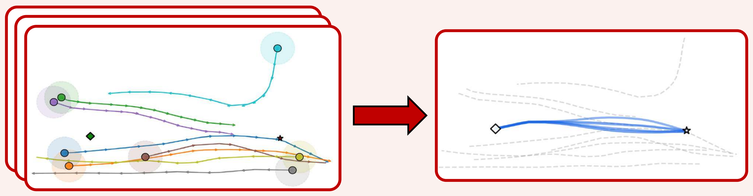
Implicit Maximum Likelihood Estimation for Real-time Generative Model Predictive Control

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

Department of Computing Science, Simon Fraser University

Task: Learning Trajectory Distributions from Data

Learn a **trajectory prior** from offline demonstrations to support real-time planning and control. Let $\tau = (s_0, a_0, r_0, \dots, s_T, a_T, r_T)$ denote a trajectory, and $\mathcal{D} = \{\tau_i\}_{i=1}^N$, a dataset by collected from previous environment interaction. Our goal is to learn the distribution of trajectories $p_\theta(\tau \mid s_0, s_H)$, conditioned on the start and goal state.

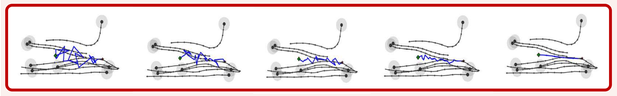


Above is an aerial view of pedestrian motion. In this scenario the model is trained on a collection of such scenes to learn typical movement patterns, allowing it to generate plausible future trajectories in crowded environments.

Background & Motivation: From Diffusion to IMLE

Control requires generative models with **High Quality Trajectories** **(no mode averaging)** and **Diverse Multi-Modal Behaviours** **(mode coverage)**

Diffusion models generate samples by progressively denoising, requiring many iterative steps for high-fidelity results.



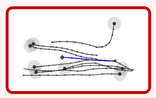
When the task gets difficult

1. Increase **model size**
2. Increase **diffusion steps**

T× forward passes through the model.

We need fast, one-step generation

IMLE generates in **one step**!



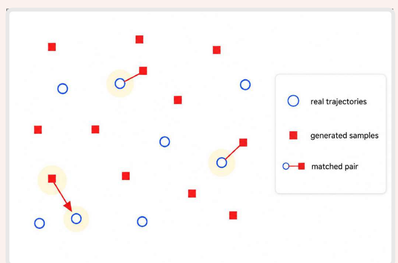
Standard IMLE Objective:

$$\min_\theta \mathbb{E}_{z_1,\dots,z_m}\left[\sum_i \min_j d\left(\tau_i, f_\theta(z_j)\right)\right]$$

Intuition: Every **real trajectory** should have at least one nearby **generated trajectory**.

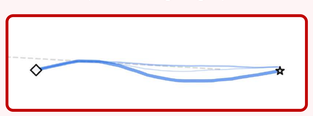
Generation becomes simple sampling:

$$\{z_1, \dots, z_m\} \sim \mathcal{N}(0, I)$$

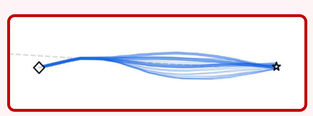


Diverse trajectory distributions are required for robust planning

Mode Collapse
(Learns to go right)



Mode Coverage
(Learns both left and right)



Method: Reward Weighted IMLE

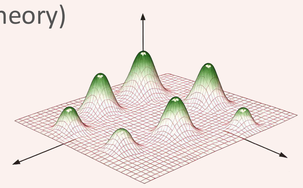
In offline reinforcement learning and control, high-reward trajectories are more important than uniformly modeling the entire dataset distribution.

1 Maximum Entropy Control (Theory)

Entropy-regularized control yields a Boltzmann distribution over trajectories.

$$p^*(\tau \mid s_0) = \left(\frac{1}{Z}\right) p_0(\tau \mid s_0) \exp\left(\beta R(\tau)\right)$$

↑ Prior / dynamics distribution ↑ Exponential in return (reward)



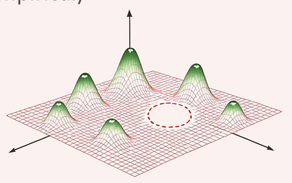
High-return trajectories are exponentially more likely.

2 Offline Dataset Approximation (Empirical)

We only observe dataset trajectories from an empirical distribution $p_D(\tau \mid s_0)$.

$$\tilde{p}_D(\tau \mid s_0) \propto p_D(\tau \mid s_0) \exp\left(\beta \cdot \sum_{t=0}^{T} r(s_t, a_t)\right)$$

↑ Empirical dataset distribution ↑ Exponential Reward Tilting



Exponentially tilt the offline trajectory distribution toward high return.

3 Conditional IMLE Training (with Reward Weighting)

High-return trajectories receive exponentially larger importance during IMLE training.

