DATA SCIENCE AND ARTIFICIAL INTELLIGENCE
UNIVERSITÀ DEGLI STUDI DI TRIESTE

Using Diffusion Models To Solve Planning And Trajectory Optimization Problems

PROBABILISTIC MACHINE LEARNING

Students
Andrea Tonello
Giacomo Serafini

Professor
Luca Bortolussi

February 10th , 2026

THE PROBLEM

- **Model-Based RL** typically separates modeling from planning: learn a dynamics model f(s,a), then plug it into a trajectory optimizer
  → shortcomings: trajectory optimizers often exploit model inaccuracies, producing adversarial-looking plans rather than optimal ones
- **Model-Free Alternatives** (value functions, policy gradients) avoid this but lose the benefits of planning, mainly long-horizon reasoning, and are more sample-inefficient

→ close this gap, so that sampling from the model and planning with it become nearly identical

We try to implement this solution using Diffusion models, taking the paper Planning with Diffusion for Flexible Behavior Synthesis as inspiration

2D Maze Solving Problem, using a Conditional Diffusion Model

- Train a diffusion model on expert trajectories for a given **horizon H** to learn the distribution of valid paths

- At evaluation time, generate collision-free trajectories starting from noise, given arbitrary *start* and *goal*, through **inpainting**:
  - Start / Goal constraints
  - Model fills in the trajectory

- Are generated trajectories valid / good / optimal?

- Generalization: can the model plan to start/goal pairs not seen during training?

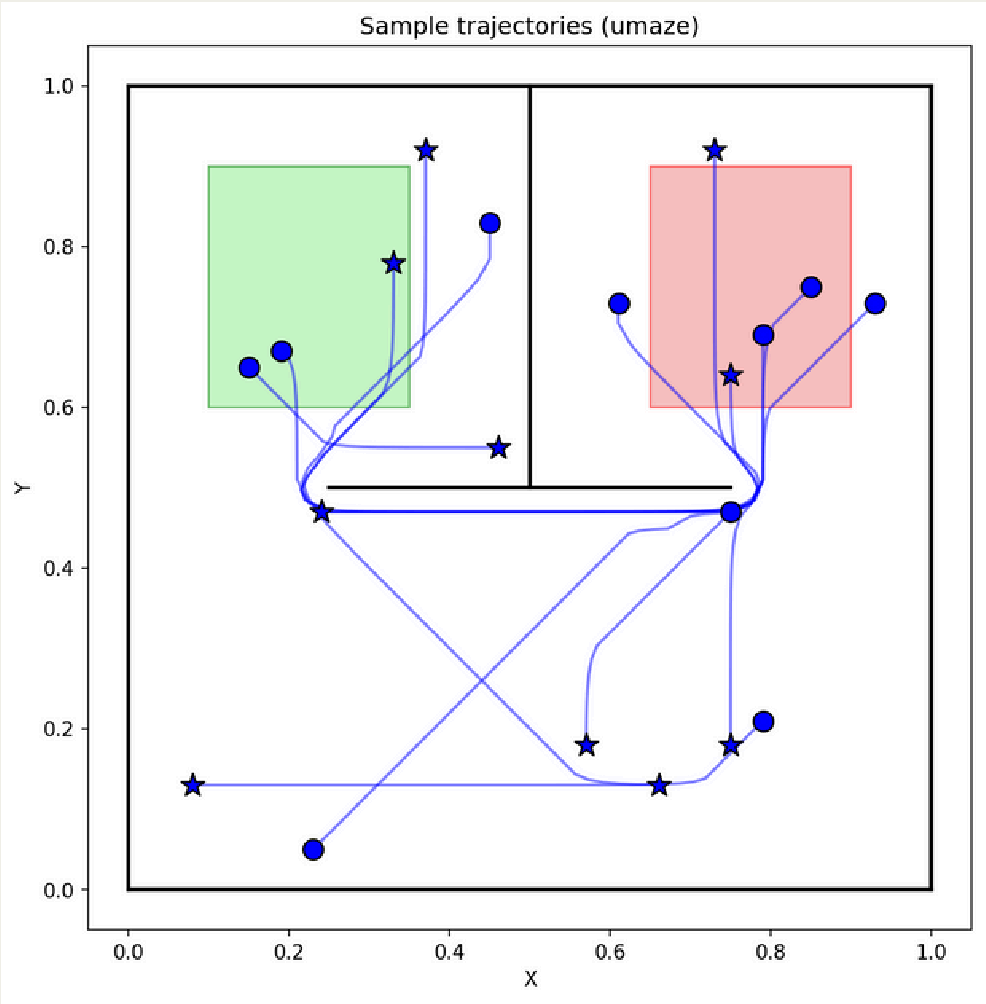
DATASET

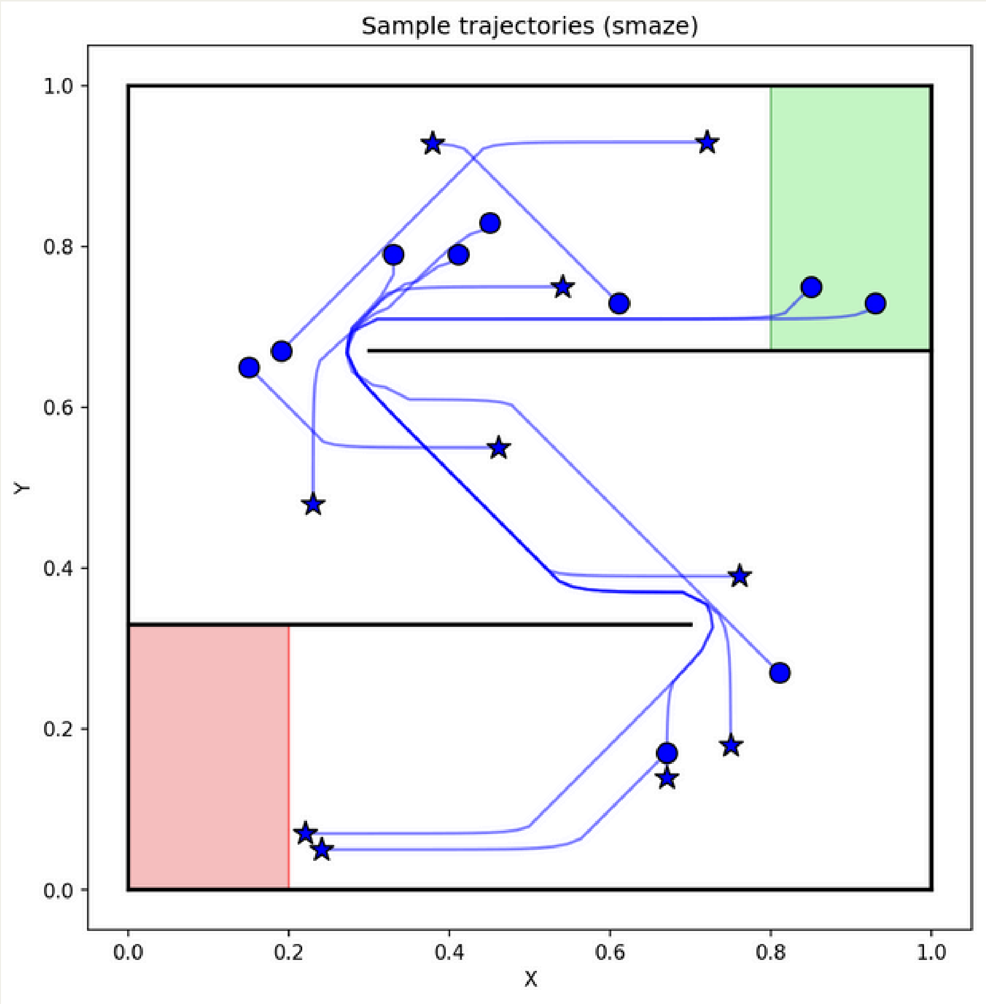
Two different maze layouts, U-maze and S-maze. For each,

- Dataset consisting of 50000 expert trajectories (A*) of random start/goal pairs

- Each trajectory is represented by a 4 × H vector
  - 4: (state, action) transitions → ((x, y) , (dx, dy))
  - H (=128): planning horizon, i.e. number of timesteps

$$\tau = \begin{bmatrix} s_0 & s_1 & \dots & s_H \\ a_0 & a_1 & \dots & a_H \end{bmatrix}$$





NOTATION DETAIL

There are two different notions of "time":

- **Planning horizon h** in {0, 1, . . . , H}: the position within the trajectory
- **Diffusion timestep t** in {0, 1, . . . , T}: which step of the denoising process we are at

MODEL - FORWARD PROCESS: NOISE SCHEDULER

Sampling process:

- $q(x_t|x_0) = \mathcal{N}\left(x_t;\ \sqrt{\overline{\alpha}_t}x_0,\ (1-\overline{\alpha}_t)I\right),$ $\qquad \overline{\alpha}_t = \prod_{i=1}^{t} 1-\beta_i$
- Sample x_t given x_0 : $\quad x_t = \sqrt{\overline{\alpha}_t}x_0 + \sqrt{1-\overline{\alpha}_t}\varepsilon$

Cosine noise scheduler: slower start, smoother transitions. Computes $\overline{\alpha}_t$ directly.

