2026 IEEE International Conference on Robotics and Automation (ICRA 2026)
June 1-5, 2026. Vienna, Austria

Implicit Maximum Likelihood Estimation for Real-time Generative Model Predictive Control

Grayson Lee, Minh Bui, Shuzi Zhou, Yankai Li, Mo Chen, Ke Li

***Abstract*— Diffusion-based models have recently shown strong performance in trajectory planning, as they are capable of capturing diverse, multimodal distributions of complex behaviors. A key limitation of these models is their slow inference speed, which results from the iterative denoising process. This makes them less suitable for real-time applications such as closed-loop model predictive control (MPC), where plans must be generated quickly and adapted continuously to a changing environment. In this paper, we investigate *Implicit Maximum Likelihood Estimation* (IMLE) as an alternative generative modeling approach for planning. IMLE offers strong mode coverage while enabling inference that is two orders of magnitude faster, making it particularly well suited for real-time MPC tasks. Our results demonstrate that IMLE achieves competitive performance on standard offline reinforcement learning benchmarks compared to the standard diffusion-based planner, while substantially improving planning speed in both open-loop and closed-loop settings. We further validate IMLE in a closed-loop human navigation scenario, operating in real-time, demonstrating how it enables rapid and adaptive plan generation in dynamic environments. Real-world videos and code are available at https://gmpc-imle.github.io/.**

I. INTRODUCTION

Recent advances in generative modeling, in domains such as image, video, and language, have inspired their application to decision-making problems. In offline reinforcement learning (RL), the objective is to learn an effective policy from a fixed dataset of past interactions without additional environment sampling. One prominent approach is Reinforcement Learning via Supervised Learning (RvS) [1], [2], which re-frames policy learning as the prediction of actions (or entire action sequences) from states, often conditioned on a desired return. By framing RL problems in the form of supervised learning (SL) the goal is to be able to leverage the success and tools from SL. Generative models offer a natural fit here, as they can capture rich distributions over behaviors and support flexible conditioning on goals, returns, or other task specification.

One such instance is learning-based trajectory optimization, which directly targets the generation of high-return trajectories. Early approaches include transformer-based models [3], [4], which generate trajectories in an auto-regressive manner; however, these can suffer from compounding errors over long horizons. Diffusion-based planners, such as Diffuser [5], have been proposed to address this limitation. These methods formulate trajectory generation as a denoising process: starting from noise, they iteratively refine samples to match the distribution of high-return trajectories. Operating directly at the trajectory level improves temporal coherence and mitigates compounding errors, enabled by their ability to capture rich, multimodal trajectory distributions. However, a key drawback is the slow inference speed due to the iterative nature of the sampling process. Recent work explores hierarchical modeling [6] and policy distillation [7] to mitigate this bottleneck and make diffusion-based planning feasible for real-time or robotics applications, but introduce additional complexity and can be hard to train in practice.

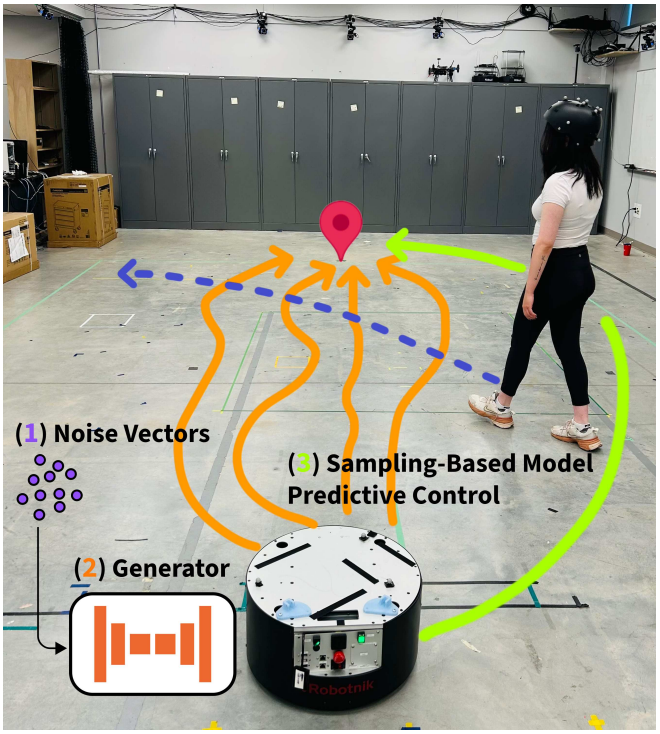

Fig. 1. **IMLE-based planning.** (1) Noise is sampled from a Gaussian distribution. (2) Our model generates a set of candidate trajectories from the sampled noise. (3) Sampling-based model predictive control is used to select and refine the top trajectory.

Implicit Maximum Likelihood Estimation (IMLE) [8]–[10] offers a promising alternative. Unlike diffusion models, IMLE generates samples in the same way as GANs [11] with a single forward pass, avoiding the computational burden of iterative denoising. In contrast to the popular GANs, which suffer mode collapse, IMLE exhibits strong mode coverage guarantees due to its loss objective. These properties make IMLE a compelling candidate for trajectory-level generative modeling and planning. In this work, we present an alternative to diffusion-based planners, which is conceptually simple and

All authors are with the Faculty of Computing Science, Simon Fraser University, Burnaby, BC, Canada.
Corresponding author: Grayson Lee, grayson1@sfu.ca

enables fast inference speed, while rivaling the performance of diffusion-based methods. Moreover, we show it is extendable to real-time applications.

- We propose a trajectory generation framework based on Implicit Maximum Likelihood Estimation (IMLE), adapted for conditional generative modeling in planning domains.
- Our approach supports single-shot trajectory sampling, avoiding the computational cost of iterative inference required by diffusion-based methods.
- We show that our method achieves competitive planning performance while significantly improving inference speed on various Mujoco benchmarks.
- We demonstrate real-time path planning of our method on a mobile robot navigating among humans in the real world, demonstrating its capacity for fast and reactive planning.

