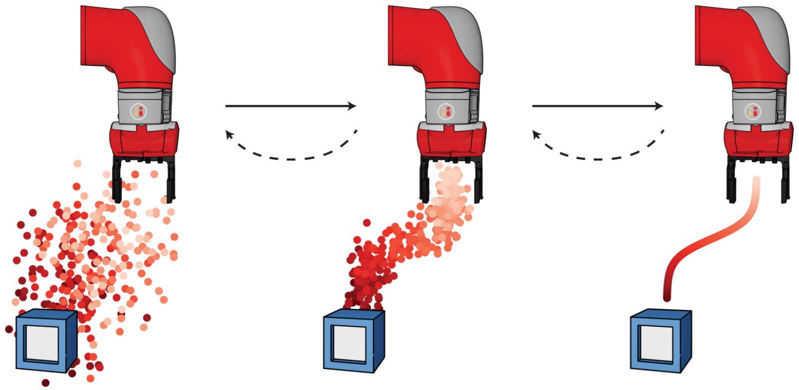
Planning with Diffusion for Flexible Behavior Synthesis

Michael Janner* Yilun Du* Joshua B. Tenenbaum Sergey Levine

*equal contribution

`diffusion-planning.github.io`



ICML 2022

A simple recipe for RL and data-driven control:

A simple recipe for RL and data-driven control:

Inputs: Dataset of transitions $\mathcal{D} = \{(\mathbf{s}_t, \mathbf{a}_t, \mathbf{s}_{t+1}), \ldots\}$

A simple recipe for RL and data-driven control:

Inputs: Dataset of transitions $\mathcal{D} = \{(\mathbf{s}_t, \mathbf{a}_t, \mathbf{s}_{t+1}), \ldots\}$, reward function $r(\cdot, \cdot)$

A simple recipe for RL and data-driven control:

Inputs: Dataset of transitions $\mathcal{D} = \{(\mathbf{s}_t, \mathbf{a}_t, \mathbf{s}_{t+1}), \ldots\}$, reward function $r(\cdot, \cdot)$, current state $\mathbf{s}_0$

A simple recipe for RL and data-driven control:

Inputs: Dataset of transitions $\mathcal{D} = \{(\mathbf{s}_t, \mathbf{a}_t, \mathbf{s}_{t+1}), \ldots\}$, reward function $r(\cdot, \cdot)$, current state $\mathbf{s}_0$

1: Train a predictive model

$$\underset{\boldsymbol{f}}{\text{minimize}}\ \mathbb{E}_{\mathbf{s}_t, \mathbf{a}_t, \mathbf{s}_{t+1} \sim \mathcal{D}} \left[\|\mathbf{s}_{t+1} - \boldsymbol{f}(\mathbf{s}_t, \mathbf{a}_t)\| \right]$$

A simple recipe for RL and data-driven control:

Inputs: Dataset of transitions $\mathcal{D} = \{(\mathbf{s}_t, \mathbf{a}_t, \mathbf{s}_{t+1}), \ldots\}$, reward function $r(\cdot, \cdot)$, current state $\mathbf{s}_0$

1: Train a predictive model

$$\underset{\boldsymbol{f}}{\text{minimize}} \ \mathbb{E}_{\mathbf{s}_t, \mathbf{a}_t, \mathbf{s}_{t+1} \sim \mathcal{D}} \left[\| \mathbf{s}_{t+1} - \boldsymbol{f}(\mathbf{s}_t, \mathbf{a}_t) \| \right]$$

2: Use model to evaluate potential plans $\mathbf{a}_{0:T}$, selecting the best one:

$$\underset{\mathbf{a}_{0:T}}{\text{maximize}} \ r(\mathbf{s}_0, \mathbf{a}_0) + r(\mathbf{s}_1, \mathbf{a}_1) + r(\mathbf{s}_2, \mathbf{a}_2) + \ldots$$

A simple recipe for RL and data-driven control:

Inputs: Dataset of transitions $\mathcal{D} = \{(\mathbf{s}_t, \mathbf{a}_t, \mathbf{s}_{t+1}), \ldots\}$, reward function $r(\cdot, \cdot)$, current state $\mathbf{s}_0$

1: Train a predictive model

$$\underset{\boldsymbol{f}}{\text{minimize}}\ \mathbb{E}_{\mathbf{s}_t, \mathbf{a}_t, \mathbf{s}_{t+1} \sim \mathcal{D}} \left[||\mathbf{s}_{t+1} - \boldsymbol{f}(\mathbf{s}_t, \mathbf{a}_t)|| \right]$$

2: Use model to evaluate potential plans $\mathbf{a}_{0:T}$, selecting the best one:

$$\underset{\mathbf{a}_{0:T}}{\text{maximize}}\ r(\mathbf{s}_0, \mathbf{a}_0) + r(\boldsymbol{f}(\mathbf{s}_0, \mathbf{a}_0), \mathbf{a}_1) + r(\boldsymbol{f}(\boldsymbol{f}(\mathbf{s}_0, \mathbf{a}_0), \mathbf{a}_1), \mathbf{a}_2) + \ldots$$

A simple recipe for RL and data-driven control:

Algorithm 1 Model-based RL (idealized)

Inputs: Dataset of transitions $\mathcal{D} = \{(\mathbf{s}_t, \mathbf{a}_t, \mathbf{s}_{t+1}), \ldots\}$, reward function $r(\cdot, \cdot)$, current state $\mathbf{s}_0$

1: Train a predictive model

$$\underset{\boldsymbol{f}}{\text{minimize}}\ \mathbb{E}_{\mathbf{s}_t, \mathbf{a}_t, \mathbf{s}_{t+1} \sim \mathcal{D}} \left[\| \mathbf{s}_{t+1} - \boldsymbol{f}(\mathbf{s}_t, \mathbf{a}_t) \| \right]$$