- $f(t) = \cos^2\left(\frac{t/T+s}{1+s}\cdot\frac{\pi}{2}\right),$ $\qquad s$ small offset to avoid initial stalling
- $\overline{\alpha}_t = \frac{f(t)}{f(0)} \quad \Longrightarrow \quad \beta_t = 1 - \frac{\overline{\alpha}_t}{\overline{\alpha}_{t-1}}$

MODEL - REVERSE PROCESS

The true posterior $q(x_{t-1}|x_t, x_0)$ has mean:

$$\tilde{\mu}_t(x_t, x_0) = \frac{\beta_t \cdot \sqrt{\overline{\alpha}_{t-1}}}{1 - \overline{\alpha}_t} \cdot x_0 + \frac{(1 - \overline{\alpha}_{t-1}) \cdot \sqrt{\overline{\alpha}_t}}{1 - \overline{\alpha}_t} \cdot x_t$$

We don't know x_0 at inference, but $x_0 = \dfrac{x_t - \sqrt{1 - \overline{\alpha}_t} \cdot \varepsilon}{\sqrt{\overline{\alpha}_t}}$

So we can rewrite

$$\mu_\theta(x_t, \varepsilon_\theta) = \frac{1}{\sqrt{1 - \beta_t}} \left(x_t - \frac{\beta_t}{\sqrt{1 - \overline{\alpha}_t}} \cdot \varepsilon_\theta \right)$$

MODEL - REVERSE PROCESS

The reverse process then samples x_{t-1} up until x_0 from the gaussian:

$$p_\theta(x_{t-1}|x_t) = \mathcal{N}(\mu_\theta(x_t, \varepsilon_\theta), \tilde{\sigma}_t^2 \cdot I)$$

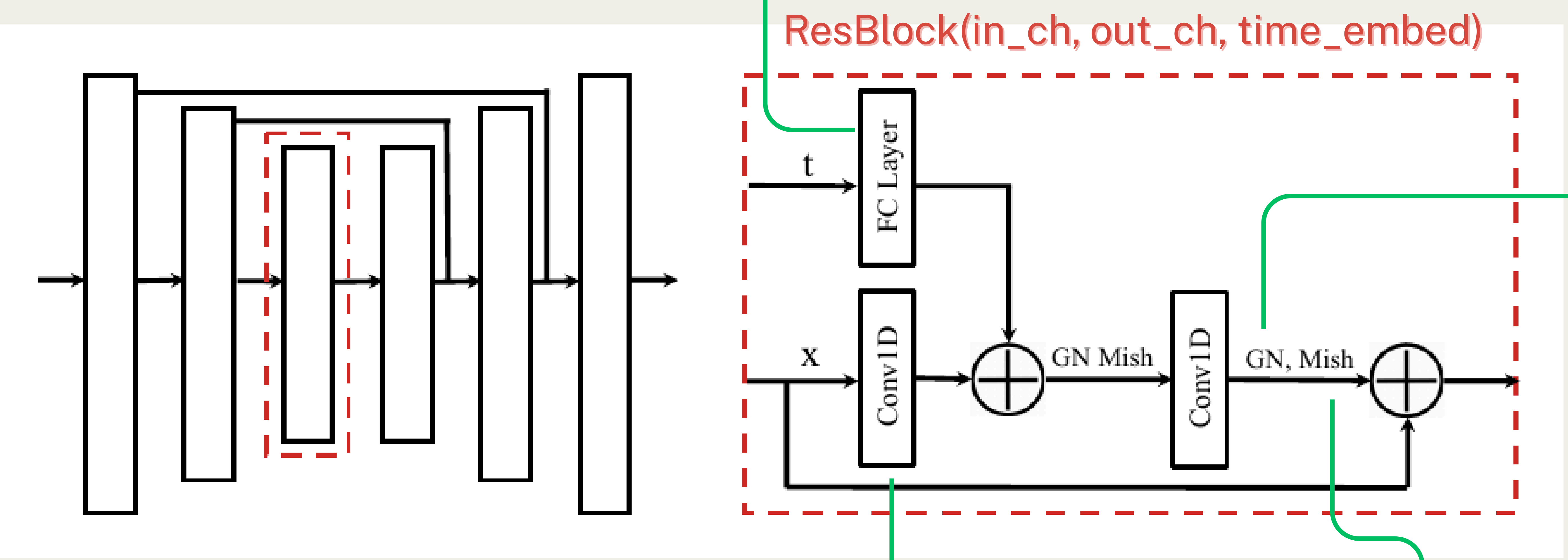
$$x_{t-1} = \frac{1}{\sqrt{\alpha_t}}\left(x_t - \frac{1-\alpha_t}{\sqrt{1-\overline{\alpha}_t}} \cdot \varepsilon_\theta\right) + \tilde{\sigma}_t z\,, \qquad z = \begin{cases} \mathcal{N}(0, \mathbf{I}) & t > 1 \\ 0 & t = 1 \end{cases}$$

With $\tilde{\sigma}_t^2 = \dfrac{1-\overline{\alpha}_{t-1}}{1-\overline{\alpha}_t}\beta_t$

MODEL - REVERSE PROCESS: TEMPORAL U-NET

GN instead of BN because during planning we have a different batch size

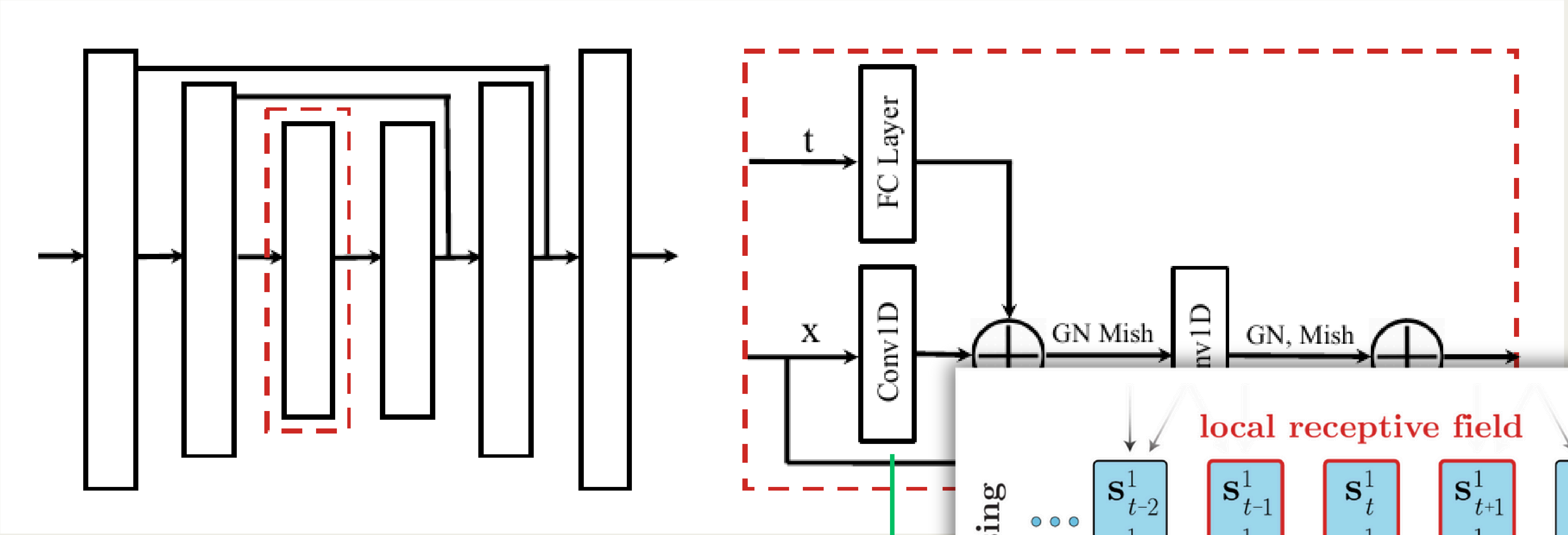
Inject diffusion timestep **t** information