II. RELATED WORK

A. Diffusion for Planning

Diffusion models for planning [5], [12], [13] treat a trajectory analogously to an image, where the temporal horizon defines the width and the state–action dimensions define the height. The forward process gradually corrupts a clean trajectory τ^0 by adding Gaussian noise at each step:

$$q(\tau^t \mid \tau^{t-1}) = \mathcal{N}\big(\tau^t; \sqrt{\alpha_t}\,\tau^{t-1}, (1-\alpha_t)I\big)\,, \qquad (1)$$

where $\beta_t \in (0,1)$ denotes the noise variance at step t, and we define $\alpha_t = 1-\beta_t$ with cumulative product $\bar{\alpha}_t = \prod_{i=1}^{t} \alpha_i$. The closed-form forward distribution after t steps is:

$$q(\tau^t \mid \tau^0) = \mathcal{N}\big(\tau^t; \sqrt{\bar{\alpha}_t}\,\tau^0, (1-\bar{\alpha}_t)I\big)\,. \qquad (2)$$

As T increases, $\bar{\alpha}_T$ decays toward zero, and τ^T converges in distribution to an isotropic Gaussian $\mathcal{N}(0, I)$.

The reverse process is then modeled as a Gaussian denoiser,

$$p_\theta(\tau^{t-1} \mid \tau^t) = \mathcal{N}\big(\tau^{t-1}; \mu_\theta(\tau^t, t), \Sigma^t\big)\,, \qquad (3)$$

with the mean μ_θ learned by a neural network and a covariance matrix Σ^t. Prior work in planning [5], [12], [13] follow the Denoising Diffusion Probabilistic Model (DDPM) framework [14], which trains a neural network ϵ_θ to predict the injected noise by minimizing:

$$\mathcal{L}_{\text{DDPM}} = \mathbb{E}_{\tau^0,\epsilon,t}\Big[\big\|\epsilon - \epsilon_\theta\big(\sqrt{\bar{\alpha}_t}\,\tau^0 + \sqrt{1-\bar{\alpha}_t}\,\epsilon, t\big)\big\|^2\Big], \quad (4)$$

where τ^0 is a trajectory from the dataset and $\epsilon \sim \mathcal{N}(0, I)$. At inference, trajectories are generated by denoising from τ^T to τ^0, conditioned on an initial state s_0 and/or goal states. To bias generation toward high-return behaviors, classifier-based guidance [15] is applied during sampling. Thus, a limitation of this approach is the inference speed, as the denoising process requires many iterative steps, when compared with single-shot generative methods.

B. Diffusion Models for Safe Navigation

A growing body of work has explored the combination of generative models with control-theoretic tools to achieve safe decision-making in dynamic environments. One of the first examples is Safe Diffuser [13], which augments the diffusion denoising process with a CBF constraint [16], [17] which they enforce while staying close to the original diffusion step. This work established the basic principle of coupling denoising with explicit safety mechanisms.

The work most directly related to ours in safe navigation is CoBL-Diffusion [18], which integrates Control Barrier Functions (CBFs) and Control Lyapunov Functions (CLFs) into a diffusion-based planning framework. CoBL-Diffusion generates only actions which they then feed through a predefined dynamics to get, and uses CBF/CLF rewards as their guidance function for the denoising process. This ensures consistency between control inputs and resulting states, allowing the diffusion sampler to respect both safety and goal-reaching requirements in dynamic multi-agent environments. However, diffusion-based methods remain computationally expensive and struggle to achieve real-time performance due to their iterative sampling procedure.

A subsequent line of work replaces diffusion with Conditional Flow Matching (CFM) [19], where reward gradients are injected directly into the ODE dynamics to bias the generative process toward safe trajectories. This significantly reduces inference time compared to diffusion, though the procedure remains iterative, however with fewer steps. In contrast, we propose a single-shot IMLE planner that enforces mode coverage and integrates constraints directly into the loss, avoiding the iterative process altogether.

III. BACKGROUND

We follow the formulation of [5], which models entire trajectories directly with a generative model, as opposed to generating them step-by-step in an autoregressive manner, enabling global consistency and flexible conditioning at the trajectory level.

A. Problem Setting

We consider trajectory optimization in the offline setting. Let $\tau = (s_0, a_0, r_0, \ldots, s_T, a_T, r_T)$ denote a trajectory. We denote the dataset by $\mathcal{D} = \{\tau_i\}_{i=1}^{N}$, collected from prior interactions with the environment.

In optimal control theory, the main objective is to find a sequence of actions that maximizes the cumulative return for a horizon T:

$$a^*_{0:T} = \arg\max_{a_{0:T}} \sum_{t=0}^{T} r(s_t, a_t). \qquad (5)$$

Generally this optimization is challenging to be solved in high dimensions due to the curse of dimensionality and the complexity of nonlinear dynamics [20].

A common probabilistic relaxation is provided by stochastic optimal control (path-integral / KL-control) [20]–[22] and

control-as-inference (CAI) [23], [24]. For temperature $1/\beta > 0$ this yields the Gibbs distribution

$$p^*(\tau \mid s_0) \;\propto\; p_0(\tau \mid s_0)\, \exp\Big(\beta \cdot \sum_{t=0}^{T} r(s_t, a_t)\Big), \quad (6)$$

where $p_0(\tau \mid s_0)$ denotes the passive/base dynamics. This distribution is the *optimal solution to the entropy-regularized control problem*; as $\beta \to \infty$ (zero-temperature limit), it concentrates on the original argmax solution.

In practice for our problem setting, we only have access to an offline dataset $\mathcal{D}$, which provides samples from a behavior distribution that may differ significantly from $p^*(\tau \mid s_0)$. Prior work [5], [12] trains a generative model of trajectories (approximating the behavior distribution) and applies energy-based guidance at inference using a learned reward function $r_\theta(\tau) \approx \sum_{t=0}^{T} r(s_t, a_t)$. Such guidance has been shown to bias sampling toward higher-return trajectories and thereby provide a rough approximation to the CAI formulation, with the behavior distribution acting as the base [25], [26].