2: Use model to evaluate potential plans $\mathbf{a}_{0:T}$, selecting the best one:

$$\underset{\mathbf{a}_{0:T}}{\text{maximize}}\ r(\mathbf{s}_0, \mathbf{a}_0) + r(\boldsymbol{f}(\mathbf{s}_0, \mathbf{a}_0), \mathbf{a}_1) + r(\boldsymbol{f}(\boldsymbol{f}(\mathbf{s}_0, \mathbf{a}_0), \mathbf{a}_1), \mathbf{a}_2) + \ldots$$

A simple recipe for RL and data-driven control:

Algorithm 1 Model-based RL (idealized)

Inputs: Dataset of transitions $\mathcal{D} = \{(\mathbf{s}_t, \mathbf{a}_t, \mathbf{s}_{t+1}), \ldots\}$, reward function $r(\cdot, \cdot)$, current state $\mathbf{s}_0$

1: Train a predictive model

$$\underset{\boldsymbol{f}}{\text{minimize}}\ \mathbb{E}_{\mathbf{s}_t, \mathbf{a}_t, \mathbf{s}_{t+1} \sim \mathcal{D}} \left[\| \mathbf{s}_{t+1} - \boldsymbol{f}(\mathbf{s}_t, \mathbf{a}_t) \| \right]$$

2: Use model to evaluate potential plans $\mathbf{a}_{0:T}$, selecting the best one:

$$\underset{\mathbf{a}_{0:T}}{\text{maximize}}\ r(\mathbf{s}_0, \mathbf{a}_0) + r(\boldsymbol{f}(\mathbf{s}_0, \mathbf{a}_0), \mathbf{a}_1) + r(\boldsymbol{f}(\boldsymbol{f}(\mathbf{s}_0, \mathbf{a}_0), \mathbf{a}_1), \mathbf{a}_2) + \ldots$$

planning "in the now": Kaelbling & Lozano-Pérez. AAAI 2010.
van Hasselt, Hessel, & Aslanides. NeurIPS 2019.

A simple recipe for RL and data-driven control:

Algorithm 1 Model-based RL (idealized)

Inputs: Dataset of transitions $\mathcal{D} = \{(\mathbf{s}_t, \mathbf{a}_t, \mathbf{s}_{t+1}), \ldots\}$, reward function $r(\cdot, \cdot)$, current state $\mathbf{s}_0$

1: Train a predictive model

supervised learning $\boxed{\min_{\boldsymbol{f}} \text{imize } \mathbb{E}_{\mathbf{s}_t, \mathbf{a}_t, \mathbf{s}_{t+1} \sim \mathcal{D}} \left[\|\mathbf{s}_{t+1} - \boldsymbol{f}(\mathbf{s}_t, \mathbf{a}_t)\| \right]}$

2: Use model to evaluate potential plans $\mathbf{a}_{0:T}$, selecting the best one:

$$\underset{\mathbf{a}_{0:T}}{\text{maximize}} \; r(\mathbf{s}_0, \mathbf{a}_0) + r(\boldsymbol{f}(\mathbf{s}_0, \mathbf{a}_0), \mathbf{a}_1) + r(\boldsymbol{f}(\boldsymbol{f}(\mathbf{s}_0, \mathbf{a}_0), \mathbf{a}_1), \mathbf{a}_2) + \ldots$$

A simple recipe for RL and data-driven control:

Algorithm 1 Model-based RL (idealized)

Inputs: Dataset of transitions $\mathcal{D} = \{(\mathbf{s}_t, \mathbf{a}_t, \mathbf{s}_{t+1}), \ldots\}$, reward function $r(\cdot,\cdot)$, current state $\mathbf{s}_0$

1: Train a predictive model

supervised learning

$$\boxed{\underset{\boldsymbol{f}}{\text{minimize}}\ \mathbb{E}_{\mathbf{s}_t,\mathbf{a}_t,\mathbf{s}_{t+1}\sim\mathcal{D}}\left[\|\mathbf{s}_{t+1} - \boldsymbol{f}(\mathbf{s}_t, \mathbf{a}_t)\|\right]}$$

2: Use model to evaluate potential plans $\mathbf{a}_{0:T}$, selecting the best one:

trajectory optimization

$$\boxed{\underset{\mathbf{a}_{0:T}}{\text{maximize}}\ r(\mathbf{s}_0, \mathbf{a}_0) + r(\boldsymbol{f}(\mathbf{s}_0, \mathbf{a}_0), \mathbf{a}_1) + r(\boldsymbol{f}(\boldsymbol{f}(\mathbf{s}_0, \mathbf{a}_0), \mathbf{a}_1), \mathbf{a}_2) + \ldots}$$

Neural nets + trajectory optimization

Great in principle, a headache in practice.

Neural nets + trajectory optimization

Great in principle, a headache in practice.

- It does sometimes work [Chua et al. 2018; Argenson & Dulac-Arnold 2021]

- But most contemporary model-based RL algorithms pull more from the model-free RL toolbox than from trajectory optimization

Neural nets + trajectory optimization

Great in principle, a headache in practice.

Why?

Neural nets + trajectory optimization

Great in principle, a headache in practice.

Why?

- Long-horizon predictions are unreliable

Neural nets + trajectory optimization

Great in principle, a headache in practice.

Why?

