraint. We describe the explicit form of the score components in Section 4.2.

The local score decomposition approximates the noisy trajectory score and does not imply exact conditional independence between distant waypoints. For Gaussian trajectory priors with local interactions, zeros in the precision matrix encode conditional independencies, and adding isotropic diffusion noise changes the precision structure. At small noise levels, indirect dependencies between distant variables are higher-order effects relative to the direct neighbor interactions. This motivates using short score windows in low-noise denoising regimes and considering larger windows at higher noise levels. We formalize this intuition next, by showing that the conditional independence structure of a multivariate Gaussian is approximately preserved under diffusion noise. This result is of independent interest and may be useful in other applications of diffusion models.

Theorem 1. *Let $x \sim \mathcal{N}(0, \Sigma)$ be a multivariate normal distribution with covariance $\Sigma \in \mathbb{R}^{n\times n}$ and precision matrix $\Omega := \Sigma^{-1}$. Let $\{x_i\}_{i=1}^n$ denote the components, and consider the associated Markov random field (MRF) where an edge (i, j) is present iff $\Omega_{ij} \neq 0$.*

Consider a diffusion process applied to x with noise level t, and let $I(x_i; x_j \mid x_{-\{i,j\}})$ denote the mutual information between components x_i and x_j. Then the rate of increase of mutual information at small t is given by

$$\frac{\mathrm{d}}{\mathrm{d}t} I(x_i; x_j \mid x_{-\{i,j\}}) = t \frac{|\sum_k \Omega_{ik}\Omega_{kj}|^2}{\Omega_{ii}\Omega_{jj}}. \tag{7}$$

Consequently, the leading-order contribution to mutual information between x_i and x_j is governed by length-2 paths in the MRF. Variables separated by longer paths incur dependence only at higher orders in t, and are therefore negligible for sufficiently small t.

The detailed proof is shown in Appendix A.1. Theorem 1 shows that the local independence structures are approximately preserved. This is important for our proposed method because it motivates computing score updates from local trajectory neighborhoods at low noise levels.

4.2 Analytical Score Terms

Now, we provide explicit expressions for $s_{\text{prior,obs,agent,dyn}}$. We first introduce the constraint set:

$$\mathcal{C}_{\text{obs}}(\pi_i^h; \mathcal{O}) := \mathcal{C}_{\text{f}}, \tag{8a}$$

$$\mathcal{C}_{\text{agent}}(\pi_i^h; \pi_{i\neq j}^h) := \left\{ y \in \mathcal{W} : \|y - \pi_j^h\|_2 \geq r_i + r_j,\ \forall j \neq i \right\}, \tag{8b}$$

$$\mathcal{C}_{\text{dyn}}(\pi_i^h; \pi_i^{h+1}, \pi_i^{h-1}) := \left\{ y \in \mathcal{W} : \|y - \pi_i^{h-1}\|_2 \leq v_{\max}\Delta t,\ \|\pi_i^{h+1} - y\|_2 \leq v_{\max}\Delta t \right\}, \tag{8c}$$

where $y \in \mathcal{W}$ denotes a candidate position for the waypoint π_i^h. Using the smoothness factor defined in Equation (4), $s_{\text{prior}} = \nabla \log \phi_{\text{prior}}(\pi)$. For the constraint components, let:

$$\chi^\sigma_{\text{obs}}(\pi_i^h; \mathcal{O}) = K_\sigma * \mathcal{C}_{\text{obs}}(\pi_i^h; \mathcal{O}) \tag{9a}$$

$$\chi^\sigma_{\text{agent}}(\pi_i^h; \pi_{i\neq j}^h) = K_\sigma * \mathcal{C}_{\text{agent}}(\pi_i^h; \pi_{i\neq j}^h) \tag{9b}$$

$$\chi^\sigma_{\text{dyn}}(\pi_i^h; \pi_i^{h+1}, \pi_i^{h-1}) = K_\sigma * \mathcal{C}_{\text{dyn}}(\pi_i^h; \pi_i^{h+1}, \pi_i^{h-1}) \tag{9c}$$

be the constraint indicators convolved with Gaussians; i.e., a closed-form solution to the forward diffusion process of a uniform distribution over the corresponding feasible set, up to a constant. Then we use $s_{\text{obs,agent,dyn}}(\cdot) = \nabla \log \chi^\sigma_{\text{obs,agent,dyn}}(\cdot)$, which pull the sample towards the feasible region with increasing strength as $\sigma \to 0$. Unlike guidance terms, the constraint terms $\nabla \log \chi^\sigma_{\text{obs}}$ are

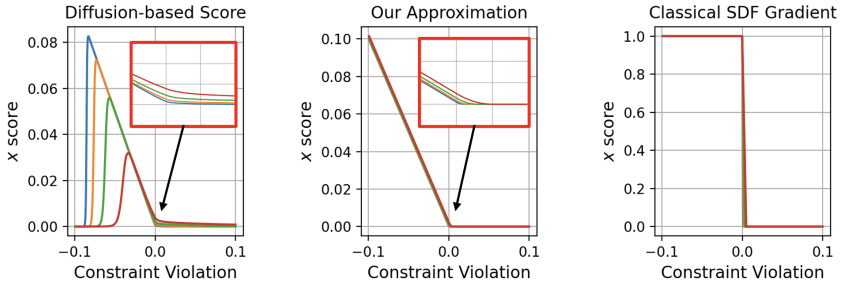


Figure 2: Effective guidance from score candidates

explicitly tied to the probability that a noisy waypoint is feasible. Computing these terms directly requires a convolution integral. For a feasible set $\mathcal{C}_{\text{free}}$ of any arbitrary constraint,

$$\chi^{\sigma}_{\text{constraint}}(\pi_i^h) = \int_{\mathcal{C}_{\text{free}}} \mathcal{N}(\pi_i^h; x, \sigma^2 I)\, \mathrm{d}x \tag{10}$$

We provide closed-form solutions for circular and rectangular feasible sets in Appendix A.2, A.3, and A.4, respectively, however, closed-form solutions are not available for general, non-convex sets, for which we propose a low-noise approximation that avoids computing the convolution integral directly.

Theorem 2 (Low-noise score approximation and projection)**.** *Let $p_\sigma(x)$ denote a Gaussian-smoothed density with noise scale $\sigma > 0$, and let $\Pi(x)$ denote the Euclidean projection of x onto a feasible set $\mathcal{C}_{\text{f}}$. Then, in the limit as $\sigma \to 0$, the score function satisfies the first-order approximation*

$$\sigma^2 \nabla \log p_\sigma(x) \approx \big(\Pi(x) - x\big). \tag{11}$$

Thus, the score is proportional (up to a factor of σ^2) to the negative gradient of the squared distance to the feasible set. In particular, letting $d(x, \mathcal{C}_{\text{f}})$ denote the Euclidean distance to $\mathcal{C}_{\text{f}}$, we have

$$\nabla \left(\tfrac{1}{2} d(x, \mathcal{C}_{\text{f}})^2\right) = x - \Pi(x), \tag{12}$$

which implies that the score direction aligns with the gradient of $\frac{1}{2} d(x, \mathcal{C}_{\text{f}})^2$ up to scaling.

See Appendix A.5 for the proof. Note that, due to the smoothing effect of the Gaussian convolution, the score function is nonzero even for points inside the feasible region, for small distances. Therefore, we also include a quadratically-scaled gradient such that the vertex of the parabola is at distance σ from the obstacle boundary. This motivates a general-purpose constraint satisfaction score function, constraining $\pi_i^h \in \mathcal{C} \to \pi_i^h \in \mathcal{C}_{\text{f}}\, \forall\, i, h$:

$$\psi^{\sigma}(\pi_i^h, \mathcal{C}_f) = \nabla d(\pi_i^h, \mathcal{C}_f) \cdot \begin{cases} d(\pi_i^h, \mathcal{C}_f) + \dfrac{\sigma}{2} & \text{if } d(\pi_i^h, \mathcal{C}_f) > 0, \\ 1/2\sigma \big(d(\pi_i^h, \mathcal{C}_f) + \sigma\big)^2 & \text{if } d(\pi_i^h, \mathcal{C}_f) \in (-\sigma,\, 0], \\ 0 & \text{otherwise.} \end{cases} \tag{13}$$

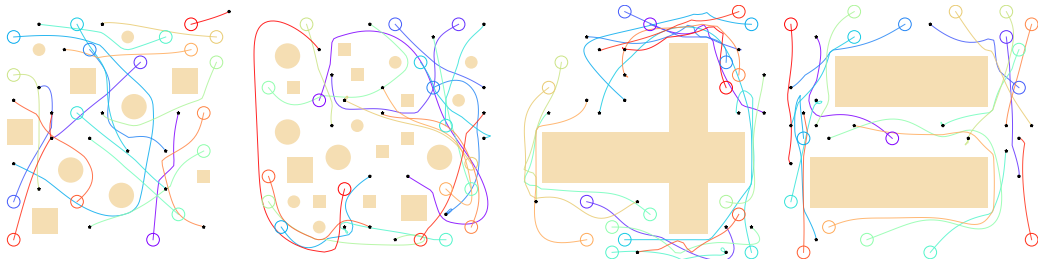
Where $d(\pi_i^h, \mathcal{C}_{\text{f}})$ is a signed distance function (being the negative distance to $\mathcal{C} \setminus \mathcal{C}_{\text{f}}$ if $\pi_i^h \in \mathcal{C}_{\text{f}}$). We instantiate ψ^σ on two feasible sets derived from Eqs. (8b)–(8c): the static obstacle-free region $\mathcal{C}_f^{\text{obs}}(\pi_i^h) := \mathcal{C}_{\text{obs}}(\pi_i^h; \mathcal{O}) = \mathcal{C}_f$, and the agent-separation feasible region $\mathcal{C}_f^{\text{agent}}(\pi_i^h;\, \pi_{j\neq i}^h) := \mathcal{C}_{\text{agent}}(\pi_i^h;\, \pi_{j\neq i}^h)$. The corresponding constraint score terms used in Eq. (6) are

$$\nabla \log \chi^{\sigma}_{\text{obs}}(\pi_i^h) = \tfrac{1}{\sigma^2}\, \psi^{\sigma}\left(\pi_i^h,\, \mathcal{C}_f^{\text{obs}}\right),$$
$$\nabla \log \chi^{\sigma}_{\text{agent}}(\pi_i^h) = \tfrac{1}{\sigma^2}\, \psi^{\sigma}\left(\pi_i^h,\, \mathcal{C}_f^{\text{agent}}\right).$$

Algorithm 1 TFAD planner

1: **Input:** start s, goal g, schedule $\{\sigma_k\}_{k=1}^K$, step size η, temp. τ
2: Initialize π with a coarse roadmap planner
3: **for** $k = 1, \ldots, K$ **do**
4: Sample $\xi_k \sim \mathcal{N}(0, I)$
5: $\pi \leftarrow \pi + \sigma_k^2\, \eta\, \nabla \log p_{\sigma_k}(\pi) + \sqrt{2\, \sigma_k^2\, \eta\, \tau}\ \xi_k$
6: $\pi_i^0 \leftarrow s_i, \quad \pi_i^H \leftarrow g_i \quad \forall\, i$
7: **end for**
8: **Return:** refined trajectory π

The complete procedure is summarized in Algorithm 1. An illustration of the true score function, our approximation, and the gradient given by SDF guidance is shown in Figure 2. Notice that the score pushes agents



(a) Basic (b) Dense (c) Room (d) Shelf

Figure 3: Representative 18-robot trajectories generated by TFDP in four benchmark environments.

out of obstacles in proportion to their penetration depth. It also pushes agents apart only when their uncertainty neighborhoods overlap or approach collision. This gives a continuous analogue of conflict resolution in MAPF, but without constructing a discrete high-level search tree. We encode velocity constraints through a soft penalty:

$$U_{\mathrm{dyn}}(\pi) = \lambda_{\mathrm{vel}} \sum_{i} \sum_{h=0}^{H-1} \phi\left(\|\pi_i^h - \pi_i^{h+1}\| - v_{\max}\right),$$

where $\phi(r) = \max\{0, r\}^2$ or a smooth approximation of the squared hinge. This discourages pairwise step displacements that exceed the velocity limit. We then set $s_{\mathrm{dyn}}(\pi_i^h) = -\nabla_{\pi_i^h} U_{\mathrm{dyn}}(\pi)$.