B. Implicit Maximum Likelihood Estimation

Implicit Maximum Likelihood Estimation (IMLE) [8]–[10] is a method for training implicit generative models by directly encouraging coverage of the data distribution. The approach uses a generator network f_θ that maps latent codes z, sampled from a standard normal distribution, to generated samples $y = f_\theta(z)$. Rather than requiring an explicit likelihood or adversarial objective, IMLE optimizes the generator to ensure that, for every data point in the training set, there exists a latent code such that the generated output is close to that data point. The method requires sampling at least as many latent codes as there are data points, i.e., $m \geq N$. To differentiate between the latent pools for unconditional and conditional IMLE, we write $\mathcal{Z} := \{z^{(j)}\}_{j=1}^{m}$ for a global latent pool and $\mathcal{Z}_i := \{z_i^{(j)}\}_{j=1}^{m}$ for a per-context pool, where $z^{(j)}, z_i^{(j)} \sim \mathcal{N}(0, I)$.

$$\min_\theta \; \mathbb{E}_{\mathcal{Z}}\Big[\sum_{i=1}^{N} \min_{z \in \mathcal{Z}} d\big(f_\theta(z),\, \tau_i\big)\Big]. \quad (7)$$

where $d(\cdot, \cdot)$ is a distance metric over trajectories and m is the total number of latent codes drawn.

In the conditional setting, Conditional IMLE (cIMLE) [27]–[29] extends this idea by training a conditional generator $f_\theta(z, c)$, adding c as a conditioning variable (e.g., the initial state or task specification). The generator maps (z, c) to a sample $\hat{\tau} = f_\theta(z, c)$, and is trained to ensure that for every ground-truth data point τ_i in the dataset, there exists some latent code z_i^* such that $f_\theta(z_i^*, c_i) \approx \tau_i$.

The cIMLE training objective is defined as:

$$\min_\theta \; \mathbb{E}_{\{\mathcal{Z}_i\}_{i=1}^{N}}\Big[\sum_{i=1}^{N} \min_{z \in \mathcal{Z}_i} d\big(f_\theta(z,\, c_i),\, \tau_i\big)\Big]. \quad (8)$$

where m is the total number of latent codes drawn. In the unconditional setting, these m samples are generated once and each data point finds its closest match among them. In conditional IMLE, we generate m samples for each conditioning context c_i.

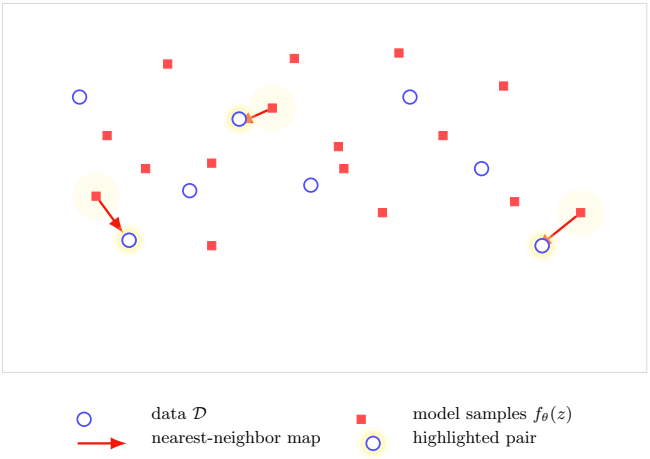

Fig. 2. Illustration of IMLE's nearest neighbor matching. The method enforces that, for every data point, there exists a generated sample in its neighborhood, ensuring that the generator covers the full data distribution.

During training, for each data point (τ_i, c_i), the model samples a fixed number of latent codes $\{z_i^{(j)}\}_{j=1}^{m}$, generates corresponding predictions $\{f_\theta(z_i^{(j)}, c_i)\}$, and selects the sample that is closest (under d) to the ground-truth trajectory τ_i as illustrated in Fig. 2. The generator parameters θ are then updated to minimize this minimum-distance loss.

IV. PLANNING WITH IMLE

We propose a generative planning framework based on Implicit Maximum Likelihood Estimation (IMLE), designed to avoid the iterative sampling process of diffusion-based methods and produce trajectory candidates in a single forward pass. By leveraging the mode-covering behavior of IMLE, our method can produce high-quality and diverse trajectory candidates for model predictive control (MPC). Moreover, we propose a way of leveraging reward weighting in the context of IMLE model training to enable effective planning even under mixed-quality data.

A. IMLE for Trajectory Generation

Given a dataset of trajectories $\{\tau_i\}_{i=1}^{N}$, our goal is to learn a conditional generative model $f_\theta(z, c)$ that maps latent codes $z \sim \mathcal{N}(0, I)$ and context c (e.g., initial state, goal state) to full trajectories. IMLE ensures that for each data point τ_i, there exists a latent code z_i^* such that the generated sample $f_\theta(z_i^*, c_i)$ closely matches the ground-truth trajectory τ_i. For our distance metric we choose ℓ^2-norm which is a standard metric for trajectory generation [5]. Formally, we minimize the following objective:

$$\mathcal{L}_{\text{IMLE}}(\theta) = \mathbb{E}_{\{\mathcal{Z}_i\}_{i=1}^{N}}\left[\sum_{i=1}^{N} \min_{z \in \mathcal{Z}_i} \|f_\theta(z, c_i) - \tau_i\|_2^2\right]. \quad (9)$$

This training paradigm ensures that the learned model covers all modes of the trajectory distribution.