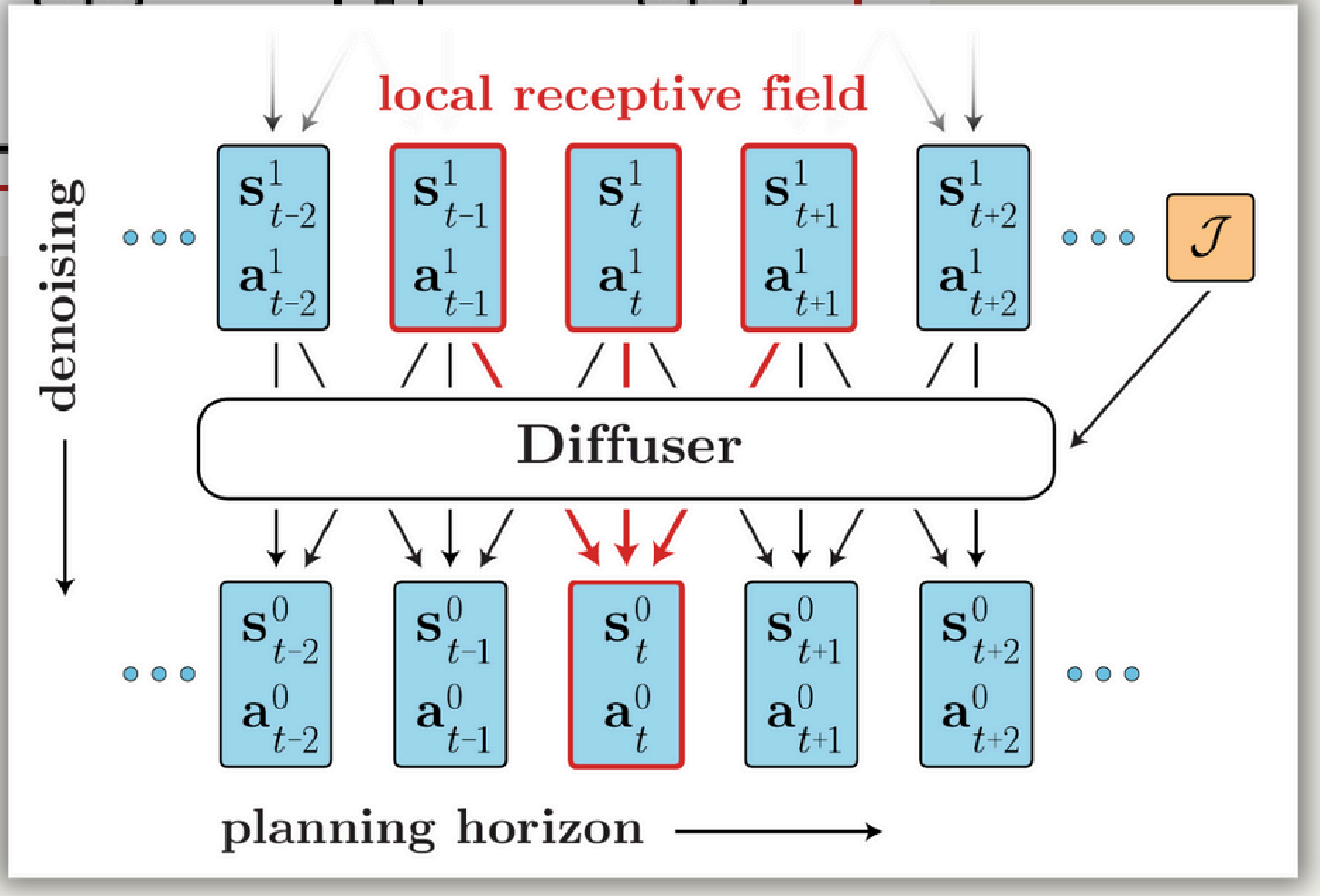


The 1D conv. slides a kernel of size 5 across the horizon axis **H**, giving temporal locality

$$\mathrm{Mish}(x) = x \cdot \tanh(\mathrm{softplus}(x))$$

MODEL - REVERSE PROCESS: TEMPORAL U-NET





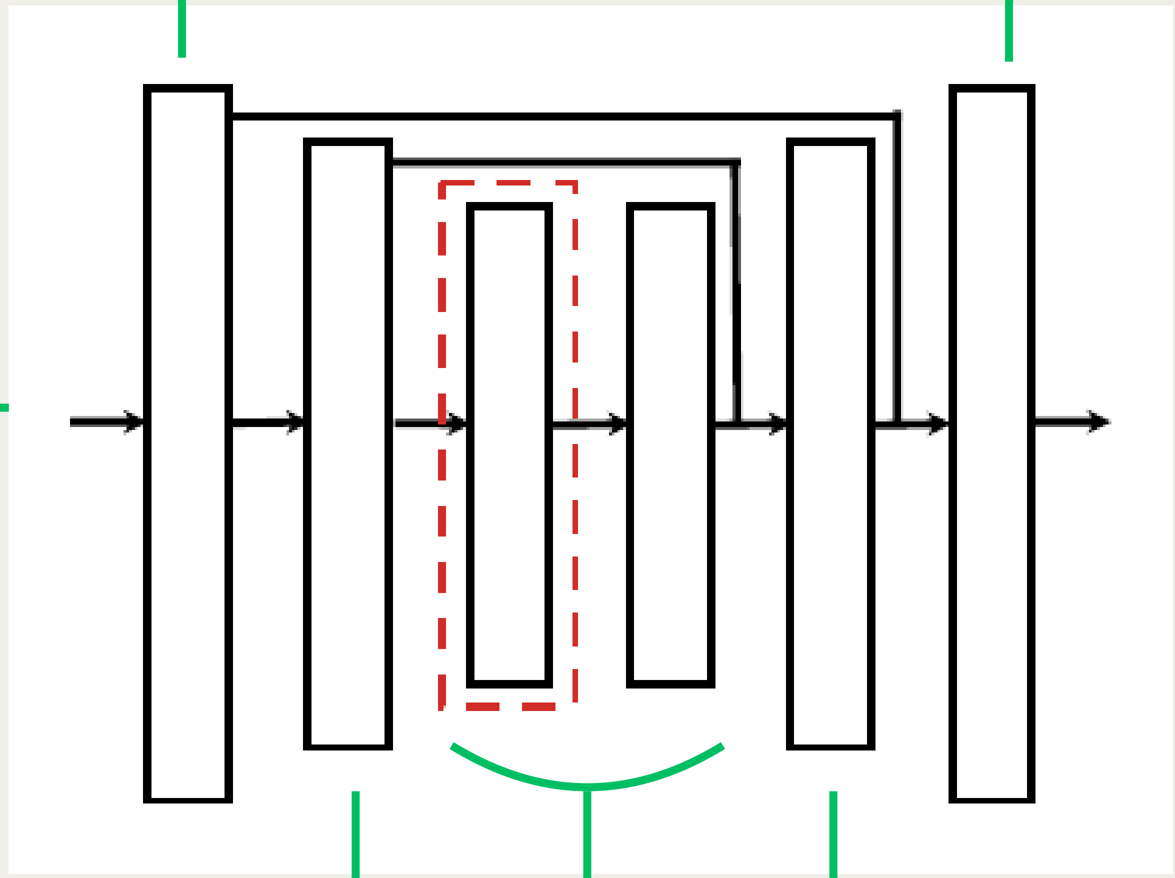
The 1D conv. slides a kernel of size 5 across the horizon axis **H**, giving temporal locality