5 Experiments

Settings. To evaluate the performance of various multi-agent path planning algorithms, we design two experimental settings targeting different aspects of the problem:
1. Feasibility, Efficiency, Quality: We adopt a recent multi-agent path planning benchmark (Liang et al., 2026b) which includes four representative map types: random maps with varying obstacle densities (basic and dense), and structured maps mimicking real-world environments (room and shelf layouts). For each map type, we consider three robot counts (6, 12, and 18), and evaluate across 25 instances with different configurations. Figure 3 illustrates the benchmark maps and representative 18-robot trajectories found by our method.
2. Scalability: To test scalability, we use a large-scale map containing over 100 heterogeneous obstacles. This setup is challenging and supports scenarios with up to 300 robots, largely beyond what existing diffusion methods handle.

Evaluation Metrics. Different algorithms are empirically evaluated based on Success rate, Running time, and Path Length. The *Success rate* indicates the proportion of test cases solved without collisions and within the time limit (900 seconds), *Running time* measures the computational efficiency required to generate a solution, *Path Length* evaluates the quality of generated trajectories.

Competing Methods. Our proposed **TFDP** is compared against diffusion-based and classical baseline methods: (1) **Multi-robot motion planning diffusion (MMD)**: A method that integrates diffusion models with a classical search-based MAPF algorithm to generate multi-agent trajectories under soft collision constraints (Shaoul et al., 2025). (2) **Simultaneous multi-robot motion planning diffusion (SMD)**: It integrates diffusion models with constrained optimization to generate collision-free trajectories for multi-agent path planning (Liang et al., 2025). (3) **Discrete-guided diffusion (DGD)**: This approach decomposes the original multi-agent path planning problem and leverages MAPF-derived spatiotemporal dependencies to guide diffusion models in generating trajectories that satisfy hard feasibility constraints. (Liang et al., 2026b). (4) **Model Predictive Path Integral (MPPI)**: A classical sampling-based method whose updates admit a score-ascent interpretation (Xue et al., 2024). This connection motivates its use as a baseline: MPPI estimates update directions from sampled perturbations, while TFDP uses analytical approximations to local scores. For MMD, SMD, and DGD, we follow the hyperparameter settings reported in their original papers. For MPPI, we treat the MPPI σ the same as diffusion σ and use the our approach's annealed energy landscapes as the optimization objective. For our method, we use 250 Langevin steps, with an exponential noise schedule decreasing from $\sigma = 0.1$ to $\sigma = 0.0001$, and an exponentially annealed temperature from

Map	Robots	TFDP			DGD			MMD			SMD			MPPI		
		S	T	P	S	T	P	S	T	P	S	T	P	S	T	P
Basic	6	**100**	**0.18**	**1.07**	**100**	12.2	1.21	**100**	27.1	1.12	**100**	254.3	1.10	**100**	0.63	1.13
	12	**100**	**0.29**	**1.08**	96	25.0	1.28	**100**	56.0	1.12	-	-	-	**100**	0.72	1.16
	18	**100**	**0.45**	**1.08**	96	65.7	1.33	96	86.3	1.13	-	-	-	99	0.81	1.19
Dense	6	**100**	**0.35**	1.26	**100**	12.3	1.26	40	36.5	1.15	**100**	287.3	**1.13**	99	0.85	1.19
	12	**100**	**0.66**	1.29	**100**	32.9	1.35	8	62.5	**1.15**	-	-	-	99	1.09	1.46
	18	**96**	**1.06**	1.32	76	47.8	1.37	8	87.1	**1.18**	-	-	-	94	1.40	1.59
Room	6	**100**	**0.43**	1.41	**100**	38.9	1.44	24	55.5	**1.16**	**100**	181.0	1.19	**100**	0.93	1.57
	12	**100**	**0.78**	1.45	**100**	84.0	1.50	4	122.6	**1.14**	-	-	-	**100**	1.31	1.69
	18	**100**	**1.24**	**1.45**	**100**	219.0	1.55	0	-	-	-	-	-	99	1.66	1.77
Shelf	6	**100**	**0.27**	1.23	**100**	64.6	1.30	32	48.5	1.18	**100**	274.3	**1.16**	**100**	0.74	1.46
	12	**100**	**0.51**	1.25	**100**	131.5	1.33	4	95.1	**1.22**	N/A	N/A	N/A	**100**	1.00	1.54
	18	**100**	**0.84**	**1.26**	84	324.8	1.38	0	N/A	N/A	N/A	N/A	N/A	98	1.24	1.62

Table 1: Success rate (S↑, in %), running time (T↓, in seconds), and path length (P↓). Arrows indicate whether higher (↑) or lower (↓) values are better. '-' indicates that the method did not complete any instances within the time limit.

$\tau = 0.3$ to $\tau = 0.0001$. We also exponentially decrease the weight of the kinetic-energy term from 0.5 to 0.001 during sampling. The step size is $\eta = 0.5$.

5.1 Comparison across Methods

Table 1 presents the performance of TFDP and baseline methods across the four benchmark maps. The results show that TFDP consistently achieves a 100% success rate in all but one configuration, matching or exceeding the success rates of the diffusion baselines. In particular, on Dense with 18 robots, TFDP achieves 96% success, exceeding DGD's 76% and MMD's 8%. Compared with DGD, the strongest diffusion baseline in these experiments, TFDP matches success in 8 configurations and improves it in the remaining 4. MMD performs well on Basic but its success rate drops sharply on Dense, Room, and Shelf. SMD reaches 100% success with 6 robots, but its computational cost limits evaluation to this setting. Appendix A.14 reports 95% Wilson intervals for TFDP success and ± 2 standard errors for runtime and path length (Table 21).

In terms of running time, TFDP is significantly faster than all other diffusion-based methods. Across the 12 configurations, TFDP requires 0.18–1.24 seconds, averaging 0.59 seconds, whereas DGD requires 12.2–324.8 seconds, averaging 88.23 seconds. The ratio of mean runtimes is approximately $150\times$, with TFDP also achieving higher average success. The difference is especially pronounced on Shelf with 18 robots: TFDP requires 0.84 seconds, versus 324.8 seconds for DGD at 84% success. MMD and SMD are also substantially slower and do not match TFDP's combination of feasibility and scalability. MPPI remains slower under the same annealing schedule, *even when run on a GPU*. Running MPPI on the same CPU as TFDP is slower still, as reported in Appendix A.9 (Table 14).

Finally, TFDP matches or improves DGD's path length in all 12 configurations and produces strictly shorter paths in 11. On Room and Shelf, these shorter paths accompany comparable feasibility and substantially lower runtime. MMD and SMD sometimes produce shorter paths, but path length must be interpreted jointly with success and evaluation coverage: MMD often fails on cluttered or structured maps, while SMD is evaluated only with 6 robots. TFDP also tends to produce shorter paths than MPPI; the latter's noisier update is a possible explanation, although this mechanism is not isolated here. Representative 18-robot trajectories appear in Figure 3.

5.2 Ablation Studies and Scalability

To assess the contribution of the analytical scores, we replace them with CHOMP-style SDF penalty guidance while keeping TFDP's denoising schedule (Zucker et al., 2013; Carvalho et al., 2023; Shaoul et al., 2025). We vary the Langevin step size $\eta \in \{0.1, 0.2, 0.5\}$ and SDF scaling factor $\alpha \in \{0.05, 0.1\}$. Note that, on Dense with 18 robots, TFDP reaches 96% success versus 84% for the best displayed SDF setting (see Table 2), thus supporting an advantage especially in obstacle-dense settings. Appendix A.6 reports a broader sweep of both methods with Wilson intervals while Figure 7 in Appendix A.16 compares their refinement processes.

Next, recall that the analytical scores are local in time, depending only on a short temporal neighborhood of each waypoint. This locality is important for scalability as it allows TFDP to refine each agent's trajectory independently, without constructing a joint high-dimensional search space. We assess locality by training a neural score estimator with different temporal receptive fields. We notice that, at low noise levels, prediction error plateaus as the window includes more distant timesteps, thus supporting the use of short temporal neighborhoods for refinement. For example, at $\sigma = 0.05$, increasing the window from 1 to 3 reduces mean squared error from 0.8679 to 0.3849, while extending it to 7 yields 0.2988. Appendix A.13 reports the full results. Note that we use this estimator used only to test locality; TFDP remains training-free.

Map	Robots	SDF guidance (η, α)						TFDP
		$\eta = .1$		$\eta = .2$		$\eta = .5$		$\eta = .5$
		.05	.10	.05	.10	.05	.10	–
Basic	6	**100**	**100**	**100**	**100**	**100**	**100**	**100**
	12	**100**	92	**100**	**100**	96	88	**100**
	18	76	80	92	96	92	92	**100**
Dense	6	92	92	**100**	96	**100**	96	**100**
	12	84	84	96	92	84	88	**99**
	18	72	52	76	76	84	80	**96**
Room	6	92	96	**100**	**100**	**100**	**100**	**100**
	12	64	76	80	96	96	96	**99**
	18	44	20	76	64	84	84	**98**
Shelf	6	92	96	**100**	**100**	**100**	**100**	**100**
	12	76	76	92	88	**100**	**100**	**100**
	18	56	52	92	76	88	92	**100**

Table 2: Success rates (%) for SDF guidance and TFDP.

Next, we ablate the coarse planner used to initialize TFDP. We compare RRT initialization with Voronoi roadmaps, straight-line interpolation, and Brownian bridges between starts and goals. On Room and Shelf, straight-line and Brownian-bridge initialization achieves only 0–17% success, versus 99–100% for Voronoi and RRT. Thus, initialization is critical in these structured layouts, but the ones adopted are also extremely easy and cheap to compute. Appendix A.8 reports all four initializations for both TFDP and MPPI.

Finally, we compare the default noise schedule with noiseless, deterministic annealed refinement. Removing noise matches or exceeds noisy success on the standard maps, whereas noise improves success point estimates by 5–7 percentage points in the crowded 8- and 12-agent Bottleneck and Standoff cases (Appendix A.10). Thus, among the tested settings, noise is most useful in the more constrained scenarios. It can also improve a weak initialization: with straight-line trajectories replacing RRT, suitable choices of temperature T and smoothing scale σ can improve success. Appendix A.11 reports this ablation on Basic and Dense, showing that the benefit depends on the sampling parameters.

Scalability. Next, we test TFDP on a large-scale map with 100 obstacles and up to 300 robots. The tested map is a 100×100 unit square with randomly placed obstacles of varying sizes. We generate random start and goal positions for each robot, ensuring that they are not in collision with obstacles or each other. Importantly, TFDP produces valid plans for such maps in 5.39 seconds, on average on an A100 GPU. None of the diffusion baselines tested were able to handle this scale, and MPPI fails to find feasible plans for more than 50 robots. Figure 6 in Appendix A.15 shows a generated plan.

We furthertest two *deadlock scenarios* in which robots must pass through a wall opening only wide enough for one robot (see Appendix A.12, Figure 4). Here, TFDP achieves much higher success than MPPI across the tested robot counts, indicating an advantage when passage through a shared bottleneck requires coordination.

6 Conclusion

This paper presented a training-free analytical diffusion planner for path finding and multi-agent path planning. The method replaces learned full-trajectory scores with decomposed analytical scores derived from smoothness, obstacle, velocity, and inter-agent feasibility terms. The analysis and experiments show that local temporal neighborhoods capture most of the useful score information needed for denoising, enabling efficient trajectory refinement without demonstrations or map-specific training. Future work will extend the method to richer kinodynamic models, more challenging maze-like environments, and large-scale multi-agent benchmarks.

Note that, although this is standard to test multi-agent planning on 2D environments, it is interesting to consider extensions to higher dimensions, which we believe will be a natural next step. We also assume a centralized planner and perfect plan executions, and so it is interesting to consider how approaches involving diffusion-like sampling processes could be extended to decentralized planning problems.