Unlike diffusion models, which rely on a multi-step denoising process, IMLE learns through a direct reconstruction

loss. This objective is simpler, requiring no noise scheduling, score matching, or iterative refinement, while still enforcing coverage of the full data distribution [8]. As a result, IMLE enables efficient rollout sampling for closed-loop planning. Because IMLE does not support iterative classifier or reward guidance, we instead bias generation toward high-reward trajectories through a modified training loss.

Algorithm 1 Reward-Weighted cIMLE Training

Input: Dataset $D = \{(\tau_i, c_i, r_i)\}_{i=1}^{N}$, generator $f_\theta(z, c)$, sample factor m, epochs K, inner steps L, step size η

Precompute: $w_i \leftarrow \exp\left(\frac{r_i - \text{median}(r)}{\beta \cdot \text{MAD}(r)}\right)$ or $w_i \leftarrow \frac{r_i - r_{\min}}{r_{\max} - r_{\min}}$

for $k = 1$ to K **do**
  Sample batch $S \subseteq [N]$
  Sample latent pools $\{\mathcal{Z}_i\}_{i=1}^{N} \sim \mathcal{N}(0, I), \ \forall i \in S$
  $z_i^\star \leftarrow \arg\min_{z \in \mathcal{Z}_i} \|f_\theta(z, c_i) - \tau_i\|_2^2, \ \forall i \in S$
  for $\ell = 1$ to L **do**
    Sample mini-batch $\tilde{S} \subseteq S$
    $\theta \leftarrow \theta - \eta \nabla_\theta \frac{N}{|\tilde{S}|} \sum_{i \in \tilde{S}} w_i \|f_\theta(z_i^\star, c_i) - \tau_i\|_2^2$
  end for
end for
return θ

B. Reward-Weighted IMLE

To achieve this bias toward higher-quality trajectories, we modify the IMLE objective to approximate the CAI target distribution $p^*(\tau \mid s_0) \propto p_0(\tau \mid s_0) \exp(R(\tau))$ from Section III. Since this distribution naturally assigns higher probability to high-return trajectories, we incorporate this weighting directly into our learning objective by reweighting each training example according to its return.

Let $r_i = R(\tau_i)$ denote the return of the trajectory τ_i, where $R(\cdot)$ is the reward or cost function for a given task (e.g., environment reward in offline RL benchmarks, or CBF-based safety cost in navigation). We introduce per-sample weights w_i as a function of r_i and define the reward-weighted variant as:

$$\mathcal{L}_{\text{weighted}}(\theta) \ = \ \mathbb{E}_{\{\mathcal{Z}_i\}_{i=1}^{N}} \left[\sum_{i=1}^{N} w_i \cdot \min_{z \in \mathcal{Z}_i} \left\| f_\theta(z, c_i) - \tau_i \right\|_2^2 \right]. \tag{10}$$

a) Exponential (Boltzmann) weights: Following CAI, we use an exponential weighting with robust centering and scaling inspired by [30]:

$$w_i \ = \ \exp\left(\frac{r_i - \text{median}(r)}{\beta \cdot \text{MAD}(r)} \right),$$

where $\beta > 0$ is a temperature parameter.

b) Linear weights: We construct a simple linear weighting scheme to compare against:

$$w_i \ = \ \frac{r_i - r_{\min}}{r_{\max} - r_{\min}}.$$

This baseline is included to verify that exponential weighting gains stem from its CAI grounding rather than a generic preference for high-return samples.

C. Architecture and Conditioning

Our generative architecture follows a U-Net backbone, as commonly used in trajectory generation [5]. Diffusion-based methods typically condition trajectories via inpainting, fixing the start and goal states throughout the denoising process to enforce consistency. As our model generates trajectories in a single shot, we instead use Feature-wise Linear Modulation (FiLM) [31] for conditioning, as recently applied in diffusion-based policy learning [32]. FiLM injects the conditioning signal at each layer of the network:

$$\text{FiLM}(x; c) = \gamma(c) \cdot x + \beta(c),$$

where $\gamma(c)$ and $\beta(c)$ are produced by MLPs applied to the conditioning input c.

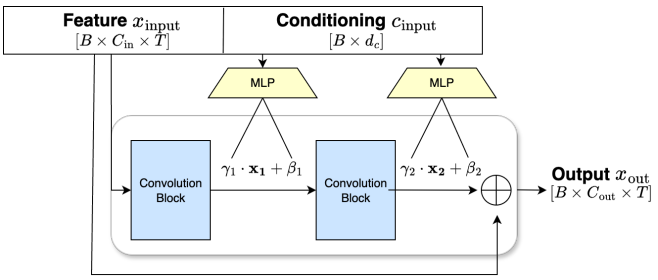

Fig. 3. FiLM-conditioned block used throughout the U-Net. Two small MLPs conditioning c to scale $\gamma(c)$ and shift $\beta(c)$, which modulate the two blocks via FiLM ($x \mapsto \gamma(c) \odot x + \beta(c)$).

This architecture ensures that the conditioning signals (initial and goal states) are available at all spatial-temporal scales of the network.

D. Applications in Model Predictive Control

We integrate our IMLE planner into different sampling-based model predictive control (MPC) frameworks to highlight its versatility across domains. In all cases, IMLE provides diverse candidate trajectories in a single forward pass, which we treat as a learned base distribution.

a) Score-Ranked MPC: On offline reinforcement learning benchmarks [33] (e.g., D4RL locomotion), we follow the score-and-ranking sampling-based MPC procedure used in prior work [5]. At each step, the planner generates a batch of candidate trajectories. These are evaluated with a learned reward function, and the top-ranked trajectory is executed. This setup ensures fairness against diffusion-based planners while emphasizing the computational efficiency of IMLE.

b) Model Predictive Path Integral (MPPI): Due to the dynamic nature of pedestrian navigation, we integrate IMLE within the Model Predictive Path Integral (MPPI) framework [22]. IMLE replaces the standard Gaussian proposal distribution with a learned multi-modal trajectory generator, producing structured rollouts under tight latency constraints.