MODEL - REVERSE PROCESS: TEMPORAL U-NET

- ResBlock(32, 64, 128)
- ResBlock(64, 64, 128)
- Skip
- Downsample - halves H

- ResBlock(128, 32, 128)
- ResBlock(32, 32, 128)

Initial proj: 4 → 32



Final proj: 32 → 4

- ResBlock(64, 128, 128)
- ResBlock(128, 128, 128)
- Skip
- No downsample

- ResBlock(128, 128, 128)
- ResBlock(128, 128, 128)

- ResBlock(256, 64, 128)
- ResBlock(64, 64, 128)
- Upsample - doubles H

MODEL - INFERENCE: PLANNING

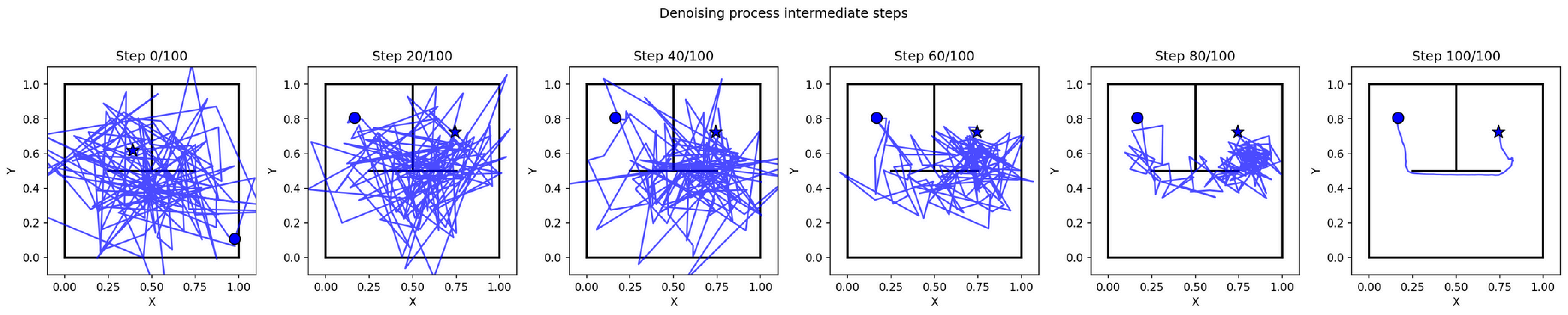
The model learned the unconditional distribution of valid trajectories during training.

At inference, **inpainting** forces start/goal to specified values at every diffusion step, and the model fills in the middle consistently with those constraints

Instead of just forcing start and goal (h = 0, h = H), we **soft inpaint the first and last eight steps**:

h	Outcome
0	Full constraint
1	87.5% constraint, 12.5% model output
2	75% constraint, 25% model output
3	62.5% constraint, 37.5% model output
...	...
8	Full model output

DIFFUSION PROCESS VISUALIZATION



Step 0 happens before denoising and inpainting constraints are applied, thus the inconsistent start and goal positions