AI USE STATEMENT

Generative AI assisted with the creation of Figure 1, its caption, and the associated manuscript edits. The figure is a schematic illustration of the method. The authors also used AI to proofread the manuscript and suggest edits for clarity and conciseness. The authors verified all AI-generated content for accuracy and appropriateness, and they take full responsibility for the final content of the paper.

ETHICS STATEMENT

The research presented in this paper is focused on improving multi-agent path planning algorithms. The methods and experiments are designed to advance the state of the art in robotics and autonomous systems, with potential applications in areas such as warehouse automation, autonomous vehicles, and robotic swarms. The research is conducted with a commitment to transparency, reproducibility, and responsible use of technology.

REPRODUCIBILITY STATEMENT

The paper describes the problem formulation, analytical score construction, planning algorithm, evaluation metrics, and sampling hyperparameters. The appendix provides supporting mathematical derivations, additional benchmark descriptions, hyperparameter sweeps, initialization and noise ablations, and confidence intervals for the reported results. An anonymized implementation will be provided with the supplementary materials, and the complete source code will be released publicly upon publication.

REFERENCES

Federico Augugliaro, Angela P. Schoellig, and Raffaello D'Andrea. Generation of collision-free trajectories for a quadrocopter fleet: A sequential convex programming approach. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 1917–1922, 2012.

João Carvalho, An T. Le, Mark Baierl, Dorothea Koert, and Jan Peters. Motion planning diffusion: Learning and planning of robot motions with diffusion models. In *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 1916–1923. IEEE, 2023. URL https://arxiv.org/abs/2308.01557.

Jacob K Christopher, Stephen Baek, and Ferdinando Fioretto. Constrained synthesis with projected diffusion models. In *Advances in Neural Information Processing Systems*, volume 37. Curran Associates, Inc., 2024.

Mrinal Kalakrishnan, Sachin Chitta, Evangelos Theodorou, Peter Pastor, and Stefan Schaal. STOMP: Stochastic trajectory optimization for motion planning. In *2011 IEEE International Conference on Robotics and Automation*, pp. 4569–4574, May 2011. doi: 10.1109/ICRA.2011.5980280. URL https://ieeexplore.ieee.org/document/5980280. ISSN: 1050-4729.

Lydia E. Kavraki, Petr Švestka, Jean-Claude Latombe, and Mark H. Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Transactions on Robotics and Automation*, 12(4):566–580, 1996. doi: 10.1109/70.508439.

James J. Kuffner and Steven M. LaValle. RRT-connect: An efficient approach to single-query path planning. In *Proceedings 2000 IEEE International Conference on Robotics and Automation*, volume 2, pp. 995–1001. IEEE, 2000. doi: 10.1109/ROBOT.2000.844730.

Steven M. LaValle. *Planning Algorithms*. Cambridge University Press, May 2006. ISBN 978-1-139-45517-6.

Jinhao Liang, Jacob K. Christopher, Sven Koenig, and Ferdinando Fioretto. Simultaneous multi-robot motion planning with projected diffusion models. In *Proceedings of the International Conference on Machine Learning*, 2025. URL https://proceedings.mlr.press/v267/liang25e.html. The draft key uses 2024, but the published version appears in ICML 2025.

Jinhao Liang, Sven Koenig, and Ferdinando Fioretto. Simulation-informed diffusion for decentralized multi-robot motion planning. abs/2605.27697, 2026a.

Jinhao Liang, Sven Koenig, and Ferdinando Fioretto. Discrete-guided diffusion for scalable and safe multi-robot motion planning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2026b. URL https://arxiv.org/abs/2508.20095. The draft key uses 2025, corresponding to the arXiv posting; the venue version appears to be AAAI 2026.

Yunhao Luo, Chen Sun, Joshua B Tenenbaum, and Yilun Du. Potential based diffusion motion planning. *arXiv preprint arXiv:2407.06169*, 2024.

Hang Ma, Daniel Harabor, Peter J. Stuckey, Jiaoyang Li, and Sven Koenig. Searching with Consistent Prioritization for Multi-Agent Path Finding, December 2018. URL http://arxiv.org/abs/1812.06356. arXiv:1812.06356 [cs].

Daniel Mellinger, Alex Kushleyev, and Vijay Kumar. Mixed-integer quadratic program trajectory generation for heterogeneous quadrotor teams. In *2012 IEEE International Conference on Robotics and Automation*, pp. 477–483, 2012.

Kazuki Mizuta and Karen Leung. CoBL-Diffusion: Diffusion-based conditional robot planning in dynamic environments using control barrier and lyapunov functions, 2024. URL https://arxiv.org/abs/2406.05309.

Mustafa Mukadam, Jing Dong, Xinyan Yan, Frank Dellaert, and Byron Boots. Continuous-time gaussian process motion planning via probabilistic inference. *The International Journal of Robotics Research*, 37(11):1319–1340, 2018a.

Mustafa Mukadam, Jing Dong, Xinyan Yan, Frank Dellaert, and Byron Boots. Continuous-Time Gaussian Process Motion Planning via Probabilistic Inference. *The International Journal of Robotics Research*, 37(11):1319–1340, September 2018b. ISSN 0278-3649, 1741-3176. doi: 10.1177/0278364918790369. URL http://arxiv.org/abs/1707.07383. arXiv:1707.07383 [cs].

Jia Pan, Liangjun Zhang, Dinesh Manocha, and UC Hill. Collision-free and curvature-continuous path smoothing in cluttered environments. *Robotics: science and systems VII*, 17:233, 2012.

Jungwon Park, Junha Kim, Inkyu Jang, and H. Jin Kim. Efficient multi-agent trajectory planning with feasibility guarantee using relative bernstein polynomial. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 434–440, 2020.

Nathan Ratliff, Matt Zucker, J. Andrew Bagnell, and Siddhartha Srinivasa. CHOMP: Gradient optimization techniques for efficient motion planning. In *2009 IEEE International Conference on Robotics and Automation*, pp. 489–494, May 2009. doi: 10.1109/ROBOT.2009.5152817. URL https://ieeexplore.ieee.org/document/5152817. ISSN: 1050-4729.

Yilun Shaoul, Itamar Mishani, Maxim Likhachev, et al. Multi-robot motion planning with diffusion models. In *Proceedings of the International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=AUCYptvAf3.

Guni Sharon, Roni Stern, Ariel Felner, and Nathan R. Sturtevant. Conflict-based search for optimal multi-agent pathfinding. *Artificial Intelligence*, 219:40–66, 2015. ISSN 0004-3702. doi: https://doi.org/10.1016/j.artint.2014.11.006. URL https://www.sciencedirect.com/science/article/pii/S0004370214001386.

Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*, 2020.

Pan Xu, Jinghui Chen, Difan Zou, and Quanquan Gu. Global Convergence of Langevin Dynamics Based Algorithms for Nonconvex Optimization. In *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. URL https://proceedings.neurips.cc/paper/2018/hash/9c19a2aa1d84e04b0bd4bc888792bd1e-Abstract.html.

Haoru Xue, Chaoyi Pan, Zeji Yi, Guannan Qu, and Guanya Shi. Full-order sampling-based mpc for torque-level locomotion control via diffusion-style annealing, 2024. URL https://arxiv.org/abs/2409.15610.

Matt Zucker, Nathan Ratliff, Anca D. Dragan, Mihail Pivtoraiko, Matthew Klingensmith, Christopher M. Dellin, J. Andrew Bagnell, and Siddhartha S. Srinivasa. CHOMP: Covariant Hamiltonian optimization for motion planning. *The International Journal of Robotics Research*, 32 (9-10):1164–1193, August 2013. ISSN 0278-3649. doi: 10.1177/0278364913488805. URL https://doi.org/10.1177/0278364913488805.

A TECHNICAL APPENDICES AND SUPPLEMENTARY MATERIAL

A.1 CONDITIONAL INFORMATION FLOW

We start with the definition of *mutual information* between a pair of random variables:

Particularly, for a bivariate normal distribution $p(x_i, x_j) = \mathcal{N}(x_i, x_j; 0, \Sigma)$,

$$I(x_i; x_j) = -\frac{1}{2} \log \left(1 - \frac{\Sigma_{ij}^2}{\Sigma_{ii}\Sigma_{jj}}\right).$$

Noting that $p(x_i, x_j \mid x_{k\neq i,j})$ has a precision matrix equal to a *submatrix* of the original for all $x_{k\neq i,j}$, we can read off the mutual information from the precision matrix of the overall distribution. The conditional mutual information, in terms of the precision matrix, is

$$I(x_i; x_j \mid x_{-\{i,j\}}) = -\frac{1}{2} \log \left(1 - \frac{(\Sigma_{ij}^{-1})^2}{\Sigma_{jj}^{-1}\Sigma_{ii}^{-1}}\right).$$

If there is no connection between the two variables in the MRF (i.e., $(\Sigma^{-1})_{ij} = 0$), the variables are independent conditioned on all other variables. Starting from a precision matrix Σ^{-1}, with nonzero entries representing adjacent random variables in an MRF, one can determine the covariance of the resulting distribution after a forward diffusion process as $\Sigma_t = \Sigma + tI$. In terms of the precision matrix, this is $\Omega_t = (\Sigma + tI)^{-1}$. For very small t, $\Omega_t \approx \Omega - \Omega(tI)\Omega$, and

$$\frac{d}{dt}\Omega_t\bigg|_{t=0} = -\Omega^2$$

Thus, if we have a Gaussian distribution given by Ω, the change after an infinitesimal amount of diffusion time is $-\Omega^2$. When Ω is treated as an adjacency matrix, Ω^2 represents the adjacency after two "hops". Thus, the net effect is that codependence between random variables spreads out across the MRF.

A.2 ANALYTICAL SCORE FUNCTION DERIVATION

For general energy landscapes $\log p(x)$, we note that the "score" of the landscape convolved with $\mathcal{N}(0, \sigma^2 I)$ can be computed as the quotient of two integrals:

$$\nabla_x \log p_\sigma(x) = \frac{1}{\sigma^2} \frac{\int (\tilde{x} - x)\mathcal{N}(\tilde{x}; 0, \sigma^2 I)p(\tilde{x})\,\mathrm{d}\tilde{x}}{\int \mathcal{N}(\tilde{x}; 0, \sigma^2 I)p(\tilde{x})\,\mathrm{d}\tilde{x}} \tag{14}$$

We then provide methods for computing these integrals for multi-agent pathfinding constraint types.

Circular exclusion constraints. These represent noncollision constraints between circular agents and circular obstacles or other agents. We assume the agent is positioned at $(0, 0) \in \mathbb{R}^2$, with radius $r := r_{agent}$, and that the obstacle is positioned at $(x_{obs}, 0) \in \mathbb{R}^2$ with effective radius $R := r_{obstacle} + r_{agent}$. The feasible region is $\mathcal{C} = \{(x, y) \in \mathbb{R}^2 \mid (x - x_{obs})^2 + y^2 > R^2\}$. Because the agent is at $(x, y) = (0, 0)$, we may replace $x \to 0$ in 14.