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

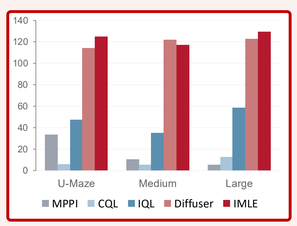
Where: $\boldsymbol{w_i} = \exp\left(\frac{r_i - \text{median}(r)}{\beta \cdot \text{MAD}(r)}\right)$ ← Normalized Advantage

*MAD – Maximum Absolute Deviation

Simulation Experiments: D4RL Planning and Control

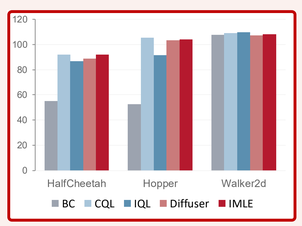
We evaluate its performance in sparse-reward planning and continuous control

Planning Task **(Maze2D)**



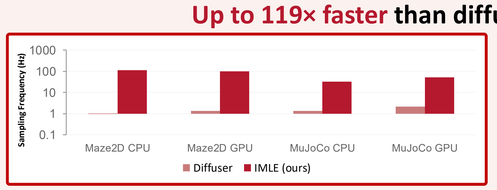
Sparse rewards: IMLE matches Diffuser; both far outperform RL and MPPI.

Control Task **(MuJoCo medium-expert)**



Continuous control: IMLE is competitive with strong RL and Diffuser baselines.

Up to 119× faster than diffusion

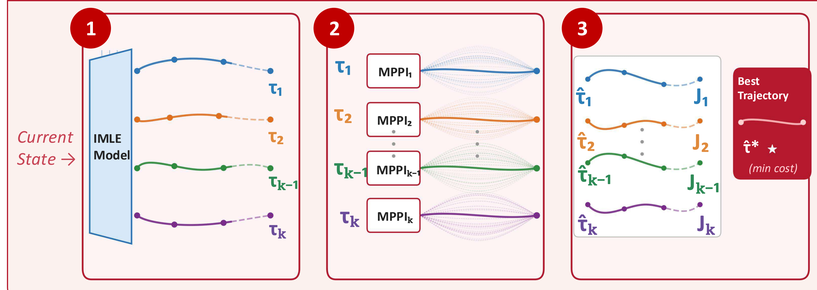


119× Maze2D CPU **74×** Maze2D GPU

25× MuJoCo CPU **24×** MuJoCo GPU

Setup: AMD EPYC 9655 Zen 5 (CPU) · NVIDIA H100 2/7 MIG, 20 GB (GPU) · D4RL benchmark · 150 random seeds

Planning: IMLE-Guided MPPI



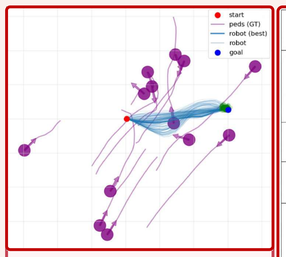
IMLE proposes k diverse candidates in one shot; MPPI refines each in parallel; we pick the best.

Deployment: Zero-Shot Social Navigation

Trained in simulation (ETH) → Evaluated on novel environment (UCY)

Reward function: Barrier-inspired safety term (0.5 m safe radius) and distance-to-goal objective.

ETH/UCY Qualitative Result (Top Down)



We also increase the constraint **[0.5→ 0.7 m]** at test-time to evaluate performance under tighter constraints.

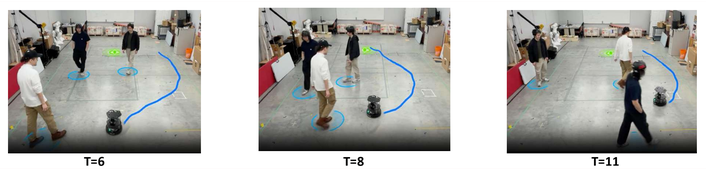
ETH/UCY Benchmark Results

Metric	Collision Radius = 0.5 m (Trained CBF)				
	MPPI	CoBL	CFM	IMLE	IMLE + MPPI (Ours)
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/s³) ↓	1.746	1.407	0.812	**0.458**	0.479
Sampling Freq. CPU (Hz) ↑	**125.00**	0.41	2.65	76.92	52.63
Sampling Freq. GPU (Hz) ↑	**142.86**	0.71	4.30	111.11	83.33

Metric	Collision Radius = 0.7 m (All methods use MPPI refinement)			
	MPPI	CoBL + MPPI	CFM + MPPI	IMLE + MPPI (Ours)
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/s³) ↓	1.746	0.759	0.787	**0.484**
Sampling Freq. CPU (Hz) ↑	**125.00**	0.41	2.55	52.63
Sampling Freq. GPU (Hz) ↑	**142.86**	0.71	4.30	83.33

Trained on simulation (ETH/UCY) → Deployed in real world

IMLE + MPPI on multiple platforms at 50 Hz



Key Takeaways

- **Real-time generative control** with **single-pass** trajectory sampling
- **Orders-of-magnitude** faster inference than diffusion-based planners
- **Diverse, mode-covering** trajectories for more robust decision making
- Demonstrated on **offline RL benchmarks** and **real-world human navigation**

More Results & Real-World Videos:
https://gmpc-imle.github.io/