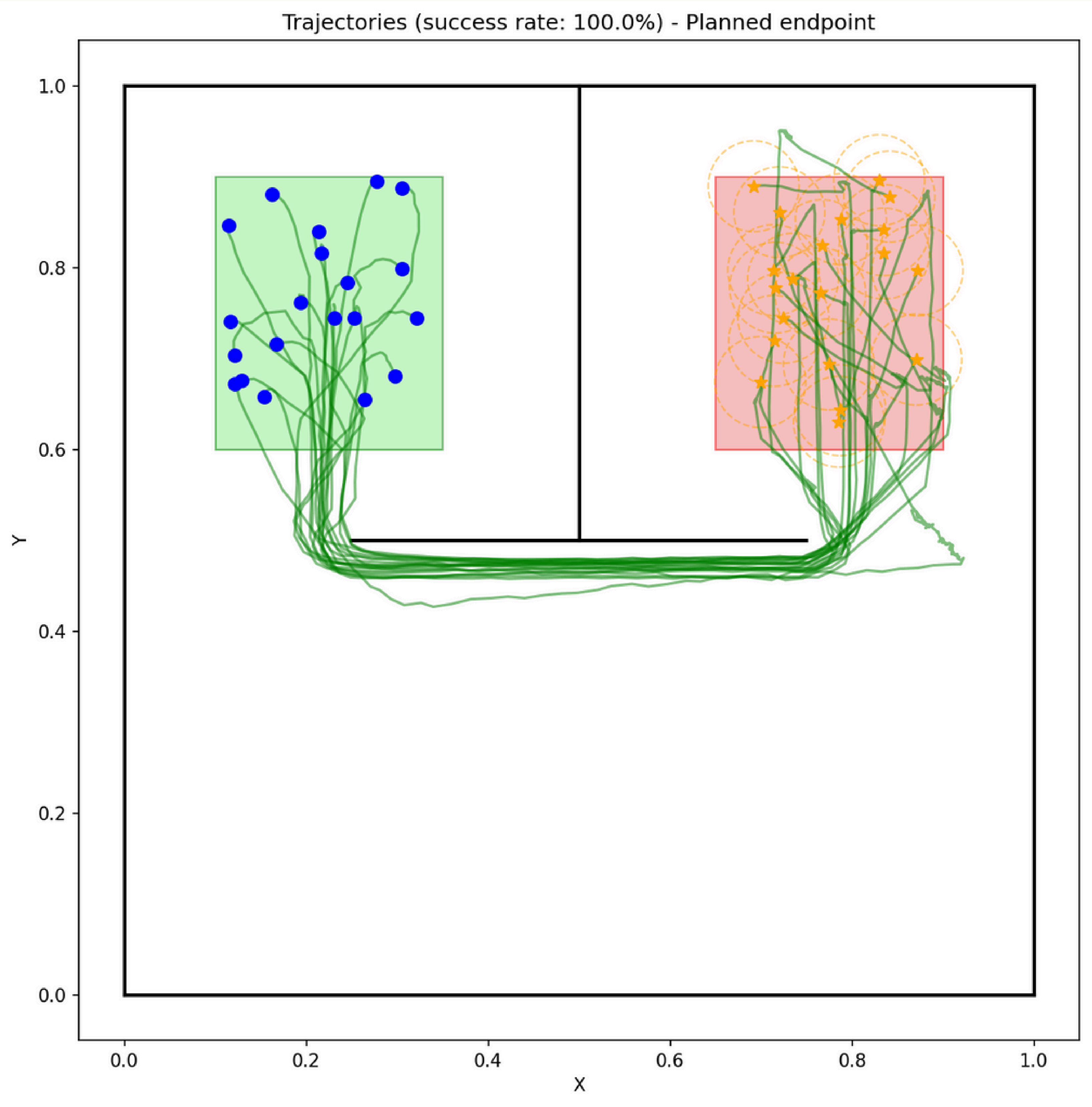
TRAINING DETAILS

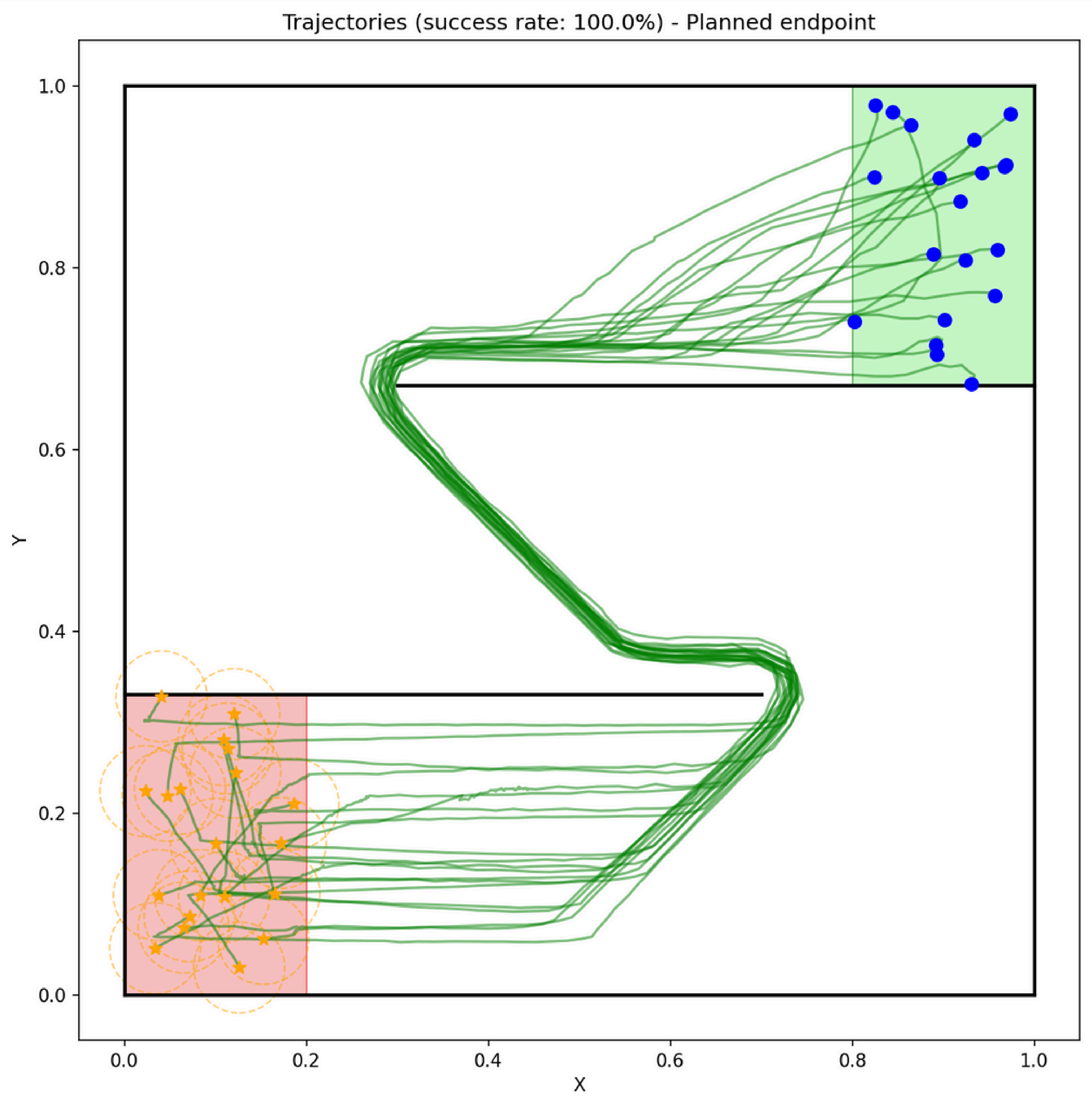
- Loss function: MSE, $\mathcal{L} = \mathbb{E}_t, x_0, \varepsilon_t \left[\parallel \varepsilon_t - \varepsilon_\theta(x_t, t) \parallel^2 \right]$

- Epochs : 200
- Batch size: 32
- Optimizer: Adam
- Learning rate: 2e-4

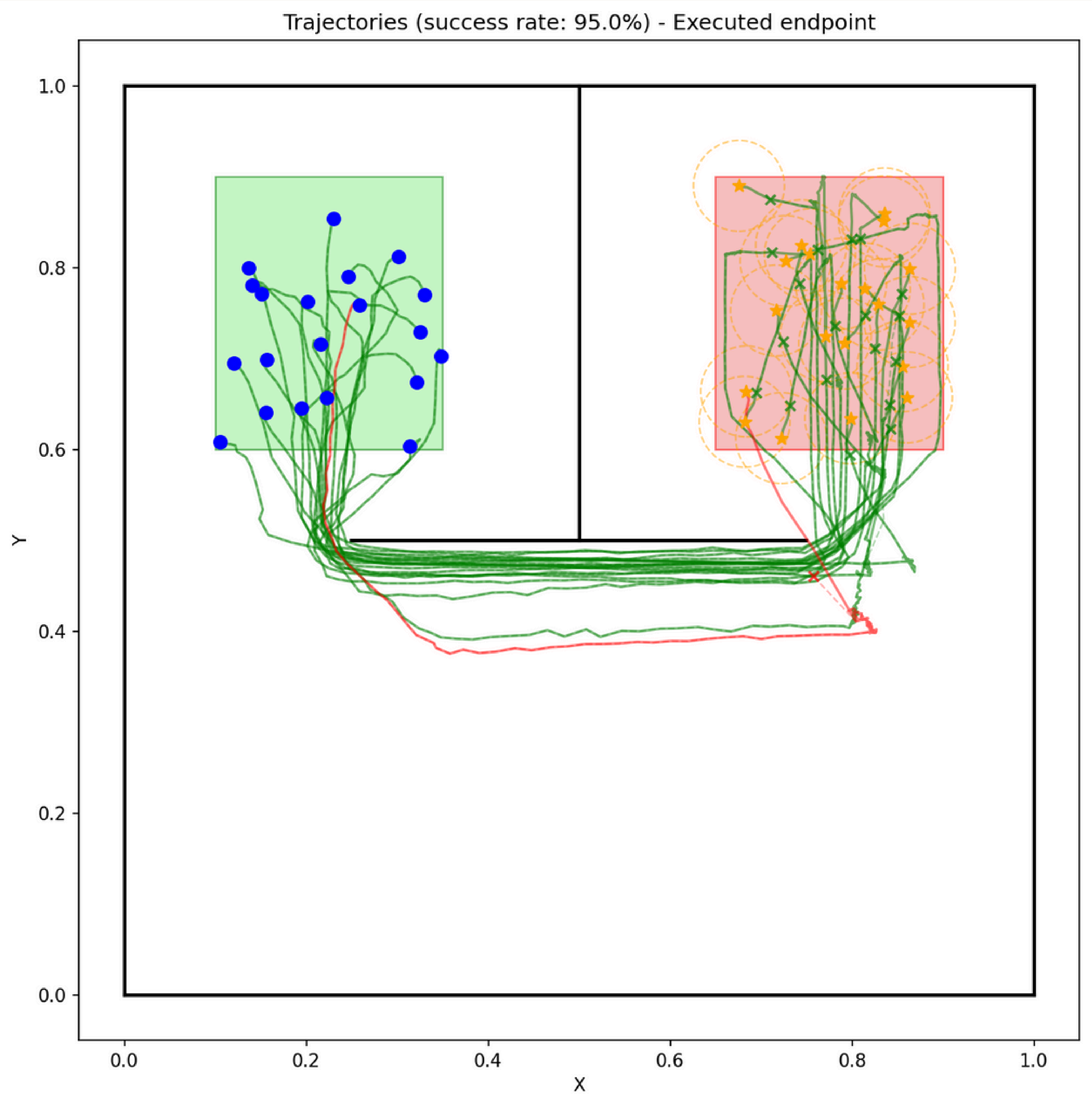
- Dataset: 50000 samples
- Diffusion steps T: 100
- Horizon H: 128