Then let $r_1 = \min\{r > 0 \mid \exists x \in \delta\mathcal{C} : |x|^2 = r\}$ and $r_2 = \max\{r > 0 \mid \exists x \in \delta\mathcal{C} : |x|^2 = r\}$ (i.e., the minimum and maximum distances from the agent that are not wholly contained or wholly excluded from the object). Furthermore, let $\theta_{\min}(r) = \min\{\theta \in [-\pi, \pi] \mid (r\cos\theta, r\sin\theta) \in \mathcal{C}\}$, $\theta_{\max}(r) = \max\{\theta \in [-\pi, \pi] \mid (r\cos\theta, r\sin\theta) \in \mathcal{C}\}$. Since the coordinate system is such that the obstacle lies on the x axis, $\theta_{min}(r) = -\theta_{max}(r)$. Then,

$$\int \tilde{x}\mathcal{N}(\tilde{x}; 0, \sigma^2 I)p(\tilde{x})\,\mathrm{d}\tilde{x} = \int_{r_1}^{r_2} r^2 \mathcal{N}(\tilde{x}; 0, \sigma^2 I) \int_{-\pi}^{\pi} (\cos\theta\,\vec{e}_1 + \sin\theta\,\vec{e}_2)\,\mathbb{I}_{\mathcal{C}}(\tilde{x})\,\mathrm{d}\theta\,\mathrm{d}r \tag{15}$$

$$\propto \int_{r_1}^{r_2} r^2 e^{-\frac{r^2}{2\sigma^2}} \int_{\theta_{\min}(r)}^{\theta_{\max}(r)} \cos\theta\,\vec{e}_1\,\mathrm{d}\theta\,\mathrm{d}r \tag{16}$$

$$= \int_{r_1}^{r_2} 2r^2 e^{-\frac{r^2}{2\sigma^2}} \sin\theta_{\max}(r)\,\vec{e}_1\,\mathrm{d}r \tag{17}$$

and,

$$\int \mathcal{N}(\tilde{x}; 0, \sigma^2 I) p(\tilde{x}) \, \mathrm{d}\tilde{x} \propto \int_{r_1}^{r_2} r e^{-\frac{r^2}{2\sigma^2}} \int_{-\pi}^{\pi} \mathbb{I}_{\mathcal{C}}(\tilde{x}) \, \mathrm{d}\theta \, \mathrm{d}r \tag{18}$$

$$= \int_{r_1}^{r_2} 2r e^{-\frac{r^2}{2\sigma^2}} \theta_{\max}(r) \, \mathrm{d}r \tag{19}$$

To compute $\theta_{\max}$,

$$\begin{aligned} &\begin{cases} x^2 + y^2 & = r^2 \\ (x - x_{obs})^2 + y^2 & = R^2 \end{cases} \implies x^2 + y^2 - 2x_{obs}x + x_{obs}^2 = R^2 \\ &\implies -2x_{obs}x + x_{obs}^2 = R^2 - r^2 \\ &\implies x = -\frac{R^2 - r^2 - x_{obs}^2}{2x_{obs}} \\ &\implies y = \sqrt{r^2 - x^2} = r \sin \theta_{\max} \end{aligned}$$

Then $r \sin \theta_{\max}$ may be used directly in 15, and one can compute $\theta_{\max} = \arctan2(y, x)$ for 18.

A.3 Rectangular Feasible Region Score Function Derivation

Consider a 2D region $\mathcal{C} = [x_{\min}, x_{\max}] \times [y_{\min}, y_{\max}]$. A uniform distribution over this region has the probability density:

$$p(x, y) = \frac{\mathbb{I}_{\mathcal{C}}(x, y)}{(x_{\max} - x_{\min})(y_{\max} - y_{\min})}$$

To create a diffusion-based generative model of this region, we can convolve this region with the Gaussian $\mathcal{N}(0, \sigma^2 I)$. Let $q = (x, y) \in \mathbb{R}^2$. Then the distribution at noise level σ is given by:

$$p_\sigma(q) = \int \mathcal{N}(q; \tilde{q}, \sigma^2 I) p(\tilde{q}) \, \mathrm{d}\tilde{q}$$

Because $p(\tilde{q})$ is proportional to the indicator function over the feasible set, $p_\sigma(q)$ is proportional to an integral over the feasible region:

$$p_\sigma(q) \propto \int_{\mathcal{C}} \mathcal{N}(q; \tilde{q}, \sigma^2 I) \, \mathrm{d}\tilde{q}$$

This represents the overlap between a Gaussian distribution with mean $q = (x, y)$ and standard deviation σ and the feasible region. Taking the score,

$$\nabla \log p_\sigma(q) = \frac{\nabla p_\sigma(q)}{p_\sigma(q)}$$

Since $\mathcal{C}$ is rectangular, we can express the denominator in terms of the Gaussian CDF, $\Phi(x) = \int_{-\infty}^{x} \mathcal{N}(\tilde{x}; 0, I) \, \mathrm{d}\tilde{x}$.

$$\begin{aligned} &\int_{x_{\min}}^{x_{\max}} \int_{y_{\min}}^{y_{\max}} \mathcal{N}(q; \tilde{q}, \sigma^2 I) \, \mathrm{d}\tilde{x} \\ &= \left(\int_{x_{\min}}^{x_{\max}} \mathcal{N}(x; \tilde{x}, \sigma^2 I) \, \mathrm{d}\tilde{x} \right) \left(\int_{y_{\min}}^{y_{\max}} \mathcal{N}(y; \tilde{y}, \sigma^2 I) \, \mathrm{d}\tilde{y} \right) \\ &= \left(\Phi\left(\frac{x_{\max} - x}{\sigma} \right) - \Phi\left(\frac{x_{\min} - x}{\sigma} \right) \right) \left(\Phi\left(\frac{y_{\max} - y}{\sigma} \right) - \Phi\left(\frac{y_{\min} - y}{\sigma} \right) \right) \end{aligned}$$

Similarly, the numerator can be computed through the following observation:

$$\int_{x_{\min}}^{x_{\max}} \mathcal{N}(x; \tilde{x}, \sigma^2 I) \, \mathrm{d}\tilde{x} = \int_{0}^{x_{\max}} \mathcal{N}(x; \tilde{x}, \sigma^2 I) \, \mathrm{d}\tilde{x} - \int_{0}^{x_{\min}} \mathcal{N}(x; \tilde{x}, \sigma^2 I) \, \mathrm{d}\tilde{x}$$

By the fundamental theorem of calculus,

$$\frac{\mathrm{d}}{\mathrm{d}x} \int_{x_{\min}}^{x_{\max}} \mathcal{N}(x; \tilde{x}, \sigma^2 I) \, \mathrm{d}\tilde{x} = \mathcal{N}(x; x_{\max}, \sigma^2 I) - \mathcal{N}(x; x_{\min}, \sigma^2 I)$$

Therefore,

$$\nabla p_\sigma(q) = (\mathcal{N}(x; x_{\max}, \sigma^2 I) - \mathcal{N}(x; x_{\min}, \sigma^2 I)) \left(\Phi\left(\frac{y_{\max} - y}{\sigma}\right) - \Phi\left(\frac{y_{\min} - y}{\sigma}\right)\right) \vec{e}_x + \left(\Phi\left(\frac{x_{\max} - x}{\sigma}\right) - \Phi\left(\frac{x_{\min} - x}{\sigma}\right)\right) (\mathcal{N}(y; y_{\max}, \sigma^2 I) - \mathcal{N}(y; y_{\min}, \sigma^2 I)) \vec{e}_y$$

Finally,

$$\nabla \log p_\sigma(q) = \frac{\mathcal{N}(x; x_{\max}, \sigma^2 I) - \mathcal{N}(x; x_{\min}, \sigma^2 I)}{\Phi\left(\frac{x_{\max} - x}{\sigma}\right) - \Phi\left(\frac{x_{\min} - x}{\sigma}\right)} \vec{e}_x + \frac{\mathcal{N}(y; y_{\max}, \sigma^2 I) - \mathcal{N}(y; y_{\min}, \sigma^2 I)}{\Phi\left(\frac{y_{\max} - y}{\sigma}\right) - \Phi\left(\frac{y_{\min} - y}{\sigma}\right)} \vec{e}_y$$

A.4 Rectangular Infeasible Region Score Function Derivation

To compute the score function for a rectangular *infeasible* region, we can reuse many of our prior results. Although technically, the resulting distribution is not normalizable, one can still plug in the equation and get a reasonable result. Let's say now that $\mathcal{C} = ([x_{\min}, x_{\max}] \times [y_{\min}, y_{\max}])^c =: \mathcal{C}^c_{\text{inclusion}}$, the complement of the feasible region used in the previous section (which we shall denote $\mathcal{C}_{\text{inclusion}}$). Then we adopt the perspective that $p_\sigma(x)$ is proportional to the overlap between a Gaussian distribution of mean $q = (x, y)$ and standard deviation σ and the feasible region. We get:

$$\begin{aligned}
\nabla p_\sigma(q) &= \nabla \int_{\mathcal{C}} \mathcal{N}(\tilde{q}; q, \sigma^2 I)\, \mathrm{d}\tilde{q} \\
&= \nabla \left(\int \mathcal{N}(\tilde{q}; q, \sigma^2 I)\, \mathrm{d}\tilde{q} - \int_{\mathcal{C}_{\text{inclusion}}} \mathcal{N}(\tilde{q}; q, \sigma^2 I)\, \mathrm{d}\tilde{q} \right) \\
&= \nabla \left(1 - \int_{\mathcal{C}_{\text{inclusion}}} \mathcal{N}(\tilde{q}; q, \sigma^2 I)\, \mathrm{d}\tilde{q} \right) \\
&= -\nabla \int_{\mathcal{C}_{\text{inclusion}}} \mathcal{N}(\tilde{q}; q, \sigma^2 I)\, \mathrm{d}\tilde{q}
\end{aligned}$$

Therefore, we simply negate the numerator calculation from A.3. For the denominator, we can use a similar trick by subtracting the denominator calculation from the previous section from 1. The overall result is:

$$\nabla \log p_\sigma(q) = \frac{-(\mathcal{N}(x; x_{\max}, \sigma^2 I) - \mathcal{N}(x; x_{\min}, \sigma^2 I)) \left(\Phi\left(\frac{y_{\max} - y}{\sigma}\right) - \Phi\left(\frac{y_{\min} - y}{\sigma}\right)\right)}{1 - \left(\Phi\left(\frac{x_{\max} - x}{\sigma}\right) - \Phi\left(\frac{x_{\min} - x}{\sigma}\right)\right)\left(\Phi\left(\frac{y_{\max} - y}{\sigma}\right) - \Phi\left(\frac{y_{\min} - y}{\sigma}\right)\right)} \vec{e}_x + \frac{-(\mathcal{N}(y; y_{\max}, \sigma^2 I) - \mathcal{N}(y; y_{\min}, \sigma^2 I)) \left(\Phi\left(\frac{x_{\max} - x}{\sigma}\right) - \Phi\left(\frac{x_{\min} - x}{\sigma}\right)\right)}{1 - \left(\Phi\left(\frac{x_{\max} - x}{\sigma}\right) - \Phi\left(\frac{x_{\min} - x}{\sigma}\right)\right)\left(\Phi\left(\frac{y_{\max} - y}{\sigma}\right) - \Phi\left(\frac{y_{\min} - y}{\sigma}\right)\right)} \vec{e}_y$$

A.5 Connection to Projection-Based Methods

The Stein score for a distribution $p_\sigma(x)$ is defined as $\nabla_x \log p_\sigma(x)$. This may be extended to a more elucidating form:

$$\begin{aligned}
\nabla_x \log p_\sigma(x) &= \frac{\nabla_x p_\sigma(x)}{p_\sigma(x)} \\
&= \frac{\int \nabla_x \mathcal{N}(x - \tilde{x}; 0, \sigma^2 I) p(\tilde{x})\, \mathrm{d}\tilde{x}}{p_\sigma(x)} \\
&= \frac{\int \mathcal{N}(x - \tilde{x}; 0, \sigma^2 I) \nabla_x \log \mathcal{N}(x - \tilde{x}; 0, \sigma^2 I) p(\tilde{x})\, \mathrm{d}\tilde{x}}{p_\sigma(x)} \\
&= \frac{1}{\sigma^2} \frac{\int_{\mathbb{R}^n} (\tilde{x} - x) \mathcal{N}(x - \tilde{x}; 0, \sigma^2 I) p(\tilde{x})\, \mathrm{d}\tilde{x}}{p_\sigma(x)}
\end{aligned}$$

For linear inequality or box constraints, the necessary integrals may be computed analytically, and for other low-dimensional constraints, the integral may be computed numerically. Furthermore, we

can note that happens for low-σ behavior. If we assume $p(x) = p(x) = \frac{\mathbb{I}_{\mathcal{C}}(x)}{m(\mathcal{C})}$, then:

$$\begin{aligned}
\lim_{\sigma\to 0} \sigma^2 \nabla_x \log p_\sigma(x) &= \lim_{\sigma\to 0} \frac{\int (\tilde{x} - x)\mathcal{N}(x - \tilde{x}; 0, \sigma^2 I) p(\tilde{x})\, \mathrm{d}\tilde{x}}{\int \mathcal{N}(x - \tilde{x}; 0, \sigma^2 I) p(\tilde{x})\, \mathrm{d}\tilde{x}} \\
&= \lim_{\sigma\to 0} \frac{\int (\tilde{x} - x) \exp\left(-\frac{(x-\tilde{x})^2}{2\sigma^2}\right) p(\tilde{x})\, \mathrm{d}\tilde{x}}{\int \exp\left(-\frac{(x-\tilde{x})^2}{2\sigma^2}\right) p(\tilde{x})\, \mathrm{d}\tilde{x}} \\
&= \lim_{\sigma\to 0} \frac{\int_{\mathcal{C}} (\tilde{x} - x) \exp\left(-\frac{(x-\tilde{x})^2}{2\sigma^2}\right) \mathrm{d}\tilde{x}}{\int_{\mathcal{C}} \exp\left(-\frac{(x-\tilde{x})^2}{2\sigma^2}\right) \mathrm{d}\tilde{x}} \\
&= \Pi_{\mathcal{C}}(x) - x
\end{aligned}$$

That is, at very low σ, taking a discrete-time Langevin step with size σ^2 is precisely the projection operator:

$$\begin{aligned}
x_{t+1} &= x_t + \sigma^2 \nabla_x \log p_\sigma(x_t) + 2\sigma z \\
&= x_t + (\Pi_{\mathcal{C}}(x_t) - x_t) + 2\sigma z \\
&= \Pi_{\mathcal{C}}(x_t) + 2\sigma z
\end{aligned}$$

Since $\sigma \ll 1$, the Brownian noise term is negligible.