The control objective combines a CBF safety penalty (with adjustable radius at inference) and a CLF goal-progress term [18], along with a temporally discounted penalty on deviations from the previous plan to reduce oscillations from mode switching, since IMLE generates diverse trajectories.

V. EXPERIMENTS

We assess the proposed IMLE-based framework across a diverse set of domains, spanning both offline reinforcement learning and real-time control tasks. Our goals are twofold: (i) to measure performance on established offline RL benchmarks, and (ii) to validate responsiveness in a dynamic, real-time environment.

Our experiments are organized into two categories. In *Offline Reinforcement Learning*, we benchmark on the D4RL MuJoCo locomotion suite [33] to study return maximization under various data distributions, and on Maze2D to test long-horizon planning under sparse rewards. Following the setup described in Section IV-D-a, we use a score-ranked MPC for locomotion tasks. For Maze2D, where demonstrations are already successful, we omit the ranking and simply execute a sampled trajectory in an open-loop evaluation. We compare against Diffuser [5], which provides a natural baseline, given we use the same UNet architecture and its role as the standard diffusion-based trajectory planner. We measure the sampling frequency on a CPU (AMD EPYC 9655 - Zen 5) and on a GPU (2/7th of NVIDIA H100 as a MIG instance), with a 20 GB memory allocation. All experiments are repeated over 150 random seeds to ensure statistical robustness and align with prior works.

In *Real-Time Navigation*, we evaluate both open-loop simulation and closed-loop deployment on a mobile robot in pedestrian environments. Our planner embeds IMLE within MPPI using the control cost described in Section IV-D-b.

All models are trained on ETH [34] and UCY [35] pedestrian trajectories processed with TrajData [36], using horizon 20 and discretization $0.4\,\mathrm{s}$. Safe behavior is encouraged via reward weighting with a Control Barrier Function (CBF) under a conservative $1\,\mathrm{m}$ safety constraint, and we evaluate collision radii of 0.5 and $0.7\,\mathrm{m}$. Because few trajectories satisfy this constraint, we augment the dataset through translation, rotation, and smoothing. Pedestrians are modeled as constant-velocity obstacles, and collisions are evaluated against ground-truth trajectories.

For simulation, we train on the augmented ETH dataset and evaluate on 500 UCY scenes. IMLE-MPPI is compared against Gaussian MPPI [22] (warm-started with a straight-line trajectory), diffusion-based planners [18], and flow matching methods. Metrics include collision rate (any collision within radius r), goal error (final distance to the goal), smoothness (maximum per-step change in velocity), and jerk (mean magnitude of the second finite difference of velocity).

For real-world experiments, we train on the combined augmented ETH and UCY datasets and deploy the policy on a mobile robot. Diffusion-based planners are excluded due to the latency of iterative denoising, which prevents real-time inference, so we focus on IMLE-based trajectory proposals within MPPI.

Although pedestrian datasets do not perfectly match robot dynamics, incorporating robot dynamics into reward weighting or MPPI refinement is left for future work.

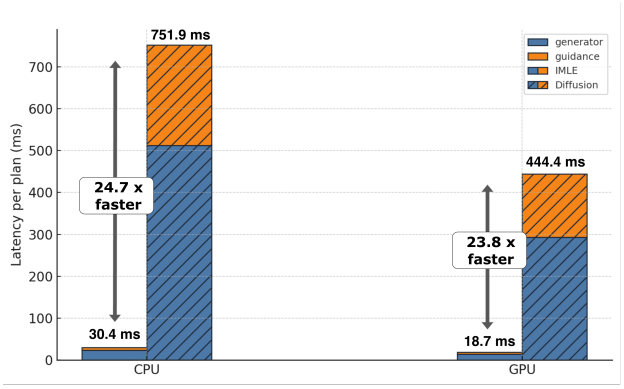

Fig. 4. **IMLE vs Diffusion.** Median per-plan latency (ms) split into generator and guidance on CPU/GPU, averaged over Walker/Hopper/HalfCheetah (Batch Size 64)

A. Offline Reinforcement Learning

1) Mujoco: We compare our IMLE-based approach with Diffuser [5], using the same model architecture and experimental setup to provide a controlled comparison. Our method achieves competitive performance across most environments while sidestepping the iterative denoising process required by diffusion models (Table I). We additionally break down the per-plan latency of each planner (Fig. 4), separating generator and guidance times, since diffusion requires forward passes through both the generator and learned reward function that provides the guidance signal at each denoising step.

Dataset	Environment	Diffuser	IMLE+Exp RW
Medium-Expert	HalfCheetah	88.9 ± 0.3	**91.9** ± 0.09
	Hopper	103.3 ± 1.3	**104.2** ± 3.81
	Walker2d	106.9 ± 0.2	**107.9** ± 0.40
Medium	HalfCheetah	42.8 ± 0.3	**43.1** ± 0.29
	Hopper	74.3 ± 1.4	**85.0** ± 4.02
	Walker2d	**79.6** ± 0.55	78.3 ± 2.75
Medium-Replay	HalfCheetah	37.7 ± 0.5	**39.5** ± 0.59
	Hopper	**93.6** ± 0.4	85.0 ± 4.02
	Walker2d	**70.6** ± 1.6	69.7 ± 3.84
Average		77.5	**78.47**
Sampling Frequency on CPU (Hz)		1.33	**32.87**
Sampling Frequency on GPU (Hz)		2.25	**53.52**

TABLE I

PERFORMANCE COMPARISON ACROSS MUJOCO LOCOMOTION DATASETS. (BATCH SIZE 64)

a) Reward Weighting Impact: We analyze how reward weighting affects performance across datasets with varying trajectory quality (Table II). As expected, Medium-Expert shows modest gains since expert trajectories already bias toward high returns. The largest improvements occur in Medium-Replay and Medium datasets, where suboptimal behavior policies create low-return trajectory distributions. The superior performance of exponential weighting aligns with CAI theory, where optimal distributions have a Boltzmann form.