- Long-horizon predictions are unreliable

- Optimizing for reward with neural net models produces adversarial examples in trajectory space

Neural nets + trajectory optimization

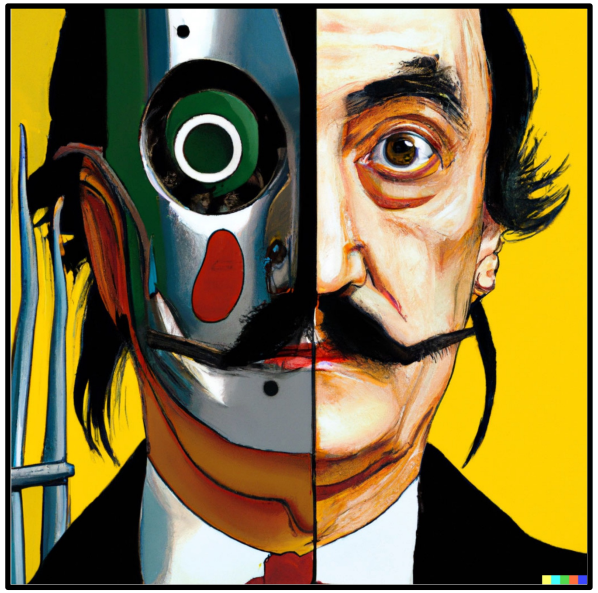
Great in principle, a headache in practice.

Why?

- Long-horizon predictions are unreliable
- Optimizing for reward with neural net models produces adversarial examples in trajectory space
- **Neural net models aren't good enough?**

Is the model the bottleneck?

- This reasoning should sound suspicious
- Generative modeling (on images) is really working!

- What can RL learn from these successes?

DALL-E 2: Ramesh et al, 2022. Imagen: Saharia et al, 2022.

Planning as generative modeling

- Offload as much of MBRL into contemporary generative modeling as possible

Algorithm 1 Model-based RL (idealized)

1: **Inputs:** Dataset of transitions $\mathcal{D} = \{(\mathbf{s}_t, \mathbf{a}_t, \mathbf{s}_{t+1}), \dots\}$, reward function $r(\cdot, \cdot)$, current state $\mathbf{s}_0$

2: Train a predictive model

$$\underset{\boldsymbol{f}}{\text{minimize}} \; \mathbb{E}_{\mathbf{s}_t, \mathbf{a}_t, \mathbf{s}_{t+1} \sim \mathcal{D}} \left[\| \mathbf{s}_{t+1} - \boldsymbol{f}(\mathbf{s}_t, \mathbf{a}_t) \| \right]$$

3: Use model to evaluate potential plans $\mathbf{a}_{0:T}$, selecting the best one:

$$\underset{\mathbf{a}_{0:T}}{\text{maximize}} \; r(\mathbf{s}_0, \mathbf{a}_0) + r(\mathbf{s}_1, \mathbf{a}_1) + r(\mathbf{s}_2, \mathbf{a}_2) + \dots$$

Planning as generative modeling

- Offload as much of MBRL into contemporary generative modeling as possible

replace **prediction** with big generative model

Algorithm 1 Model-based RL (idealized)

Inputs: Dataset of transitions $\mathcal{D} = \{(\mathbf{s}_t, \mathbf{a}_t, \mathbf{s}_{t+1}), \ldots\}$, reward function $r(\cdot, \cdot)$, current state $\mathbf{s}_0$

1: Train a predictive model

$$\boxed{\underset{\boldsymbol{f}}{\text{minimize}}\ \mathbb{E}_{\mathbf{s}_t, \mathbf{a}_t, \mathbf{s}_{t+1} \sim \mathcal{D}} \left[\| \mathbf{s}_{t+1} - \boldsymbol{f}(\mathbf{s}_t, \mathbf{a}_t) \| \right]}$$

2: Use model to evaluate potential plans $\mathbf{a}_{0:T}$, selecting the best one:

$$\underset{\mathbf{a}_{0:T}}{\text{maximize}}\ r(\mathbf{s}_0, \mathbf{a}_0) + r(\mathbf{s}_1, \mathbf{a}_1) + r(\mathbf{s}_2, \mathbf{a}_2) + \ldots$$

Planning as generative modeling

- Offload as much of MBRL into contemporary generative modeling as possible

replace **prediction and planning** with big generative model

Algorithm 1 Model-based RL (idealized)

Inputs: Dataset of transitions $\mathcal{D} = \{(\mathbf{s}_t, \mathbf{a}_t, \mathbf{s}_{t+1}), \ldots\}$, reward function $r(\cdot, \cdot)$, current state $\mathbf{s}_0$

1: Train a predictive model

$$\underset{\boldsymbol{f}}{\text{minimize}} \ \mathbb{E}_{\mathbf{s}_t, \mathbf{a}_t, \mathbf{s}_{t+1} \sim \mathcal{D}} \left[\|\mathbf{s}_{t+1} - \boldsymbol{f}(\mathbf{s}_t, \mathbf{a}_t)\| \right]$$

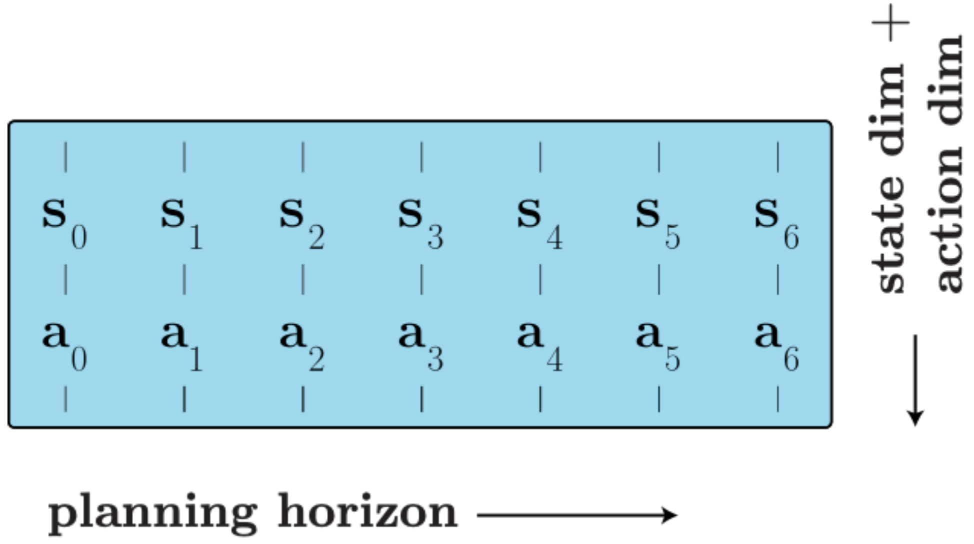
2: Use model to evaluate potential plans $\mathbf{a}_{0:T}$, selecting the best one:

$$\underset{\mathbf{a}_{0:T}}{\text{maximize}} \ r(\mathbf{s}_0, \mathbf{a}_0) + r(\mathbf{s}_1, \mathbf{a}_1) + r(\mathbf{s}_2, \mathbf{a}_2) + \ldots$$