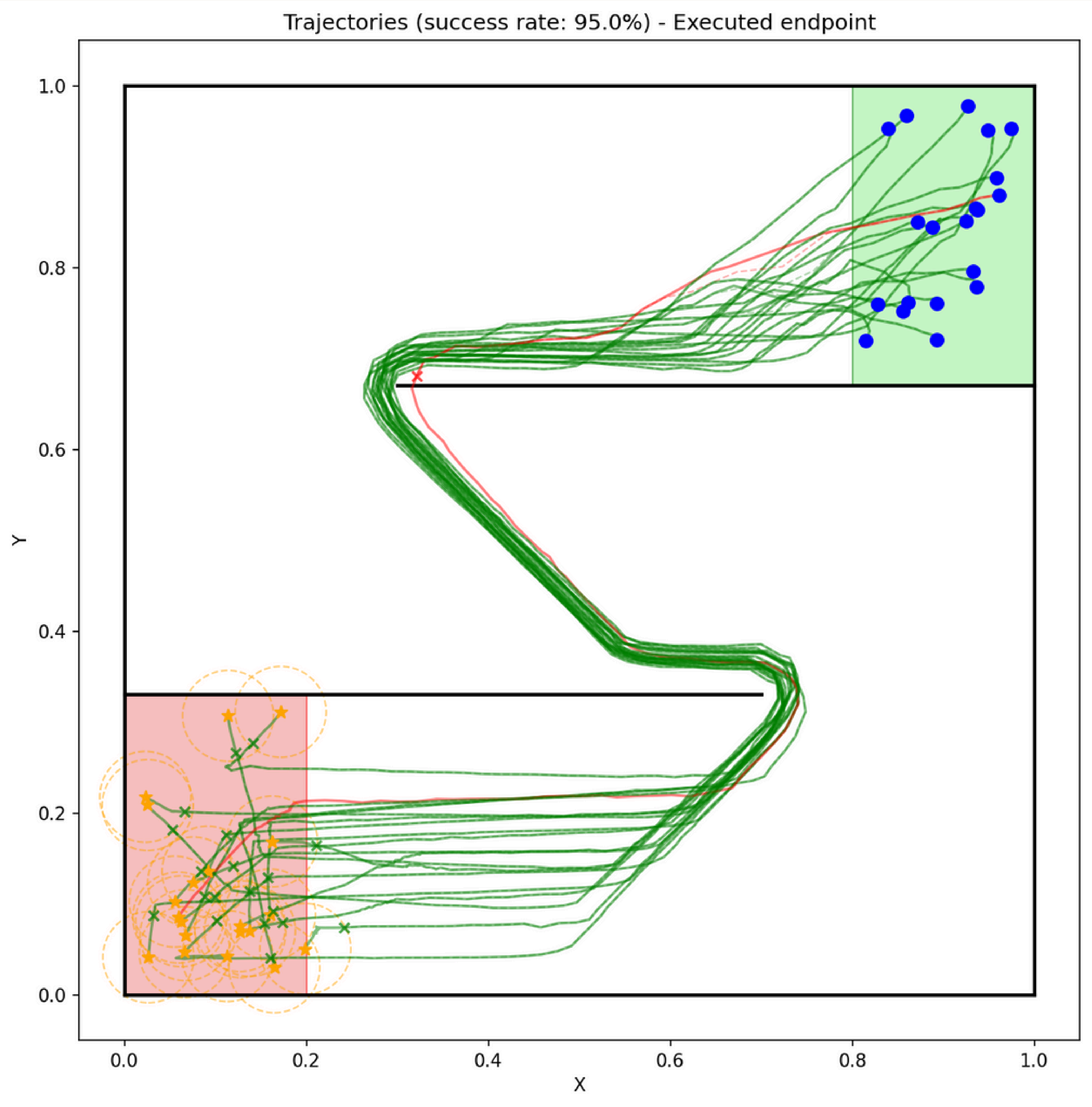
RESULTS - PLANNED TRAJECTORIES



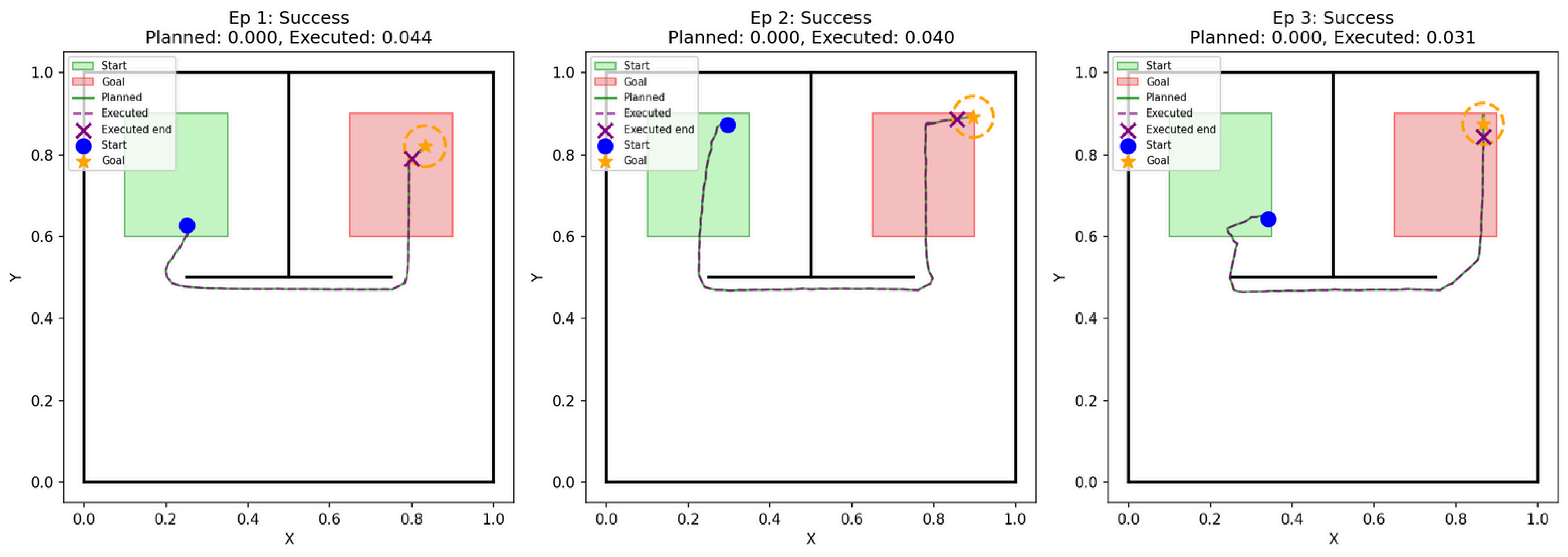


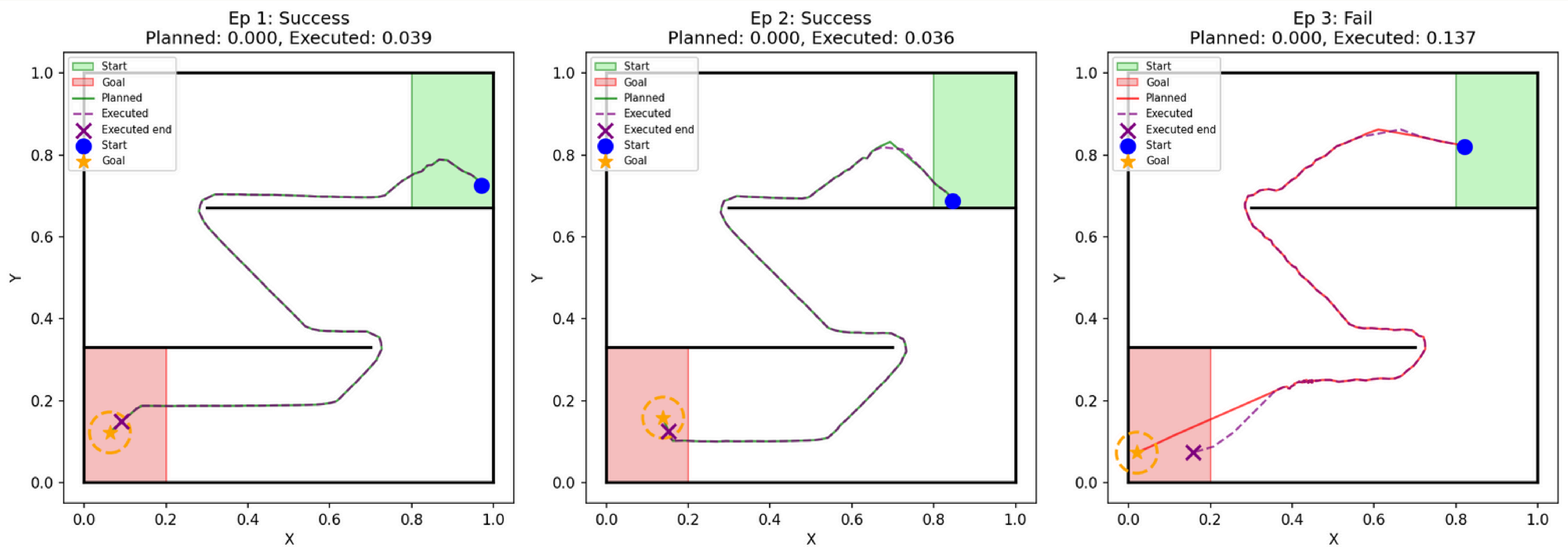
RESULTS - EXECUTED TRAJECTORIES



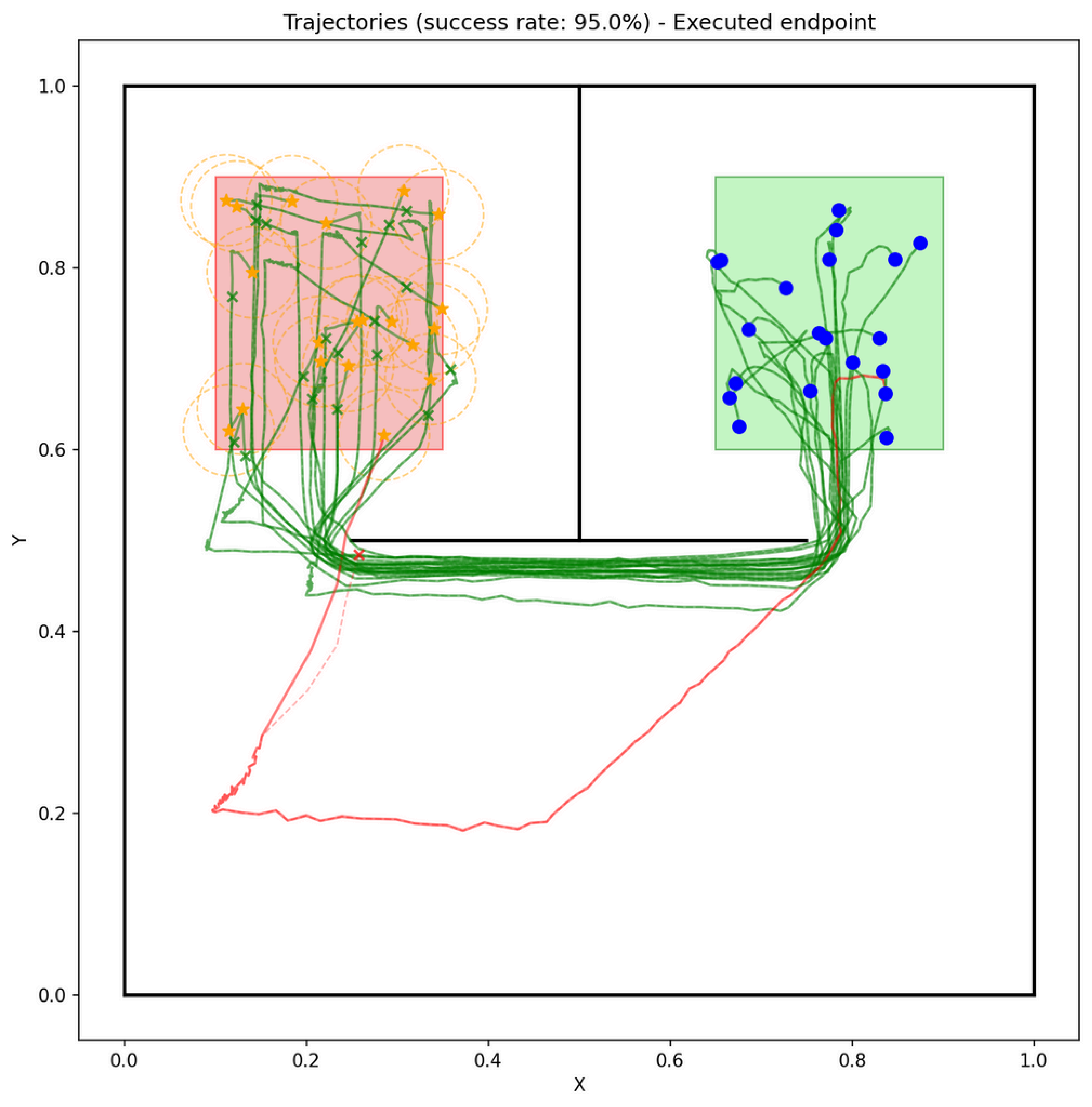


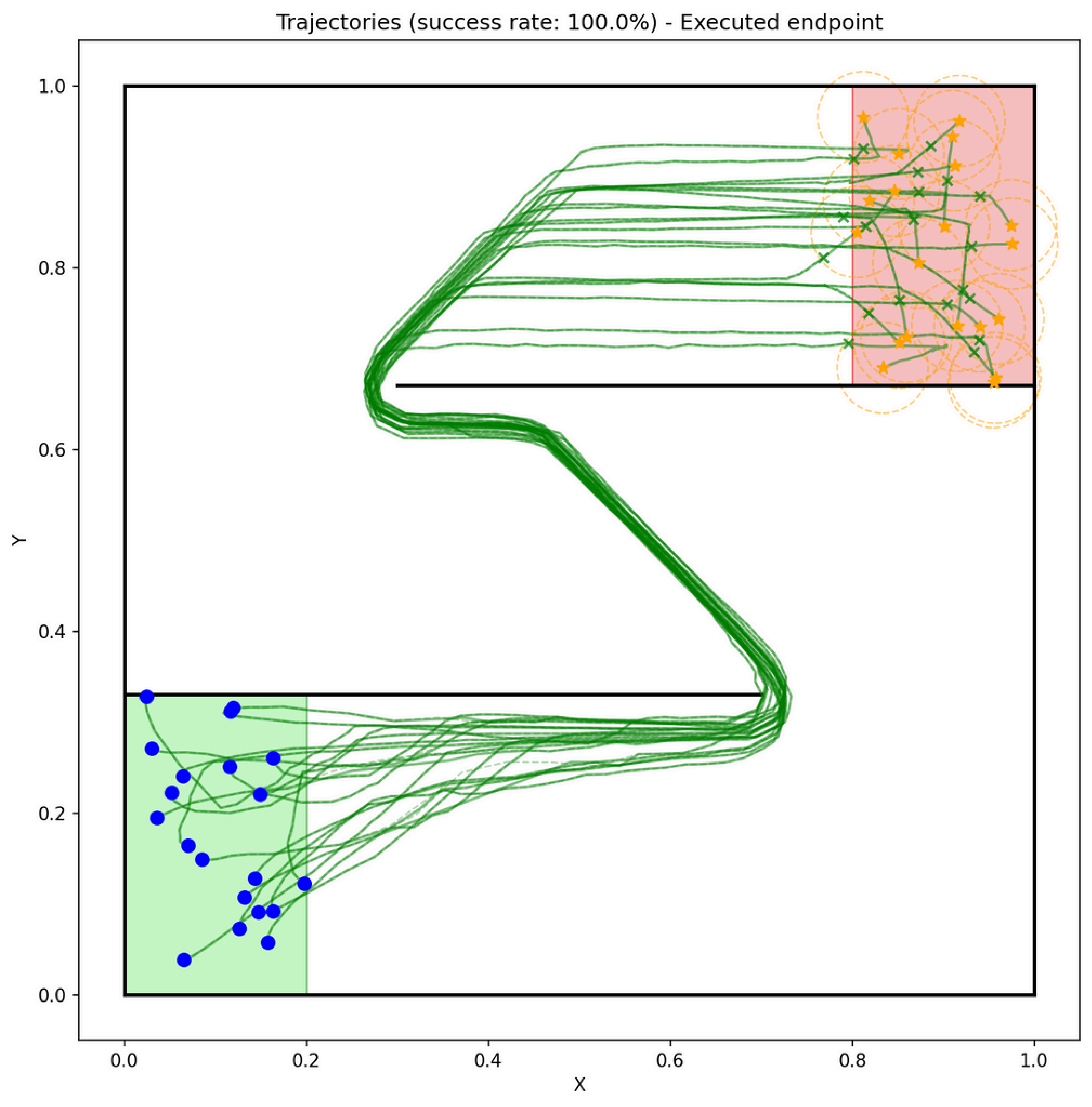
RESULTS - EXECUTED TRAJECTORIES



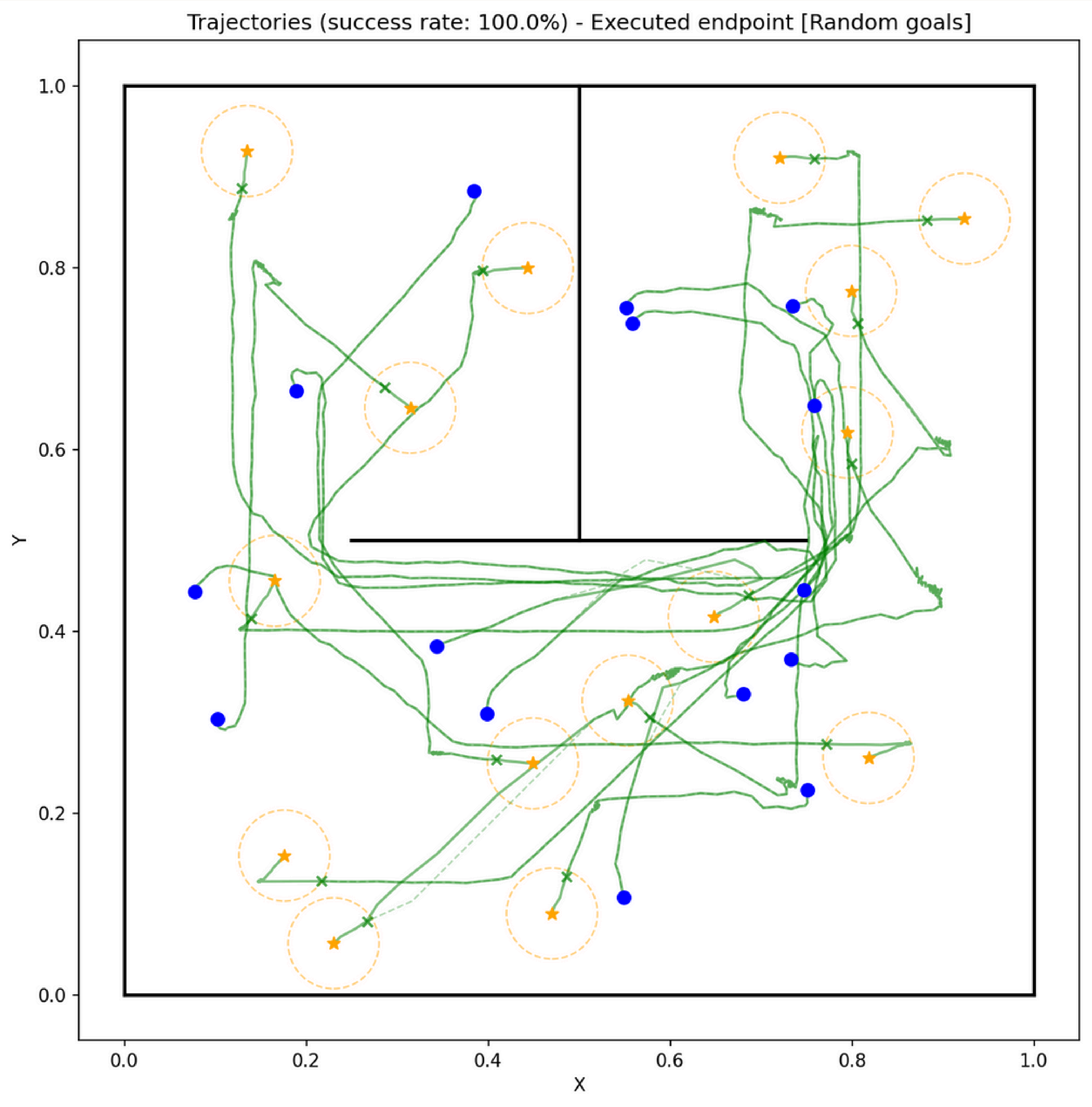