Dataset	Environment	No RW	Lin RW	Exp RW
Medium-Expert	HalfCheetah	87.8 ± 0.94	90.6 ± 0.26	**91.9** ± 0.09
	Hopper	102.7 ± 4.07	**111.2** ± 0.38	104.2 ± 3.81
	Walker2d	108.8 ± 0.05	**109.0** ± 0.04	107.9 ± 0.40
Medium	HalfCheetah	35.5 ± 0.86	**44.0** ± 0.09	43.1 ± 0.29
	Hopper	71.1 ± 5.39	65.0 ± 4.33	**85.0** ± 4.02
	Walker2d	74.4 ± 3.23	**81.6** ± 0.58	78.3 ± 2.75
Medium-Replay	HalfCheetah	39.0 ± 0.62	38.0 ± 0.69	**39.5** ± 0.59
	Hopper	85.4 ± 5.51	75.9 ± 6.37	**85.0** ± 4.02
	Walker2d	21.8 ± 6.57	30.5 ± 8.89	**69.7** ± 3.84
Average		66.9	71.6	**78.47**

TABLE II

PERFORMANCE COMPARISON OF REWARD-WEIGHTED IMLE VARIANTS.

2) Maze2D: A key advantage of diffusion-based planners over their policy variants is their ability to perform well in sparse reward settings [37]. We observe that our IMLE-based planner achieves comparable performance on Maze2D, while providing a substantial speedup (Table III). Since all trajectories in this dataset are goal-reaching by construction, no reward weighting is necessary.

Dataset	Environment	Diffuser	IMLE
Single Task	U-Maze	113.9 ± 3.1	**124.8** ± 0.65
	Medium	**121.5** ± 2.7	117.3 ± 3.53
	Large	123.0 ± 6.4	**129.2** ± 4.89
Average		119.5	**123.7**
Multi Task	U-Maze	128.9 ± 1.8	**132.3** ± 0.97
	Medium	127.2 ± 3.4	**127.8** ± 2.60
	Large	132.1 ± 5.8	**137.1** ± 4.41
Average		129.4	**132.4**
Sampling Frequency on CPU (Hz)		0.96	**114.63**
Sampling Frequency on GPU (Hz)		1.37	**101.28**

TABLE III

PERFORMANCE COMPARISON ACROSS MAZE2D DATASETS. (BATCH SIZE 1)

We additionally implemented our method in JAX, which achieves higher GPU sampling throughput than PyTorch (87.56 vs. 53.52 Hz on Locomotion and 133.98 vs. 101.28 Hz on Maze2D).

B. Real-Time Navigation

1) Simulation: We compare with CoBL [18], a DDIM sampler with 50 sampling steps, and an adapted Conditional Flow Matching (CFM) model [38] using 9 ODE integration steps. Both generative baselines employ the same guidance function with 10 guidance iterations per sampling step.

With a collision radius of $0.5\,\mathrm{m}$, IMLE and IMLE+MPPI yield smoother, lower-jerk trajectories, consistent with their ability to capture realistic motion patterns from the dataset (Table IV). Because UCY pedestrians are well approximated by constant-velocity motion, objective-based planners (MPPI, CoBL, and CFM) more readily satisfy short-horizon safety constraints. Warm-starting MPPI with IMLE proposals improves constraint satisfaction while preserving trajectory smoothness.

Metric	MPPI	CoBL	CFM	IMLE	IMLE+MPPI
Collision Radius = 0.5 m					
Collision Rate (%) ↓	10.00	10.20	5.80	9.80	**4.60**
Goal Error (m) ↓	0.521	**0.050**	0.431	0.181	0.360
Smoothness (m/s) ↓	0.772	0.606	0.623	0.397	**0.394**
Jerk (m/s^3) ↓	1.746	1.407	0.812	**0.458**	0.479
Collision Radius = 0.7 m					
Collision Rate (%) ↓	**16.40**	22.20	28.40	20.60	—
Goal Error (m) ↓	0.570	**0.048**	0.073	0.186	—
Smoothness (m/s) ↓	0.760	0.605	0.612	**0.394**	—
Jerk (m/s^3) ↓	1.781	1.396	0.740	**0.461**	—
Sampling Frequency on CPU (Hz) ↑	**125.00**	0.41	2.65	76.92	52.63
Sampling Frequency on GPU (Hz) ↑	**142.86**	0.71	4.30	111.11	83.33

TABLE IV

PLANNER PERFORMANCE ACROSS UCY SCENES. (BATCH SIZE 64)

At $0.7\,\mathrm{m}$, the mismatch between the safety constraint and the training distribution increases, as few trajectories maintain this level of clearance (Table IV). In this regime, generative models are most effective as proposal distributions for downstream optimization.

Metric	Line	CoBL	CFM	IMLE
Collision Radius = 0.7 m				
Collision Rate (%) ↓	16.40	**8.80**	11.40	10.20
Goal Error (m) ↓	0.570	0.431	0.447	**0.396**
Smoothness (m/s) ↓	0.760	0.760	0.613	**0.394**
Jerk (m/s^3) ↓	1.746	0.759	0.787	**0.484**
Sampling Frequency on CPU (Hz) ↑	**125.00**	0.41	2.55	52.63
Sampling Frequency on GPU (Hz) ↑	**142.86**	0.71	4.30	83.33

TABLE V

PERFORMANCE COMPARISON BETWEEN DIFFERENT WARM-START STRATEGIES FOR MPPI ACROSS UCY SCENES. (BATCH SIZE 64)

Warm-start strategies for MPPI show that while generative models alone often violate safety margins, using them as proposal distributions significantly improves performance (Table V). In particular, IMLE provides high-quality trajectories while maintaining real-time sampling.