A generative model of trajectories

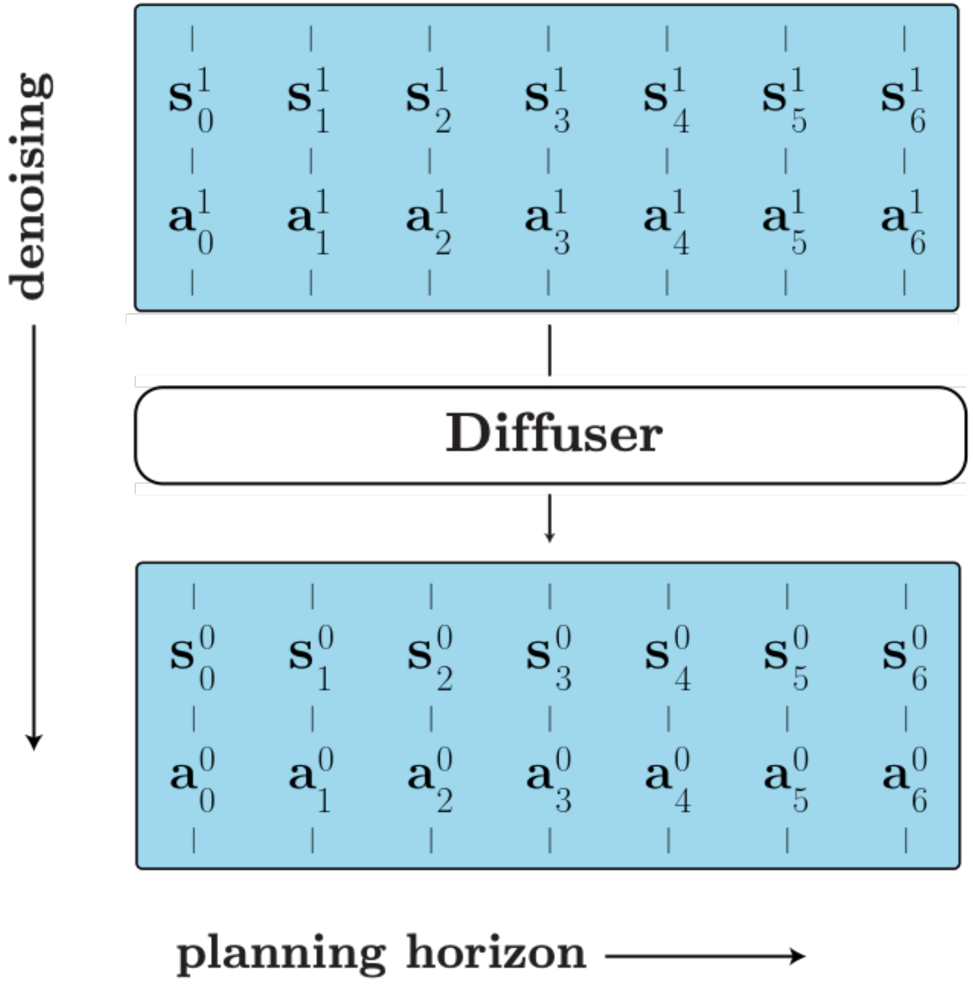
- Represent trajectories as single-channel images

$\tau =$



A generative model of trajectories

- Represent trajectories as single-channel images
- Train a diffusion model to iteratively denoise entire trajectory



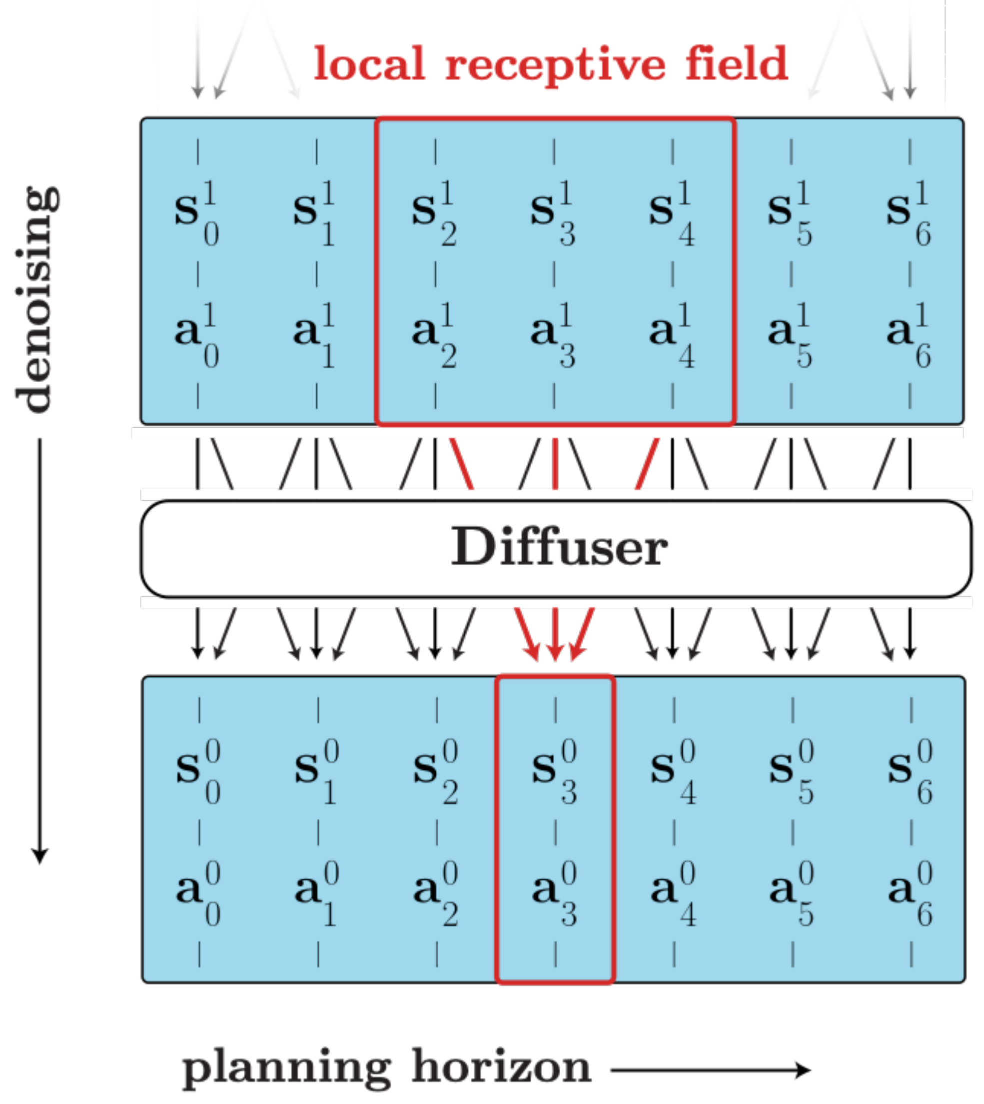
diffusion models: Sohl-Dickstein et al. ICML 2015 & Ho et al. NeurIPS 2020.