A.6 Larger Hyperparameter Sweep

We vary the Langevin step size η for TFDP and CHOMP-style SDF guidance, with SDF scaling factors $\alpha \in \{0.05, 0.1, 0.2, 0.5\}$. The reported Wilson confidence intervals use 10 trials of 30 maps each. Gray columns identify TFDP, and arrows indicate whether higher (↑) or lower (↓) values are better. Bold marks the best comparable central values, including ties; the captions specify the comparison groups.

TFDP, smaller steps. Table 3 tests whether small updates suffice for reliable refinement. Success remains at 0.01 for $\eta \leq 0.02$, while $\eta = 0.10$ raises it to 0.94–0.97 with 6 robots. The same step size reaches only 0.31–0.68 with 18 robots. Thus, the smaller-step regime is particularly restrictive as the number of interacting robots increases.

Map	Robots	Step size η			
		0.01	0.02	0.05	0.10
Basic	6	0.01 ± 0.01	0.01 ± 0.01	0.31 ± 0.06	**0.97** ± 0.02
	12	0.01 ± 0.01	0.01 ± 0.01	0.07 ± 0.03	**0.87** ± 0.04
	18	0.01 ± 0.01	0.01 ± 0.01	0.01 ± 0.01	**0.68** ± 0.06
Dense	6	0.01 ± 0.01	0.01 ± 0.01	0.30 ± 0.06	**0.96** ± 0.02
	12	0.01 ± 0.01	0.01 ± 0.01	0.07 ± 0.03	**0.84** ± 0.04
	18	0.01 ± 0.01	0.01 ± 0.01	0.01 ± 0.01	**0.56** ± 0.06
Room	6	0.01 ± 0.01	0.01 ± 0.01	0.24 ± 0.05	**0.94** ± 0.03
	12	0.01 ± 0.01	0.01 ± 0.01	0.03 ± 0.02	**0.69** ± 0.06
	18	0.01 ± 0.01	0.01 ± 0.01	0.01 ± 0.01	**0.31** ± 0.06
Shelf	6	0.01 ± 0.01	0.01 ± 0.01	0.30 ± 0.06	**0.96** ± 0.02
	12	0.01 ± 0.01	0.01 ± 0.01	0.04 ± 0.02	**0.75** ± 0.05
	18	0.01 ± 0.01	0.01 ± 0.01	0.02 ± 0.02	**0.61** ± 0.06

Table 3: TFDP: success probability (↑), with Wilson intervals, for smaller step sizes. Bold marks row maxima within this displayed range.

TFDP, larger steps. Table 4 shows that larger updates substantially improve feasibility. At $\eta = 0.50$, success is at least 0.95 in every setting. Increasing η further raises the Dense 18-robot point estimate from 0.95 to 0.97, but lowers Room from 0.99 to 0.94 at $\eta = 1.00$. This supports $\eta = 0.50$ as a broadly effective choice, without identifying a universal optimum.

Map	Robots	Step size η			
		0.20	0.50	0.80	1.00
Basic	6	**0.99** ± 0.01	**0.99** ± 0.01	**0.99** ± 0.01	**0.99** ± 0.01
	12	**0.99** ± 0.01	**0.99** ± 0.01	**0.99** ± 0.01	**0.99** ± 0.01
	18	0.97 ± 0.02	**0.99** ± 0.01	**0.99** ± 0.01	**0.99** ± 0.01
Dense	6	**0.99** ± 0.01	**0.99** ± 0.01	**0.99** ± 0.01	**0.99** ± 0.01
	12	0.96 ± 0.02	0.98 ± 0.02	**0.99** ± 0.01	0.98 ± 0.02
	18	0.89 ± 0.04	0.95 ± 0.03	**0.97** ± 0.02	**0.97** ± 0.02
Room	6	**0.99** ± 0.01	**0.99** ± 0.01	**0.99** ± 0.01	**0.99** ± 0.01
	12	0.97 ± 0.02	**0.99** ± 0.01	0.98 ± 0.02	0.98 ± 0.02
	18	0.92 ± 0.03	**0.99** ± 0.01	0.95 ± 0.03	0.94 ± 0.03
Shelf	6	**0.99** ± 0.01	**0.99** ± 0.01	**0.99** ± 0.01	**0.99** ± 0.01
	12	0.98 ± 0.01	**0.99** ± 0.01	**0.99** ± 0.01	**0.99** ± 0.01
	18	0.95 ± 0.03	0.98 ± 0.01	**0.99** ± 0.01	0.98 ± 0.01

Table 4: TFDP: success probability (↑), with Wilson intervals, for larger step sizes. Bold marks row maxima within this displayed range.

SDF with $\alpha = 0.1$, smaller steps. Table 5 shows the interaction between step size and robot count. At $\eta = 0.10$, Room success decreases from 0.94 with 6 robots to 0.31 with 18, while Basic decreases from 0.97 to 0.68. Even the best step size in this panel leaves a substantial feasibility gap as coordination becomes more demanding.

Map	Robots	Step size η			
		0.01	0.02	0.05	0.10
Basic	6	0.01 ± 0.01	0.01 ± 0.01	0.31 ± 0.06	**0.97** ± 0.02
	12	0.01 ± 0.01	0.01 ± 0.01	0.07 ± 0.03	**0.87** ± 0.04
	18	0.01 ± 0.01	0.01 ± 0.01	0.01 ± 0.01	**0.68** ± 0.06
Dense	6	0.01 ± 0.01	0.01 ± 0.01	0.30 ± 0.06	**0.96** ± 0.02
	12	0.01 ± 0.01	0.01 ± 0.01	0.07 ± 0.03	**0.84** ± 0.04
	18	0.01 ± 0.01	0.01 ± 0.01	0.01 ± 0.01	**0.56** ± 0.06
Room	6	0.01 ± 0.01	0.01 ± 0.01	0.24 ± 0.05	**0.94** ± 0.03
	12	0.01 ± 0.01	0.01 ± 0.01	0.03 ± 0.02	**0.69** ± 0.06
	18	0.01 ± 0.01	0.01 ± 0.01	0.01 ± 0.01	**0.31** ± 0.06
Shelf	6	0.01 ± 0.01	0.01 ± 0.01	0.30 ± 0.06	**0.96** ± 0.02
	12	0.01 ± 0.01	0.01 ± 0.01	0.04 ± 0.02	**0.75** ± 0.05
	18	0.01 ± 0.01	0.01 ± 0.01	0.02 ± 0.02	**0.61** ± 0.06

Table 5: SDF guidance with $\alpha = 0.1$: success probability (↑), with Wilson intervals, for smaller step sizes. Bold marks row maxima within this displayed range.

SDF with $\alpha = 0.1$, larger steps. Table 6 shows high success over several neighboring step sizes. Basic remains at 0.99 for all robot counts when $\eta \geq 0.50$, and Shelf remains between 0.98 and 0.99. Dense and Room retain greater sensitivity with 18 robots, indicating that a broad effective range on simpler cases does not imply uniform robustness across layouts.

Map	Robots	Step size η			
		0.20	0.50	0.80	1.00
Basic	6	**0.99** $\pm$ 0.01	**0.99** $\pm$ 0.01	**0.99** $\pm$ 0.01	**0.99** $\pm$ 0.01
	12	**0.99** $\pm$ 0.01	**0.99** $\pm$ 0.01	**0.99** $\pm$ 0.01	**0.99** $\pm$ 0.01
	18	0.97 $\pm$ 0.02	**0.99** $\pm$ 0.01	**0.99** $\pm$ 0.01	**0.99** $\pm$ 0.01
Dense	6	**0.99** $\pm$ 0.01	**0.99** $\pm$ 0.01	**0.99** $\pm$ 0.01	**0.99** $\pm$ 0.01
	12	0.96 $\pm$ 0.02	0.98 $\pm$ 0.02	**0.99** $\pm$ 0.01	0.98 $\pm$ 0.02
	18	0.89 $\pm$ 0.04	0.95 $\pm$ 0.03	**0.97** $\pm$ 0.02	**0.97** $\pm$ 0.02
Room	6	**0.99** $\pm$ 0.01	**0.99** $\pm$ 0.01	**0.99** $\pm$ 0.01	**0.99** $\pm$ 0.01
	12	0.97 $\pm$ 0.02	**0.99** $\pm$ 0.01	0.98 $\pm$ 0.02	0.98 $\pm$ 0.02
	18	0.92 $\pm$ 0.03	**0.99** $\pm$ 0.01	0.95 $\pm$ 0.03	0.94 $\pm$ 0.03
Shelf	6	**0.99** $\pm$ 0.01	**0.99** $\pm$ 0.01	**0.99** $\pm$ 0.01	**0.99** $\pm$ 0.01
	12	0.98 $\pm$ 0.01	**0.99** $\pm$ 0.01	**0.99** $\pm$ 0.01	**0.99** $\pm$ 0.01
	18	0.95 $\pm$ 0.03	0.98 $\pm$ 0.01	**0.99** $\pm$ 0.01	0.98 $\pm$ 0.01

Table 6: SDF guidance with $\alpha = 0.1$: success probability (↑), with Wilson intervals, for larger step sizes. Bold marks row maxima within this displayed range.

SDF with $\alpha = 0.5$, smaller steps. Table 7 tests the largest SDF scale in the small-step regime. The highest success in every row occurs at $\eta = 0.10$, while $\eta \leq 0.02$ yields only 0.01. The remaining variation at $\eta = 0.10$, from 0.31 on Room with 18 robots to 0.97 on Basic with 6, shows that the same update scale can yield very different planning outcomes.

Map	Robots	Step size η			
		0.01	0.02	0.05	0.10
Basic	6	0.01 $\pm$ 0.01	0.01 $\pm$ 0.01	0.31 $\pm$ 0.06	**0.97** $\pm$ 0.02
	12	0.01 $\pm$ 0.01	0.01 $\pm$ 0.01	0.07 $\pm$ 0.03	**0.87** $\pm$ 0.04
	18	0.01 $\pm$ 0.01	0.01 $\pm$ 0.01	0.01 $\pm$ 0.01	**0.68** $\pm$ 0.06
Dense	6	0.01 $\pm$ 0.01	0.01 $\pm$ 0.01	0.30 $\pm$ 0.06	**0.96** $\pm$ 0.02
	12	0.01 $\pm$ 0.01	0.01 $\pm$ 0.01	0.07 $\pm$ 0.03	**0.84** $\pm$ 0.04
	18	0.01 $\pm$ 0.01	0.01 $\pm$ 0.01	0.01 $\pm$ 0.01	**0.56** $\pm$ 0.06
Room	6	0.01 $\pm$ 0.01	0.01 $\pm$ 0.01	0.24 $\pm$ 0.05	**0.94** $\pm$ 0.03
	12	0.01 $\pm$ 0.01	0.01 $\pm$ 0.01	0.03 $\pm$ 0.02	**0.69** $\pm$ 0.06
	18	0.01 $\pm$ 0.01	0.01 $\pm$ 0.01	0.01 $\pm$ 0.01	**0.31** $\pm$ 0.06
Shelf	6	0.01 $\pm$ 0.01	0.01 $\pm$ 0.01	0.30 $\pm$ 0.06	**0.96** $\pm$ 0.02
	12	0.01 $\pm$ 0.01	0.01 $\pm$ 0.01	0.04 $\pm$ 0.02	**0.75** $\pm$ 0.05
	18	0.01 $\pm$ 0.01	0.01 $\pm$ 0.01	0.02 $\pm$ 0.02	**0.61** $\pm$ 0.06

Table 7: SDF guidance with $\alpha = 0.5$: success probability (↑), with Wilson intervals, for smaller step sizes. Bold marks row maxima within this displayed range.