2) Mobile Robot: We deploy the planner onboard a mobile robot with a high-frequency low-level controller, replanning at up to 50 Hz on the onboard CPU using a batch of 8 trajectories per step. We evaluate real-world navigation with one to four pedestrians moving freely in a shared indoor environment with unknown goals, observing only past positions. The planner continuously updates its trajectory distribution to maintain collision avoidance while progressing toward the goal.

VI. CONCLUSIONS AND FUTURE WORK

We present a generative planning framework that adapts IMLE for real-time MPC. By combining conditional IMLE with reward weighting, it enables single-shot trajectory generation with mode coverage while biasing toward high-return behaviors. Across offline RL benchmarks and real-time navigation tasks, IMLE achieves competitive performance with orders-of-magnitude faster inference than diffusion-based planners.

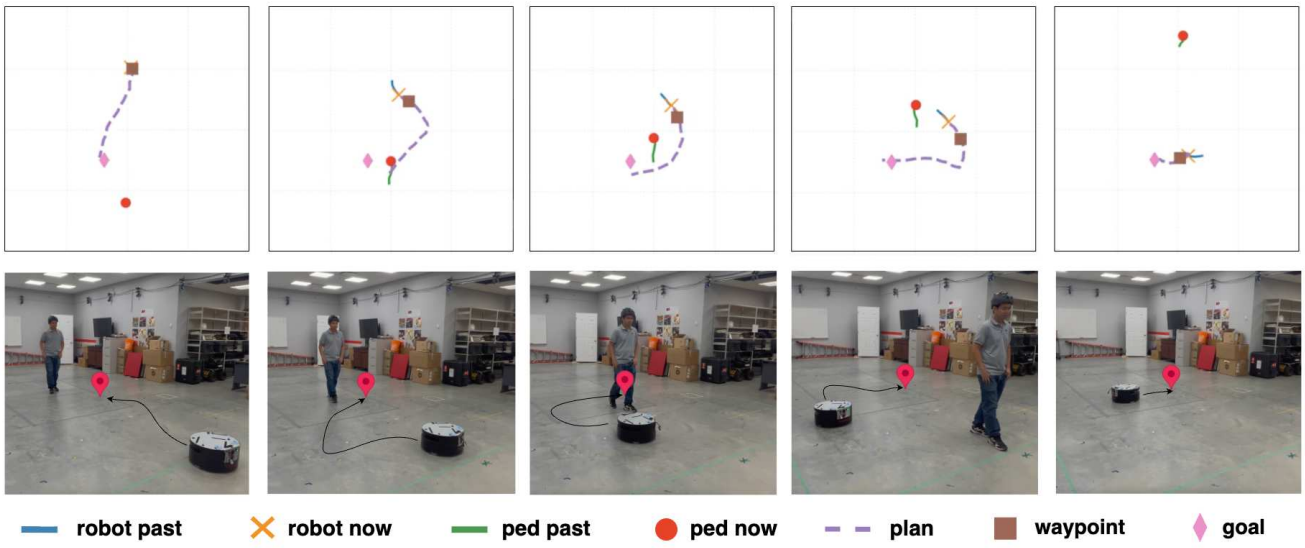

Fig. 5. Using IMLE-generated plans at 50 Hz, the robot reaches the goal while avoiding collisions. The top row shows the conditioning variables (robot and pedestrian past and current states). We also plot the generated plan and the waypoint tracked by the low-level controller.

Several directions remain. First, evaluating IMLE in data-scarce regimes, where diffusion models often struggle due to the isotropic Gaussian forward process [10], while IMLE has shown stronger data efficiency in policy learning [29]. Second, leveraging IMLE's coverage for uncertainty-aware planning in dynamic environments, where diverse proposals can improve downstream optimization and safety filtering [39]. Finally, like other generative models, IMLE degrades when optimal trajectories lie outside the training distribution; adaptive mechanisms [40] may improve robustness beyond simple data augmentation.

ACKNOWLEDGMENT

This research was enabled in part by support from NSERC, the BC DRI Group, and the Digital Research Alliance of Canada. The authors thank Chirag Vashist and Shichong Peng for insightful discussions on IMLE, and Kurtis Yang for assistance with real-world experimental setup.

REFERENCES

[1] S. Emmons, B. Eysenbach, I. Kostrikov, and S. Levine, "Rvs: What is essential for offline RL via supervised learning?" In *International Conference on Learning Representations*, 2022.

[2] J. Schmidhuber, "Reinforcement learning upside down: Don't predict rewards–just map them to actions," *arXiv preprint arXiv:1912.02875*, 2019.

[3] L. Chen, K. Lu, A. Rajeswaran, *et al.*, "Decision transformer: Reinforcement learning via sequence modeling," *Advances in Neural Information Processing Systems*, vol. 34, pp. 15 084–15 097, 2021.

[4] M. Janner, Q. Li, and S. Levine, "Offline reinforcement learning as one big sequence modeling problem," in *Advances in Neural Information Processing Systems*, 2021.

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

[6] Z. Dong, J. Hao, Y. Yuan, *et al.*, "Diffuserlite: Towards real-time diffusion planning," *Advances in Neural Information Processing Systems*, vol. 37, pp. 122 556–122 583, 2024.

[7] H. Lu, Y. Shen, D. Li, J. Xing, and D. Han, "Habitizing diffusion planning for efficient and effective decision making," in *Forty-second International Conference on Machine Learning*, 2025.

[8] K. Li and J. Malik, *Implicit maximum likelihood estimation*, 2018. arXiv: 1809.09087 [cs.LG]. [Online]. Available: https://arxiv.org/abs/1809.09087.

[9] M. Aghabozorgi, S. Peng, and K. Li, "Adaptive imle for few-shot pretraining-free generative modelling," in *International Conference on Machine Learning*, 2023. [Online]. Available: https://api.semanticscholar.org/CorpusID:260810286.

[10] C. Vashist, S. Peng, and K. Li, "Rejection sampling imle: Designing priors for better few-shot image synthesis," in *European Conference on Computer Vision*, Springer, 2024, pp. 441–456.