A generative model of trajectories

- Represent trajectories as single-channel images

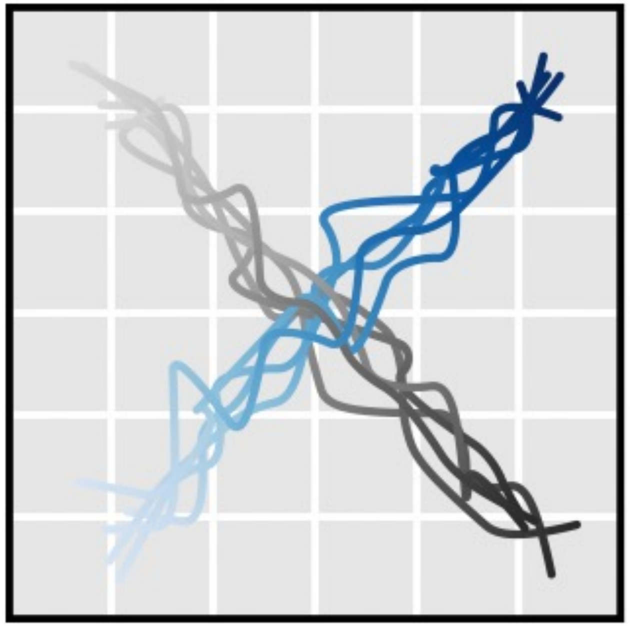
- Train a diffusion model to iteratively denoise entire trajectory

- Use (one-dimensional) convolutions for temporal equivariance and horizon-independence



Compositionality via local consistency

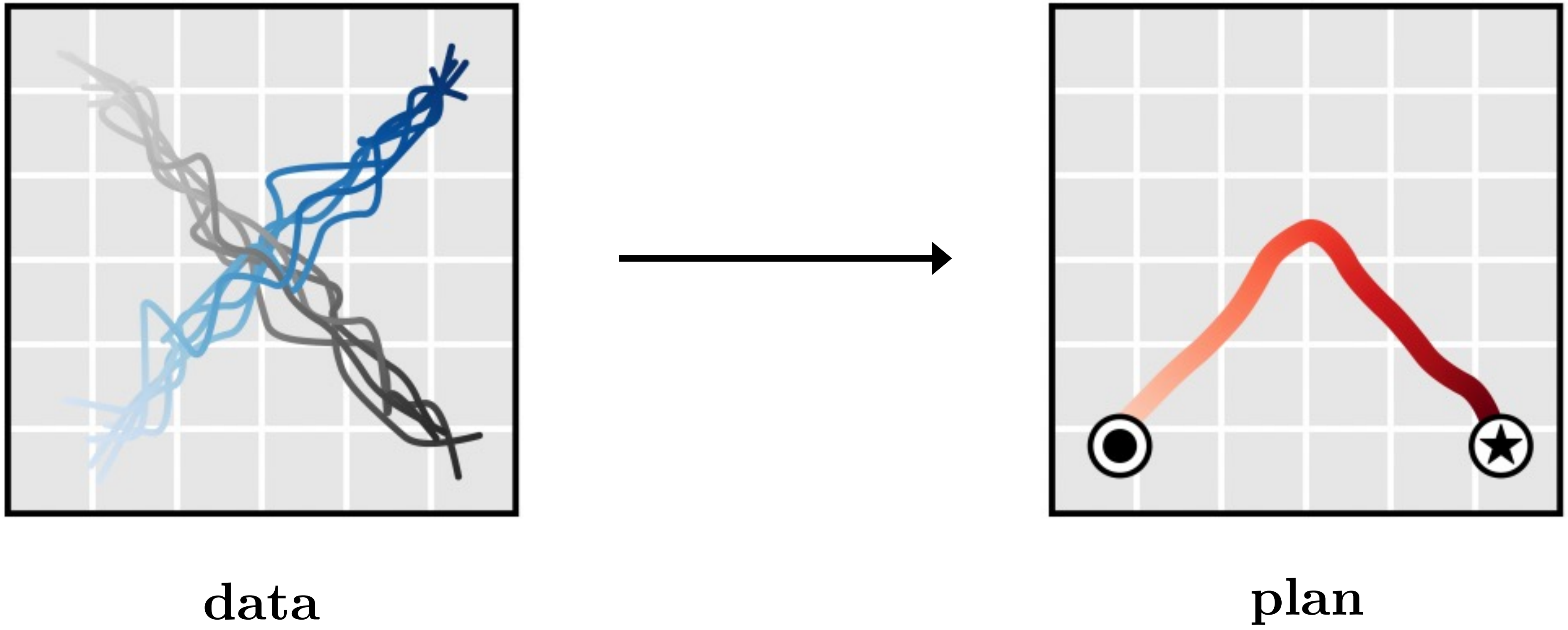
- Diffuser is non-Markovian, but still compositional due to temporal convolutions