SDF with $\alpha = 0.5$, larger steps. Table 8 reports at least 0.95 success in every setting at $\eta = 0.50$. Larger steps offer little improvement on Basic and Shelf, while Room with 18 robots declines to 0.94 at $\eta = 1.00$. The table supports a moderate step size across these cases; small differences among the larger-step estimates should be read together with their uncertainty intervals.

Map	Robots	Step size η			
		0.20	0.50	0.80	1.00
Basic	6	**0.99** $\pm$ 0.01	**0.99** $\pm$ 0.01	**0.99** $\pm$ 0.01	**0.99** $\pm$ 0.01
	12	**0.99** $\pm$ 0.01	**0.99** $\pm$ 0.01	**0.99** $\pm$ 0.01	**0.99** $\pm$ 0.01
	18	0.97 $\pm$ 0.02	**0.99** $\pm$ 0.01	**0.99** $\pm$ 0.01	**0.99** $\pm$ 0.01
Dense	6	**0.99** $\pm$ 0.01	**0.99** $\pm$ 0.01	**0.99** $\pm$ 0.01	**0.99** $\pm$ 0.01
	12	0.96 $\pm$ 0.02	0.98 $\pm$ 0.02	**0.99** $\pm$ 0.01	0.98 $\pm$ 0.02
	18	0.89 $\pm$ 0.04	0.95 $\pm$ 0.03	**0.97** $\pm$ 0.02	**0.97** $\pm$ 0.02
Room	6	**0.99** $\pm$ 0.01	**0.99** $\pm$ 0.01	**0.99** $\pm$ 0.01	**0.99** $\pm$ 0.01
	12	0.97 $\pm$ 0.02	**0.99** $\pm$ 0.01	0.98 $\pm$ 0.02	0.98 $\pm$ 0.02
	18	0.92 $\pm$ 0.03	**0.99** $\pm$ 0.01	0.95 $\pm$ 0.03	0.94 $\pm$ 0.03
Shelf	6	**0.99** $\pm$ 0.01	**0.99** $\pm$ 0.01	**0.99** $\pm$ 0.01	**0.99** $\pm$ 0.01
	12	0.98 $\pm$ 0.01	**0.99** $\pm$ 0.01	**0.99** $\pm$ 0.01	**0.99** $\pm$ 0.01
	18	0.95 $\pm$ 0.03	0.98 $\pm$ 0.01	**0.99** $\pm$ 0.01	0.98 $\pm$ 0.01

Table 8: SDF guidance with $\alpha = 0.5$: success probability (↑), with Wilson intervals, for larger step sizes. Bold marks row maxima within this displayed range.

A.7 Single-Agent Reduction (RRT-Connect)

Table 9 evaluates whether the multi-robot problem can be solved directly by applying RRT-Connect in the joint configuration space on Dense. Success falls from 0.83 with 4 robots to 0.07 with 10, while mean runtime rises from 2.97 to 21.98 seconds. Time on success conditions on solved instances and therefore does not reflect the cost of unsuccessful searches; mean runtime includes both outcomes. These results expose the practical scaling difficulty of joint-space search under the tested budget, despite its asymptotic completeness properties.

Robots	Success ↑	Time on success ↓	Mean runtime ↓
4	0.83 $\pm$ 0.13	2.07 $\pm$ 0.49	2.97 $\pm$ 0.66
6	0.34 $\pm$ 0.17	4.55 $\pm$ 1.91	11.35 $\pm$ 1.42
8	0.14 $\pm$ 0.11	3.37 $\pm$ 2.87	17.30 $\pm$ 1.10
10	0.07 $\pm$ 0.07	–	21.98 $\pm$ 1.09

Table 9: RRT-Connect on Dense. Success is a fraction; times are in seconds. Conditional solve time is reported for successful trials when available; "–" denotes an unreported value.

A.8 Coarse Planner Ablation

We compare straight-line interpolation, Brownian bridges with maximum variance $\sigma^2 = 0.025^2$, Voronoi roadmaps, and RRT. The tables report success (S, in %), runtime (T, seconds), and path length (P). Path length must be interpreted jointly with success, since short trajectories can remain infeasible.

TFDP with simple initializations. Table 10 tests refinement from straight lines and Brownian bridges. Straight lines achieve 89–98% success on Basic, but both initializers reach only 0–17% on Room and Shelf. Brownian perturbations do not consistently improve feasibility and reduce Dense success with 18 robots from 68% to 45%. Thus, local trajectory variation alone does not resolve the routing difficulty of structured maps; the short runtimes and paths in failed cases should not be interpreted as better solutions.

Map	Robots	Linear			Brownian bridge		
		S↑	T↓	P↓	S↑	T↓	P↓
Basic	6	**98**	0.15 ± 0.00	**1.01** ± 0.00	91	**0.14** ± 0.01	1.02 ± 0.00
	12	**96**	0.19 ± 0.00	**1.02** ± 0.00	85	**0.17** ± 0.01	**1.02** ± 0.00
	18	**89**	0.25 ± 0.00	**1.02** ± 0.00	80	**0.23** ± 0.01	1.03 ± 0.00
Dense	6	**92**	0.16 ± 0.00	**1.05** ± 0.00	86	**0.15** ± 0.01	1.07 ± 0.00
	12	**87**	0.23 ± 0.00	**1.06** ± 0.00	72	**0.21** ± 0.01	1.07 ± 0.00
	18	**68**	0.33 ± 0.00	**1.07** ± 0.00	45	**0.30** ± 0.01	1.09 ± 0.00
Room	6	**7**	0.12 ± 0.00	**1.08** ± 0.00	**7**	**0.11** ± 0.00	1.10 ± 0.00
	12	**0**	0.15 ± 0.00	1.09 ± 0.00	**0**	**0.14** ± 0.00	1.10 ± 0.00
	18	**0**	0.19 ± 0.00	1.10 ± 0.00	**0**	**0.18** ± 0.01	1.11 ± 0.00
Shelf	6	16	0.12 ± 0.00	**1.02** ± 0.00	**17**	**0.11** ± 0.00	1.04 ± 0.00
	12	**4**	0.15 ± 0.00	**1.03** ± 0.00	3	**0.14** ± 0.01	1.04 ± 0.00
	18	**0**	0.19 ± 0.00	1.04 ± 0.00	**0**	**0.17** ± 0.01	1.05 ± 0.00

Table 10: TFDP with linear and Brownian-bridge initialization. Bold compares the two initializers within each row and metric; path length is ranked only for initializers with nonzero success.

TFDP with obstacle-aware initializations. Table 11 shows that Voronoi and RRT restore 99–100% success on Room and Shelf. Voronoi yields lower total runtime in every row, while RRT yields shorter paths on the structured maps: on Room with 18 robots, path length decreases from 2.00 to 1.56, at a runtime increase from 0.82 to 2.06 seconds. On Dense with 18 robots, RRT also raises success from 77% to 92%. Initialization therefore affects feasibility, efficiency, and path quality.

Map	Robots	Voronoi			RRT		
		S↑	T↓	P↓	S↑	T↓	P↓
Basic	6	**100**	**0.30** ± 0.00	1.20 ± 0.00	**100**	0.40 ± 0.02	**1.09** ± 0.01
	12	**100**	**0.47** ± 0.00	1.22 ± 0.00	99	0.54 ± 0.02	**1.09** ± 0.00
	18	99	**0.71** ± 0.00	1.24 ± 0.00	**100**	0.82 ± 0.04	**1.12** ± 0.00
Dense	6	96	**0.39** ± 0.00	**1.29** ± 0.01	**99**	0.67 ± 0.02	1.32 ± 0.02
	12	90	**0.63** ± 0.00	**1.32** ± 0.00	**98**	1.13 ± 0.02	1.37 ± 0.01
	18	77	**0.90** ± 0.00	**1.34** ± 0.00	**92**	1.91 ± 0.04	1.39 ± 0.01
Room	6	**100**	**0.31** ± 0.00	1.93 ± 0.01	**100**	0.71 ± 0.01	**1.43** ± 0.01
	12	**100**	**0.52** ± 0.00	1.96 ± 0.01	**100**	1.27 ± 0.02	**1.53** ± 0.01
	18	**100**	**0.82** ± 0.00	2.00 ± 0.00	**100**	2.06 ± 0.03	**1.56** ± 0.01
Shelf	6	**100**	**0.31** ± 0.00	1.68 ± 0.01	**100**	0.53 ± 0.01	**1.30** ± 0.01
	12	**100**	**0.51** ± 0.00	1.73 ± 0.01	99	0.90 ± 0.02	**1.34** ± 0.01
	18	**100**	**0.79** ± 0.00	1.76 ± 0.00	99	1.52 ± 0.04	**1.38** ± 0.01

Table 11: TFDP with Voronoi and RRT initialization. Bold compares the two initializers within each row and metric; path length is ranked only for initializers with nonzero success.

MPPI with simple initializations. Table 12 asks whether the same initialization difficulty appears with sampling-based updates. MPPI reaches 96–98% success on Basic with straight lines, but at most 17% on Room and 25% on Shelf. With 18 robots on Dense, Brownian bridges reduce success from 59% to 42%. The similar pattern for MPPI and TFDP indicates that poor initial routes are a shared obstacle for local refinement, rather than a limitation specific to analytical scores.

Map	Robots	Linear			Brownian bridge		
		S↑	T↓	P↓	S↑	T↓	P↓
Basic	6	98	0.65 ± 0.01	**1.03** ± 0.00	**99**	**0.60** ± 0.03	1.04 ± 0.00
	12	**97**	0.65 ± 0.01	**1.04** ± 0.00	95	**0.61** ± 0.03	**1.04** ± 0.00
	18	**96**	0.68 ± 0.01	**1.05** ± 0.00	87	**0.61** ± 0.03	**1.05** ± 0.00
Dense	6	**92**	0.66 ± 0.01	**1.08** ± 0.01	83	**0.62** ± 0.03	1.10 ± 0.00
	12	**78**	0.67 ± 0.02	**1.12** ± 0.00	63	**0.62** ± 0.04	1.14 ± 0.00
	18	**59**	0.70 ± 0.01	**1.14** ± 0.00	42	**0.64** ± 0.04	1.15 ± 0.00
Room	6	**17**	0.65 ± 0.01	**1.13** ± 0.00	15	**0.60** ± 0.03	**1.13** ± 0.00
	12	**2**	0.66 ± 0.01	**1.14** ± 0.00	0	**0.61** ± 0.03	1.14 ± 0.00
	18	**0**	0.66 ± 0.01	1.15 ± 0.00	**0**	**0.61** ± 0.03	1.15 ± 0.00
Shelf	6	**25**	0.62 ± 0.01	1.06 ± 0.00	23	**0.59** ± 0.03	**1.05** ± 0.00
	12	**6**	0.63 ± 0.01	**1.07** ± 0.00	2	**0.60** ± 0.03	1.08 ± 0.00
	18	**1**	0.62 ± 0.02	**1.08** ± 0.00	0	**0.59** ± 0.03	1.09 ± 0.00

Table 12: MPPI with linear and Brownian-bridge initialization. Bold compares the two initializers within each row and metric; path length is ranked only for initializers with nonzero success.