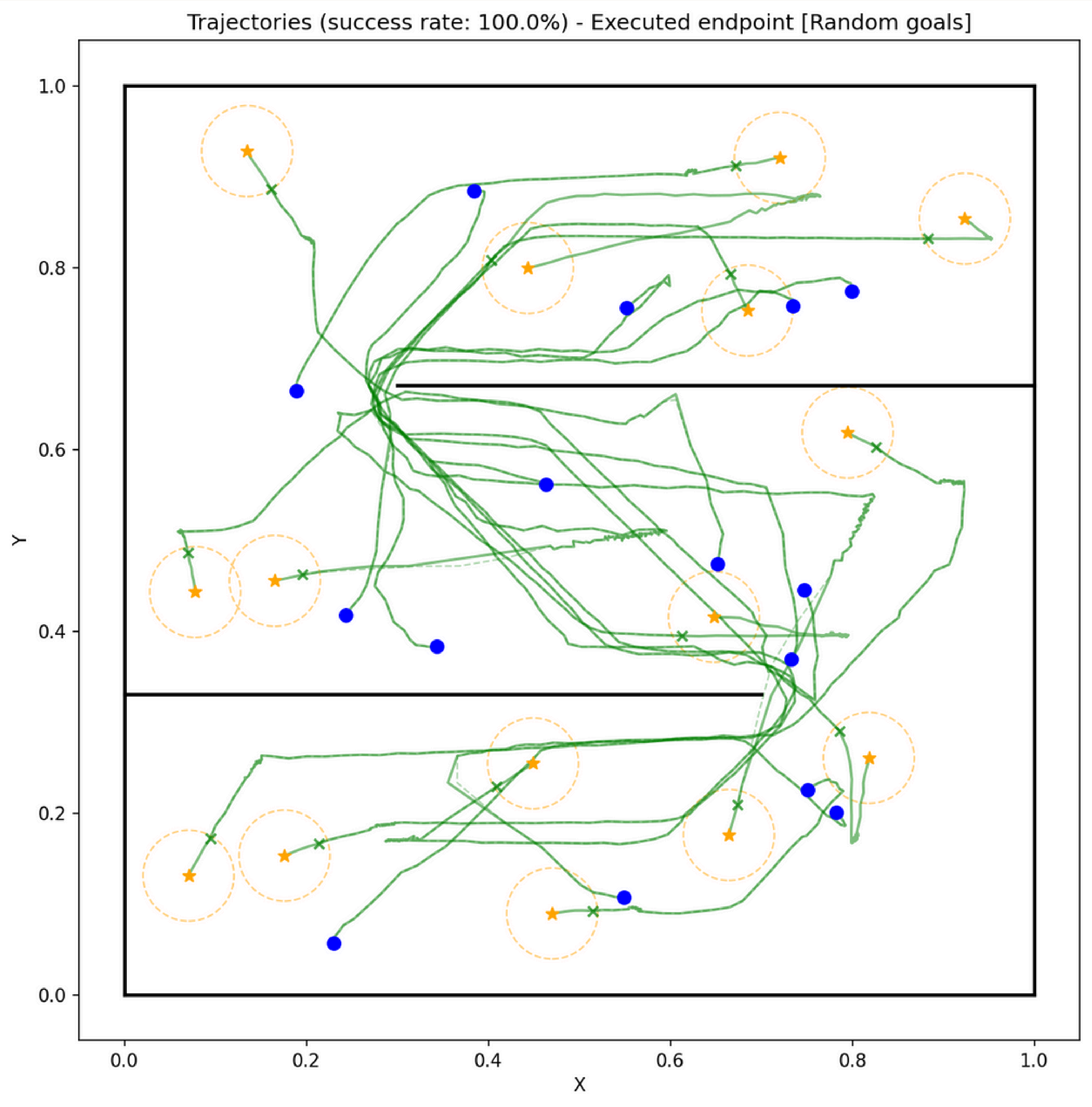
RESULTS - GENERALIZATION: INVERTING THE GOALS





RESULTS - GENERALIZATION: RANDOM GOALS





CONCLUSIONS AND FUTURE WORKS

Diffusion proved to be a viable approach to solve planning problems, however, we noted with our implementation that:

- Near goals, the model sometimes contorted itself; smooth inpainting alleviated the problem, but it did not solve it.
    - Possible cause: training data sits still if it reaches goal early → model does not learn to decelerate smoothly

- Wall compenetration.
    - Possible solutions: wall locations as an additional input channel to the U-Net, collision-penalizing loss

Thank You For Your Attention