data

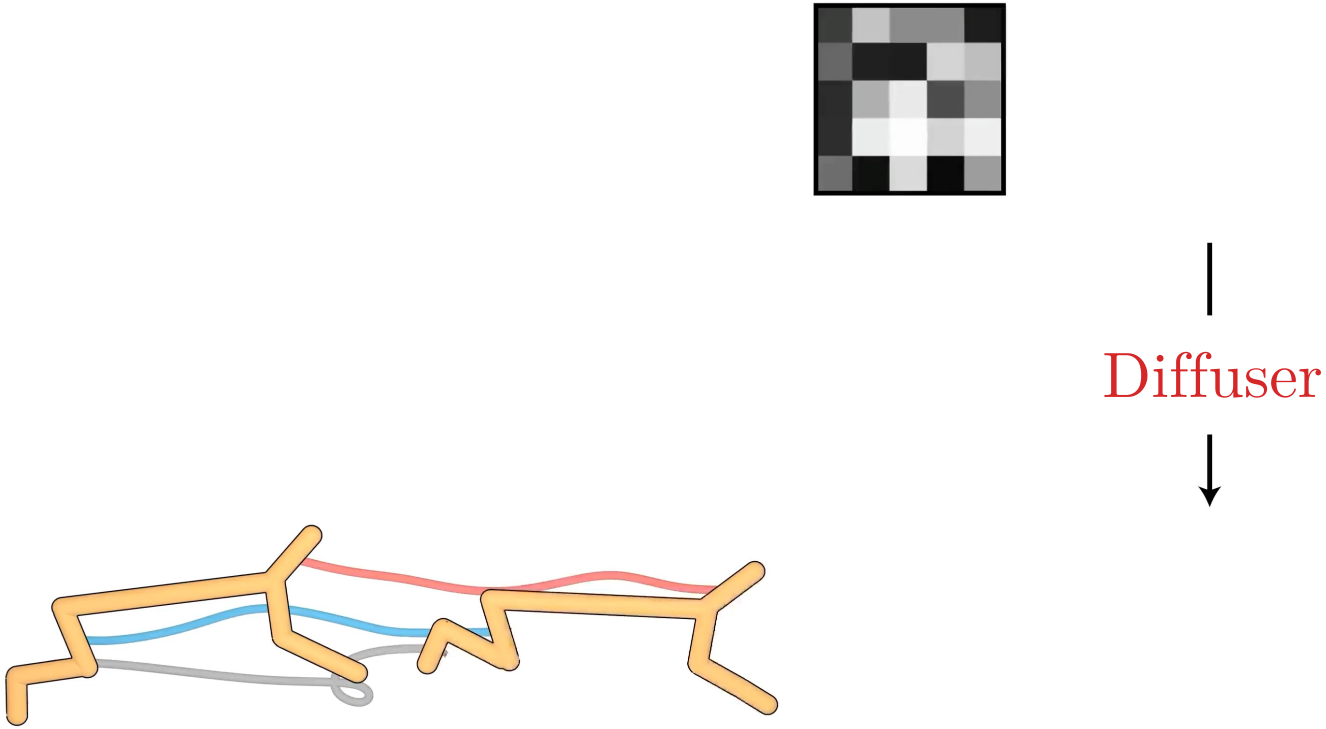
Compositionality via local consistency

- Diffuser is non-Markovian, but still compositional due to temporal convolutions