MPPI with obstacle-aware initializations. Table 13 reports 98–100% success on Room and Shelf, confirming that informative initial routes also benefit MPPI. RRT shortens paths on both layouts, generally at higher runtime; on Room with 18 robots, path length decreases from 2.05 to 1.77 while runtime increases from 1.09 to 1.66 seconds. On Dense with 18 robots, RRT instead improves success from 77% to 94% at the cost of longer paths. No initializer dominates all three metrics.

Map	Robots	Voronoi			RRT		
		S↑	T↓	P↓	S↑	T↓	P↓
Basic	6	**100**	0.69 ± 0.03	1.21 ± 0.01	**100**	**0.63** ± 0.01	**1.13** ± 0.01
	12	**100**	0.80 ± 0.03	1.26 ± 0.00	**100**	**0.72** ± 0.01	**1.16** ± 0.01
	18	**99**	0.93 ± 0.03	1.29 ± 0.00	**99**	**0.81** ± 0.02	**1.19** ± 0.00
Dense	6	95	**0.79** ± 0.03	**1.34** ± 0.01	**99**	0.85 ± 0.02	1.46 ± 0.02
	12	89	**0.95** ± 0.03	**1.40** ± 0.00	**99**	1.09 ± 0.02	1.53 ± 0.01
	18	77	**1.14** ± 0.04	**1.44** ± 0.00	**94**	1.40 ± 0.02	1.59 ± 0.01
Room	6	**100**	**0.74** ± 0.03	1.90 ± 0.01	**100**	0.93 ± 0.01	**1.57** ± 0.01
	12	**100**	**0.90** ± 0.03	1.98 ± 0.00	**100**	1.31 ± 0.02	**1.69** ± 0.02
	18	**100**	**1.09** ± 0.03	2.05 ± 0.00	99	1.66 ± 0.02	**1.77** ± 0.01
Shelf	6	**100**	**0.74** ± 0.03	1.64 ± 0.01	**100**	**0.74** ± 0.02	**1.46** ± 0.02
	12	**100**	**0.91** ± 0.03	1.73 ± 0.01	**100**	1.00 ± 0.02	**1.54** ± 0.02
	18	**100**	**1.10** ± 0.03	1.80 ± 0.00	98	1.24 ± 0.02	**1.62** ± 0.01

Table 13: MPPI with Voronoi and RRT initialization. Bold compares the two initializers within each row and metric; path length is ranked only for initializers with nonzero success.

A.9 MPPI CPU RUNTIME

Table 14 compares MPPI runtime on the CPU used for TFDP. As robot count grows from 6 to 18, runtime increases from 8.18 to 32.36 seconds on Basic and from 10.84 to 40.98 seconds on Dense. All entries exceed the corresponding TFDP times in Table 1. Thus, TFDP retains its runtime advantage on shared hardware, although this comparison does not isolate the cost of individual algorithmic components.

Map	Robots	MPPI time (s) ↓
Basic	6	8.18 ± 0.13
	12	19.43 ± 0.26
	18	32.36 ± 0.47
Dense	6	10.84 ± 0.26
	12	24.85 ± 0.54
	18	40.98 ± 0.86
Room	6	9.87 ± 0.20
	12	22.88 ± 0.41
	18	38.62 ± 0.63
Shelf	6	9.12 ± 0.17
	12	21.39 ± 0.35
	18	36.31 ± 0.53

Table 14: MPPI runtime on the same CPU as TFDP; lower is better.

A.10 EFFECT OF NOISE

Table 15 compares stochastic refinement with deterministic annealed updates. Removing noise matches or slightly improves success on the standard maps. In the crowded stress tests, noise instead improves the point estimates: with 12 robots, Bottleneck rises from 0.38 to 0.45, and Standoff from 0.39 to 0.45. This suggests that stochastic exploration is most useful when many robots compete for limited passage space. The uncertainty ranges overlap, so these gains describe observed trends without establishing statistical significance.

Scenario	TFDP, noisy ↑	TFDP, noiseless ↑
Dense (6 agents)	**0.99** $\pm$ 0.01	**0.99** $\pm$ 0.01
Dense (12 agents)	**0.98** $\pm$ 0.02	**0.98** $\pm$ 0.01
Dense (18 agents)	0.93 $\pm$ 0.03	**0.95** $\pm$ 0.02
Basic (6 agents)	0.99 $\pm$ 0.01	**1.00** $\pm$ 0.00
Basic (12 agents)	**0.99** $\pm$ 0.01	**0.99** $\pm$ 0.01
Basic (18 agents)	0.99 $\pm$ 0.01	**1.00** $\pm$ 0.00
Room (6 agents)	0.99 $\pm$ 0.01	**1.00** $\pm$ 0.00
Room (12 agents)	**0.99** $\pm$ 0.01	**0.99** $\pm$ 0.01
Room (18 agents)	**0.99** $\pm$ 0.01	**0.99** $\pm$ 0.01
Shelf (6 agents)	0.99 $\pm$ 0.01	**1.00** $\pm$ 0.00
Shelf (12 agents)	**0.99** $\pm$ 0.01	**0.99** $\pm$ 0.01
Shelf (18 agents)	0.98 $\pm$ 0.01	**0.99** $\pm$ 0.01
Bottleneck (2 agents)	**0.99** $\pm$ 0.01	0.98 $\pm$ 0.01
Bottleneck (4 agents)	0.94 $\pm$ 0.03	**0.95** $\pm$ 0.03
Bottleneck (6 agents)	**0.90** $\pm$ 0.04	**0.90** $\pm$ 0.04
Bottleneck (8 agents)	**0.81** $\pm$ 0.05	0.75 $\pm$ 0.05
Bottleneck (12 agents)	**0.45** $\pm$ 0.06	0.38 $\pm$ 0.06
Standoff (2 agents)	**0.99** $\pm$ 0.01	**0.99** $\pm$ 0.01
Standoff (4 agents)	**0.94** $\pm$ 0.03	0.93 $\pm$ 0.03
Standoff (6 agents)	**0.90** $\pm$ 0.04	0.88 $\pm$ 0.04
Standoff (8 agents)	**0.81** $\pm$ 0.05	0.76 $\pm$ 0.05
Standoff (12 agents)	**0.45** $\pm$ 0.06	0.39 $\pm$ 0.06

Table 15: Success fractions with noisy and deterministic annealed refinement. Bold marks the higher central value in each row, including ties.

A.11 Effect of σ and T

We vary the initial smoothing scale σ and temperature T, which are exponentially annealed to $\sigma = 0.001$ and $T = 0.0001$. These experiments use straight-line initialization to examine whether the refinement schedule can compensate for a simple initial trajectory. Rows vary σ, columns vary T, and each entry reports the success fraction.

Dense maps. Table 16 shows that smoothing and temperature must be selected jointly. At $\sigma = 0.05$, increasing T from 0.0001 to 0.8 raises success from 0.40 to 0.72; at $\sigma = 0.2$, the same change reduces it from 0.64 to 0.04. The maximum of 0.72 occurs at several parameter pairs. Additional exploration can therefore improve refinement from straight lines, but excessive noise at large smoothing scales can sharply reduce feasibility.

$\sigma \backslash T$	0.0001	0.0005	0.001	0.005	0.01	0.05	0.1	0.3	0.5	0.8
0.01	0.28	0.28	0.24	0.28	0.24	0.28	0.28	0.28	0.28	**0.40**
0.025	0.32	0.32	0.36	0.36	0.36	0.32	0.36	0.36	**0.56**	**0.56**
0.05	0.40	0.40	0.40	0.36	0.40	0.36	0.40	0.64	0.68	**0.72**
0.075	0.48	0.48	0.48	0.48	0.48	0.44	0.44	0.60	**0.68**	0.60
0.1	0.44	0.44	0.44	0.44	0.44	0.44	0.44	**0.72**	0.64	0.40
0.125	0.48	0.48	0.48	0.48	0.48	0.52	0.60	0.60	**0.64**	0.32
0.15	0.52	0.52	0.52	0.52	0.52	0.56	0.64	**0.68**	0.52	0.12
0.175	0.60	0.64	0.64	0.56	0.56	0.64	**0.72**	0.68	0.20	0.08
0.2	0.64	0.60	0.60	**0.68**	0.64	0.60	0.64	0.52	0.12	0.04

Table 16: TFDP success fraction (↑) on Dense. Rows vary initial σ and columns vary initial T; bold marks the best temperature in each row, including ties.

Basic maps. Table 17 reaches 0.96 success at $(\sigma, T) = (0.05, 0.8)$ and $(0.1, 0.5)$, compared with 0.72 at the lowest temperature for either smoothing scale. This shows that suitable stochastic refinement can substantially improve a simple initialization in this setting. The benefit is not monotonic: at $\sigma = 0.2$, success falls from 0.84 at $T = 0.1$ to 0.08 at $T = 0.8$. Even on Basic, increasing temperature alone is insufficient.

$\sigma \backslash T$	0.0001	0.0005	0.001	0.005	0.01	0.05	0.1	0.3	0.5	0.8
0.01	0.72	0.72	0.72	0.72	0.72	0.72	0.72	0.72	**0.76**	0.68
0.025	0.72	0.72	0.72	0.72	0.72	0.72	0.72	0.76	0.72	**0.88**
0.05	0.72	0.72	0.72	0.72	0.72	0.72	0.72	0.80	0.88	**0.96**
0.075	0.72	0.72	0.72	0.72	0.72	0.76	0.76	0.84	**0.92**	0.88
0.1	0.72	0.72	0.72	0.72	0.72	0.72	0.76	0.88	**0.96**	0.76
0.125	0.76	0.76	0.76	0.76	0.76	0.72	0.76	**0.92**	0.88	0.48
0.15	0.76	0.76	0.76	0.76	0.76	0.76	**0.84**	0.80	0.64	0.40
0.175	0.72	0.72	0.72	0.72	0.72	0.76	**0.92**	0.76	0.64	0.20
0.2	0.76	0.76	0.76	0.76	0.76	0.80	**0.84**	0.72	0.52	0.08

Table 17: TFDP success fraction (↑) on Basic. Rows vary initial σ and columns vary initial T; bold marks the best temperature in each row, including ties.

A.12 Density/Stress Tests

We test maps with randomized start and goal positions and a wall opening wide enough for one robot. In Bottleneck, robots start on the same side; in Standoff, half the start–goal pairs are reversed to create opposing traffic. Figure 4 shows the layouts.

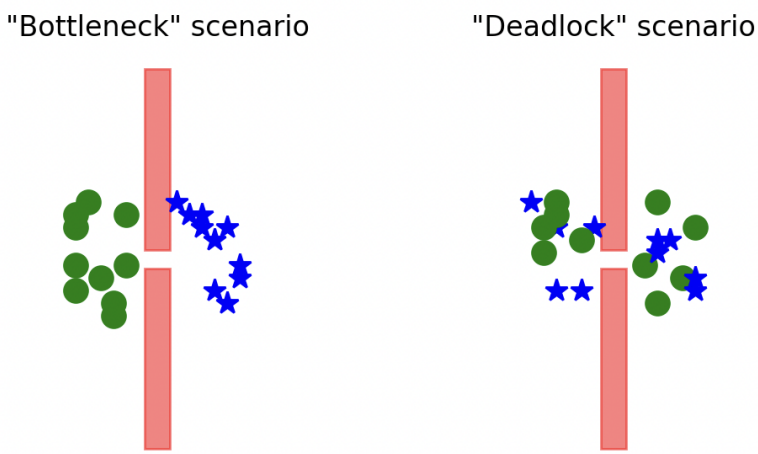


Figure 4: Stress-test layouts: Bottleneck (left) and Standoff (right, labeled "Deadlock"). Both have an opening wide enough for one robot.

Bottleneck. Table 18 measures feasibility when robots must traverse a shared narrow opening in the same direction. TFDP has higher success point estimates at every tested count; with 8 robots, success is 0.69 versus MPPI's 0.20. However, TFDP falls to 0.31 with 12 robots, while MPPI reaches 0.02. The results show a substantial advantage under congestion, but also identify crowded passages as a remaining source of failure.