[11] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, *et al.*, "Generative adversarial nets," *Advances in Neural Information Processing Systems*, vol. 27, 2014.

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

[13] W. Xiao, T.-H. Wang, C. Gan, R. Hasani, M. Lechner, and D. Rus, "Safediffuser: Safe planning with diffusion probabilistic models," in *The Thirteenth International Conference on Learning Representations*, 2023.

[14] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," *Advances in Neural Information Processing Systems*, vol. 33, pp. 6840–6851, 2020.

[15] P. Dhariwal and A. Nichol, "Diffusion models beat gans on image synthesis," *Advances in neural information processing systems*, vol. 34, pp. 8780–8794, 2021.

[16] Q. Nguyen and K. Sreenath, "Exponential control barrier functions for enforcing high relative-degree safety-critical constraints," in *2016 American Control Conference (ACC)*, IEEE, 2016, pp. 322–328.

[17] A. D. Ames, S. Coogan, M. Egerstedt, G. Notomista, K. Sreenath, and P. Tabuada, "Control barrier functions: Theory and applications," in *2019 18th European Control Conference (ECC)*, 2019, pp. 3420–3431. DOI: 10.23919/ECC.2019.8796030.

[18] K. Mizuta and K. Leung, "Cobl-diffusion: Diffusion-based conditional robot planning in dynamic environments using control barrier and lyapunov functions," in *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2024, pp. 13 801–13 808.

[19] K. Mizuta and K. Leung, "Unified generation-refinement planning: Bridging flow matching and sampling-based mpc," *arXiv preprint arXiv:2508.01192*, 2025.

[20] H. J. Kappen, "Path integrals and symmetry breaking for optimal control theory," *Journal of statistical mechanics: theory and experiment*, vol. 2005, no. 11, P11011, 2005.

[21] E. Todorov, "Compositionality of optimal control laws," *Advances in Neural Information Processing Systems*, vol. 22, 2009.

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

[23] B. D. Ziebart, *Modeling purposeful adaptive behavior with the principle of maximum causal entropy*. Carnegie Mellon University, 2010.

[24] S. Levine, "Reinforcement learning and control as probabilistic inference: Tutorial and review," *arXiv preprint arXiv:1805.00909*, 2018.

[25] C. Lu, H. Chen, J. Chen, H. Su, C. Li, and J. Zhu, "Contrastive energy prediction for exact energy-guided diffusion sampling in offline reinforcement learning," in *International Conference on Machine Learning*, PMLR, 2023, pp. 22 825–22 855.

[26] R. Feng, C. Yu, W. Deng, P. Hu, and T. Wu, "On the guidance of flow matching," in *Forty-second International Conference on Machine Learning*, 2025.

[27] K. Li, S. Peng, T. Zhang, and J. Malik, "Multimodal image synthesis with conditional implicit maximum likelihood estimation," *International Journal of Computer Vision*, vol. 128, no. 10, pp. 2607–2628, 2020.

[28] S. Peng, S. A. Moazenipourasil, and K. Li, "Chimle: Conditional hierarchical imle for multimodal conditional image synthesis," *Advances in Neural Information Processing Systems*, vol. 35, pp. 280–296, 2022.

[29] K. Rana, R. Lee, D. Pershouse, and N. Suenderhauf, "Imle policy: Fast and sample efficient visuomotor policy learning via implicit maximum likelihood estimation," in *Proceedings of Robotics: Science and Systems (RSS)*, 2025.

[30] X. B. Peng, A. Kumar, G. Zhang, and S. Levine, "Advantage-weighted regression: Simple and scalable off-policy reinforcement learning," *arXiv preprint arXiv:1910.00177*, 2019.

[31] E. Perez, F. Strub, H. De Vries, V. Dumoulin, and A. Courville, "Film: Visual reasoning with a general conditioning layer," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 32, 2018.

[32] C. Chi, Z. Xu, S. Feng, *et al.*, "Diffusion policy: Visuomotor policy learning via action diffusion," *The International Journal of Robotics Research*, p. 02 783 649 241 273 668, 2023.

[33] J. Fu, A. Kumar, O. Nachum, G. Tucker, and S. Levine, "D4rl: Datasets for deep data-driven reinforcement learning," *arXiv preprint arXiv:2004.07219*, 2020.

[34] S. Pellegrini, A. Ess, K. Schindler, and L. Van Gool, "You'll never walk alone: Modeling social behavior for multi-target tracking," in *2009 IEEE 12th International Conference on Computer Vision*, 2009, pp. 261–268.

[35] A. Lerner, Y. Chrysanthou, and D. Lischinski, "Crowds by example," *Computer Graphics Forum*, vol. 26, no. 3, pp. 655–664, 2007.

[36] B. Ivanovic, G. Song, I. Gilitschenski, and M. Pavone, "trajdata: A unified interface to multiple human trajectory datasets," in *Proceedings of the Neural Information Processing Systems (NeurIPS) Track on Datasets and Benchmarks*, New Orleans, USA, Dec. 2023.

[37] H. Lu, D. Han, Y. Shen, and D. Li, "What makes a good diffusion planner for decision making?" In *The Thirteenth International Conference on Learning Representations*, 2025.

[38] Y. Lipman, R. T. Chen, H. Ben-Hamu, M. Nickel, and M. Le, "Flow matching for generative modeling," *arXiv preprint arXiv:2210.02747*, 2022.

[39] A. Wagenmaker, P. Dong, R. Tsao, C. Finn, and S. Levine, "Posterior behavioral cloning: Pretraining bc policies for efficient rl finetuning," *arXiv preprint arXiv:2512.16911*, 2025.

[40] Z. Liang, Y. Mu, M. Ding, F. Ni, M. Tomizuka, and P. Luo, "Adaptdiffuser: Diffusion models as adaptive self-evolving planners," in *International Conference on Machine Learning*, PMLR, 2023, pp. 20 725–20 745.