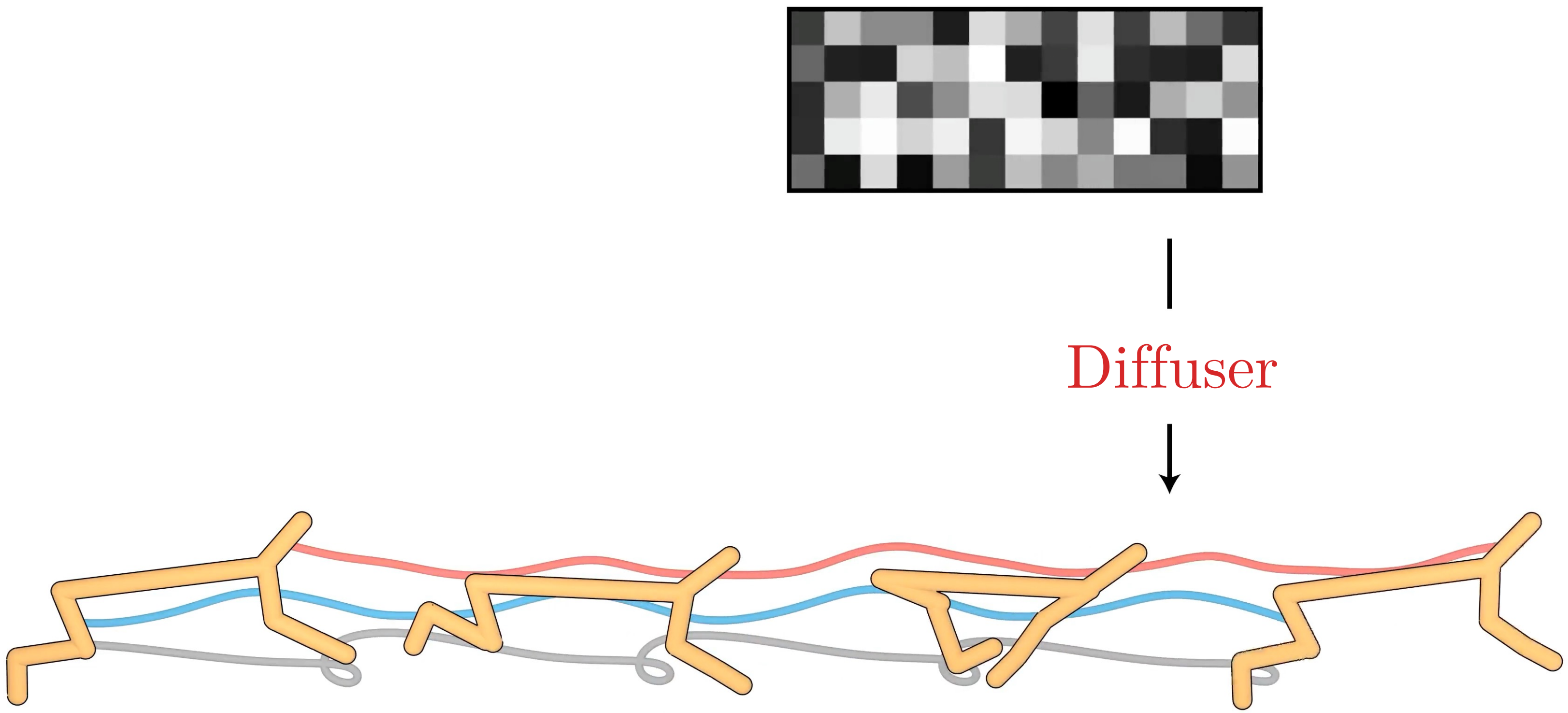
Variable-length predictions

- Trajectory horizon is determined by the size of the noise initialization



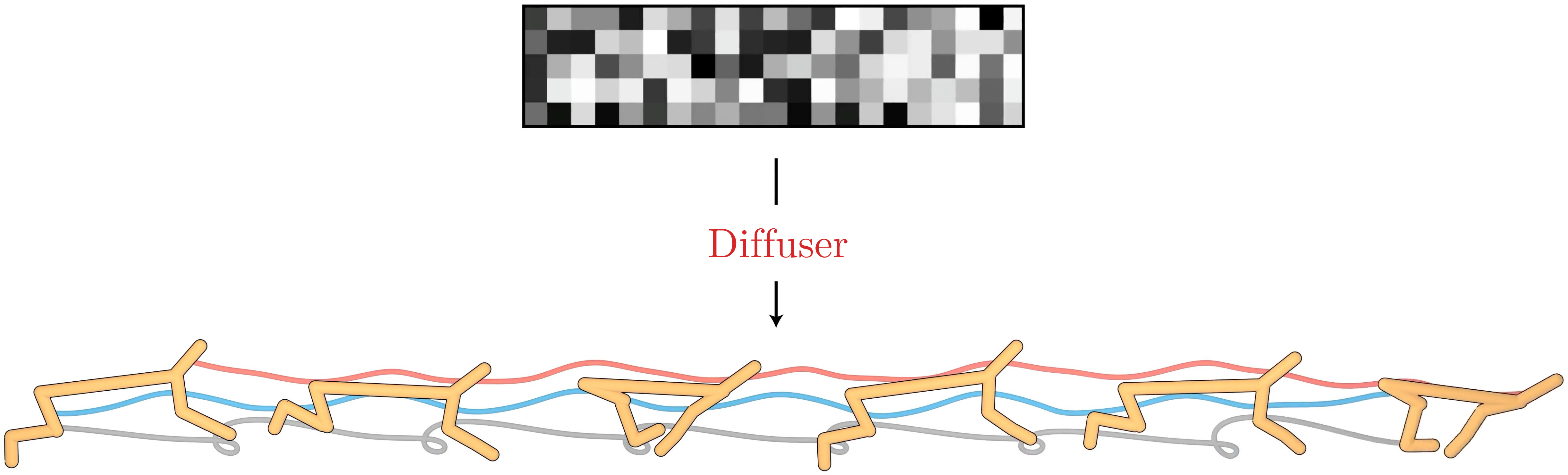
Variable-length predictions

- Trajectory horizon is determined by the size of the noise initialization



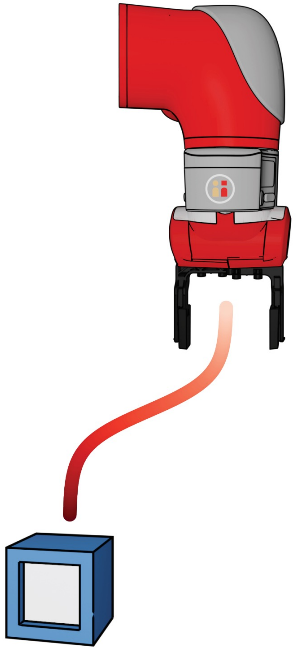
Variable-length predictions