Robots	MPPI success ↑	TFDP success ↑
2	0.96 ± 0.02	**0.98** ± 0.02
4	0.76 ± 0.05	**0.95** ± 0.03
6	0.51 ± 0.06	**0.89** ± 0.04
8	0.20 ± 0.05	**0.69** ± 0.06
12	0.02 ± 0.02	**0.31** ± 0.06

Table 18: Bottleneck success fractions. Bold marks the higher central value in each row.

Standoff. Table 19 introduces opposing traffic through the same type of opening, requiring robots moving in different directions to coordinate passage. TFDP achieves 0.70 success with 8 robots, compared with 0.23 for MPPI; with 12 robots, the values fall to 0.32 and 0.03. The advantage therefore persists under opposing traffic, while the low success at the largest count shows that analytical refinement does not consistently resolve these tightly coupled interactions.

Robots	MPPI success ↑	TFDP success ↑
2	0.96 ± 0.02	**0.99** ± 0.01
4	0.78 ± 0.05	**0.92** ± 0.03
6	0.51 ± 0.06	**0.85** ± 0.05
8	0.23 ± 0.05	**0.70** ± 0.06
12	0.03 ± 0.02	**0.32** ± 0.06

Table 19: Standoff success fractions. Bold marks the higher central value in each row.

A.13 Locality of Learned Scores

We test how much temporal context is useful for score estimation by training a multilayer perceptron with AdaLN-Zero conditioning on noise level for a simple trajectory distribution. This network is a diagnostic model; TFDP itself remains training-free. Table 20 reports final-epoch mean squared error as the temporal window and noise level vary.

Results. Expanding the window from 1 to 3 reduces error at every noise level, showing that neighboring waypoints carry useful information. At $\sigma = 0.05$, error decreases from 0.8679 to 0.3849, then reaches its minimum of 0.2988 at window 7; window 9 gives no further improvement. At $\sigma = 0.2$, error continues to decrease through window 9, reaching 0.2037. Thus, short neighborhoods are effective in this diagnostic, while broader context remains useful at higher noise. This supports the use of local score approximations at low noise, without establishing exact temporal independence or directly measuring downstream planning performance.

Window	$\sigma = 0.025$	$\sigma = 0.05$	$\sigma = 0.1$	$\sigma = 0.2$
1	0.8391	0.8679	0.9317	0.9036
3	0.4587	0.3849	0.3572	0.3734
5	**0.4169**	0.3118	0.2566	0.2712
7	0.4752	**0.2988**	0.2299	0.2334
9	0.4464	0.3006	**0.2184**	**0.2037**

Table 20: Final-epoch mean squared error (↓) of the diagnostic score estimator. Bold marks the lowest error across window sizes for each noise level.

A.14 Confidence Bounds for 250-Step TFDP Results

Table 21 quantifies uncertainty for TFDP with 250 denoising steps. Basic and Shelf have 100% observed success at every robot count, but their Wilson lower bounds are 98.5%, distinguishing empirical success from guaranteed feasibility. Dense with 18 robots is the least reliable reported setting, at 96% with a $[92.7, 97.9]\%$ interval. Runtime increases with robot count on every map, while mean path length changes more modestly. Together, the estimates support high observed reliability at these scales while making the uncertainty in that conclusion explicit.

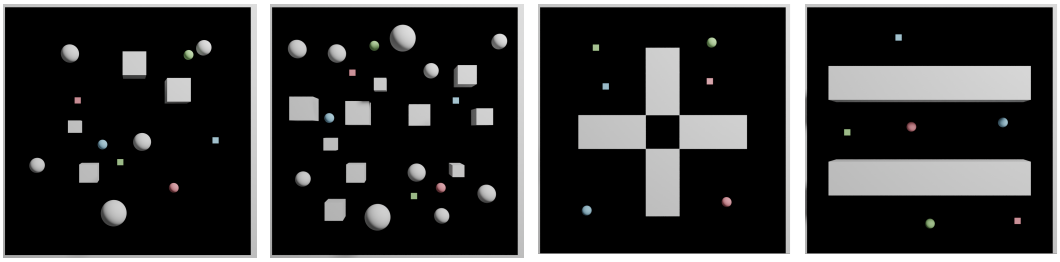
Map	Robots	S↑ (%, 95% CI)	T↓ (s)	P↓
Basic	6	100 [98.5, 100.0]	0.18 ± 0.02	1.07 ± 0.05
	12	100 [98.5, 100.0]	0.29 ± 0.08	1.08 ± 0.04
	18	100 [98.5, 100.0]	0.45 ± 0.07	1.08 ± 0.03
Dense	6	100 [98.5, 100.0]	0.35 ± 0.09	1.26 ± 0.15
	12	99 [96.8, 99.7]	0.66 ± 0.07	1.29 ± 0.09
	18	96 [92.7, 97.9]	1.06 ± 0.19	1.32 ± 0.03
Room	6	100 [98.5, 100.0]	0.43 ± 0.09	1.41 ± 0.17
	12	99 [96.8, 99.7]	0.78 ± 0.12	1.45 ± 0.08
	18	98 [95.4, 99.2]	1.24 ± 0.23	1.45 ± 0.07
Shelf	6	100 [98.5, 100.0]	0.27 ± 0.07	1.23 ± 0.16
	12	100 [98.5, 100.0]	0.51 ± 0.04	1.25 ± 0.08
	18	100 [98.5, 100.0]	0.84 ± 0.08	1.26 ± 0.04

Table 21: TFDP success with 95% Wilson intervals, runtime, and path length. Runtime and path length use mean ± 2 standard errors; gray identifies TFDP results.

Confidence intervals. Success uses 95% Wilson score intervals, which remain within $[0, 100]\%$ even near the boundaries. Runtime and path length use mean ± 2 standard errors. These summaries quantify uncertainty in the reported estimates; they do not separate variability due to sampled instances from variability due to random seeds.

A.15 Additional Planning Visualizations

Benchmark geometry. Figure 5 shows the four map types without trajectory overlays. Basic and Dense vary obstacle density, while Room and Shelf impose structured routes through openings and corridors. These layouts probe complementary difficulties: local obstacle avoidance, coordination in crowded spaces, and selection of routes around extended barriers. This distinction helps explain why the simple initializations in Appendix A.8 perform much worse on the structured maps.



(a) Basic. (b) Dense. (c) Room. (d) Shelf.

Figure 5: Example benchmarks used for our experiments.

Larger environment. Figure 6 illustrates trajectories in the large map used for the scalability experiment. The routes span open regions, dense obstacle clusters, and narrow passages, showing the geometric variety handled within one planning instance. The figure complements the scalability experiment with up to 300 robots, which reports an average runtime of 5.39 seconds on an A100 GPU. Spatial crossings in this static view do not by themselves indicate collisions, which depend on when robots traverse them.

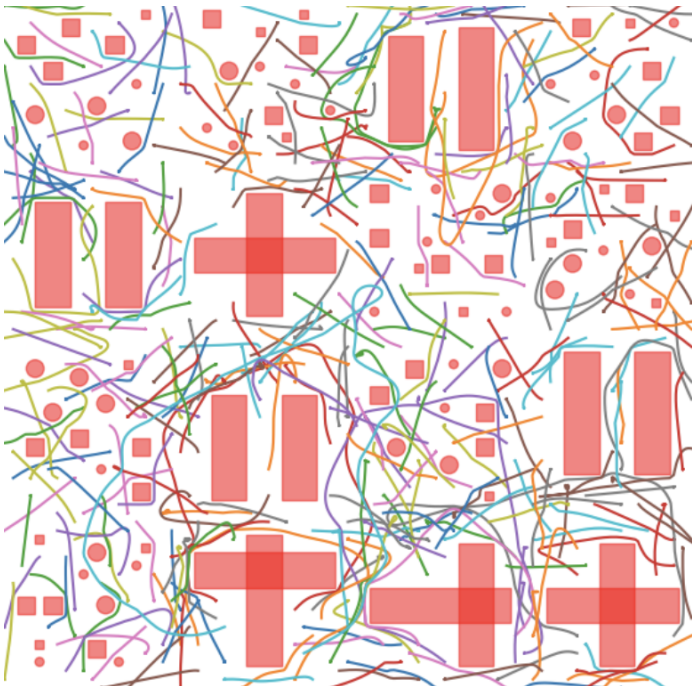


Figure 6: Generated trajectories on the larger map used for the scalability experiment. Colored curves show robot paths; red regions show obstacles.

A.16 Optimization Timelines

Figure 7 tracks the fraction of feasible plans over 250 refinement iterations on each benchmark map. TFDP uses $\eta = 0.5$; the two CHOMP-style SDF variants use $\alpha = 0.05$ with $\eta \in \{0.2, 0.5\}$. SDF guidance produces feasible plans earlier, while TFDP improves sharply during the middle of the schedule and finishes with the highest feasible fraction on all four maps. On Dense, the final fractions are approximately 0.96 for TFDP and 0.84 for both SDF variants; on Room, they are approximately 0.98, 0.86, and 0.73 for TFDP and SDF with $\eta = 0.5$ and 0.2, respectively. These curves show why final feasibility and early progress need to be assessed separately. Since the horizontal axis counts iterations, the figure describes refinement behavior rather than wall-clock efficiency.

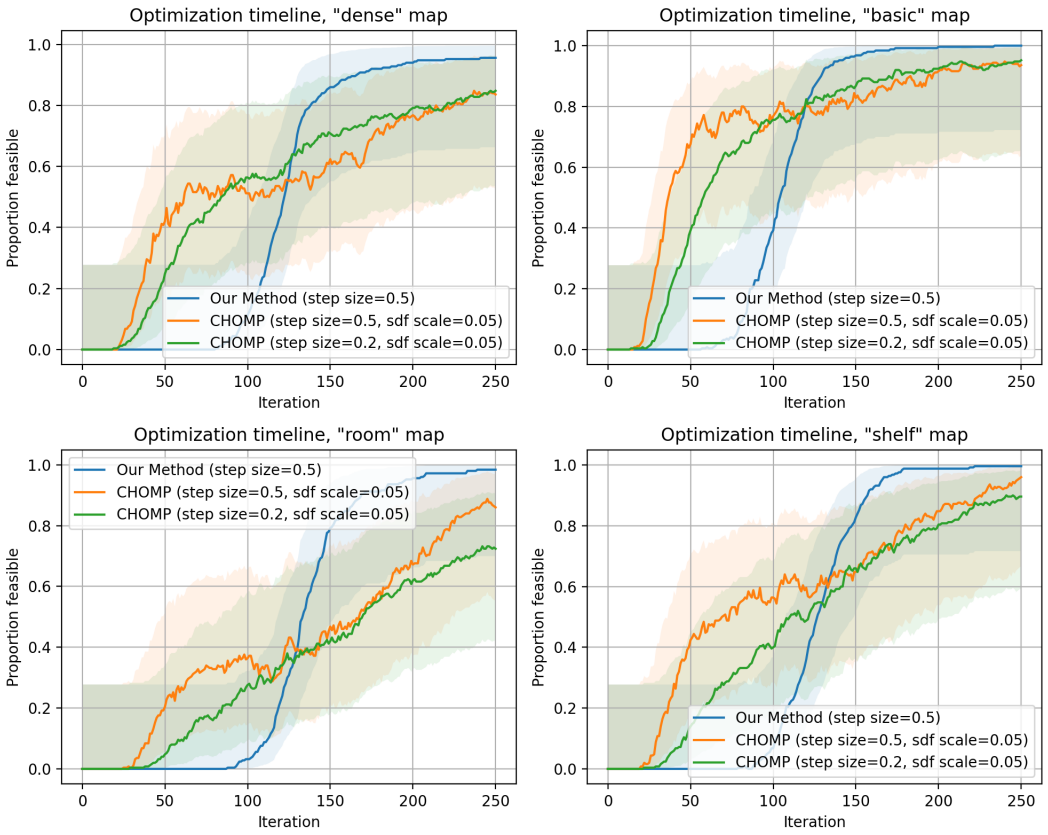


Figure 7: Fraction of feasible plans over refinement iterations on Dense (top left), Basic (top right), Room (bottom left), and Shelf (bottom right). Blue denotes TFDP; orange and green denote CHOMP-style SDF guidance at step sizes 0.5 and 0.2, respectively.