- Trajectory horizon is determined by the size of the noise initialization



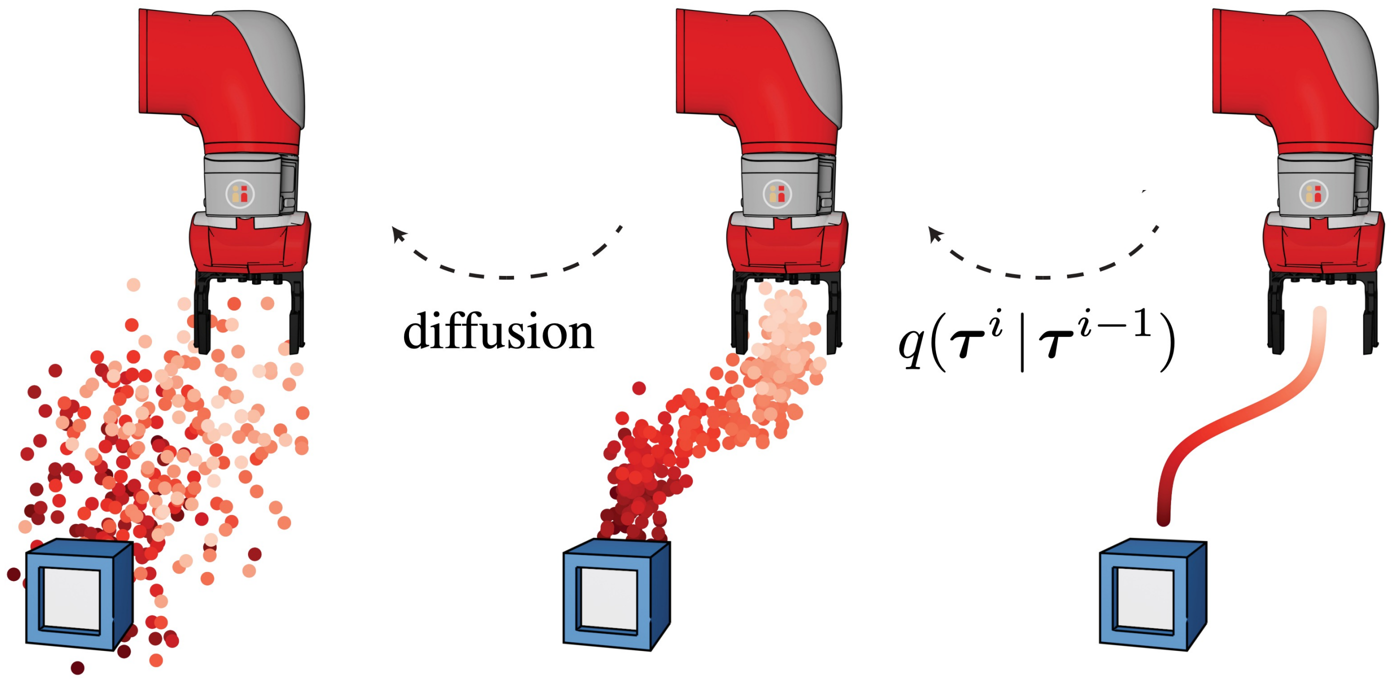
Non-autoregressive prediction

- Prediction is **non-autoregressive**: entire trajectory is predicted simultaneously



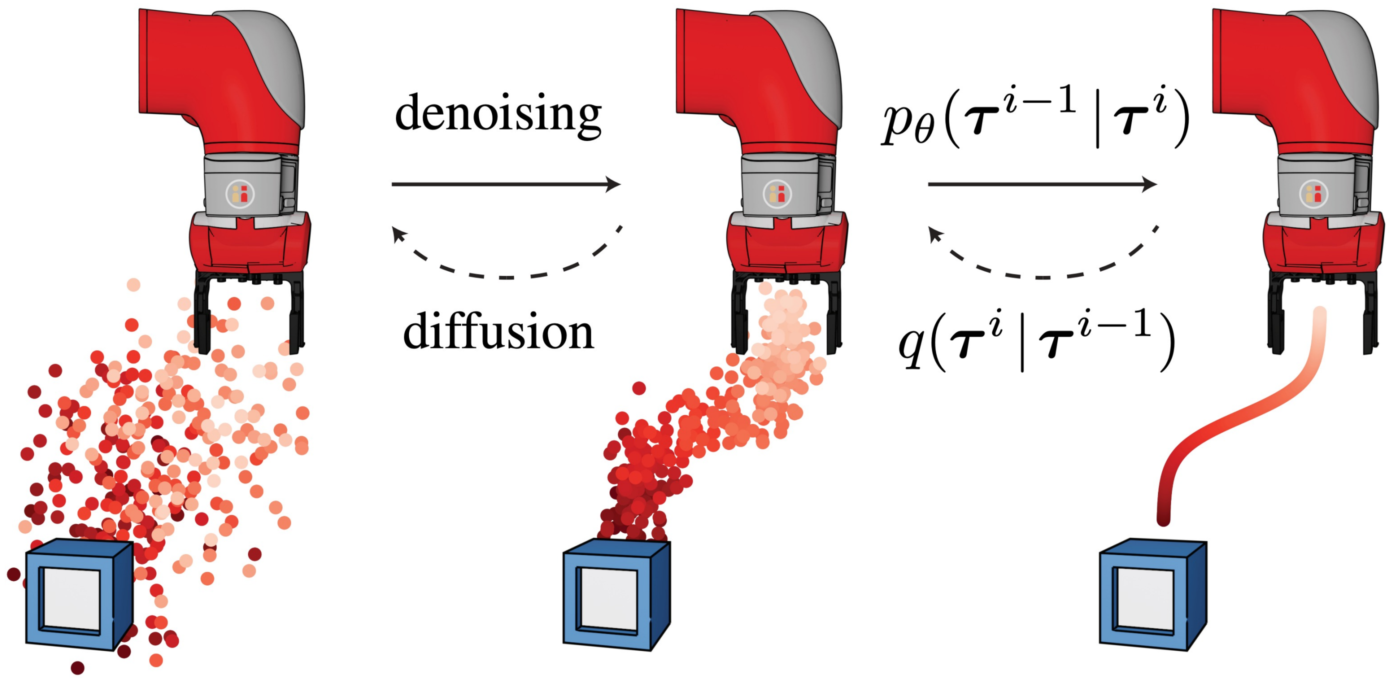
Non-autoregressive prediction

- Prediction is **non-autoregressive**: entire trajectory is predicted simultaneously



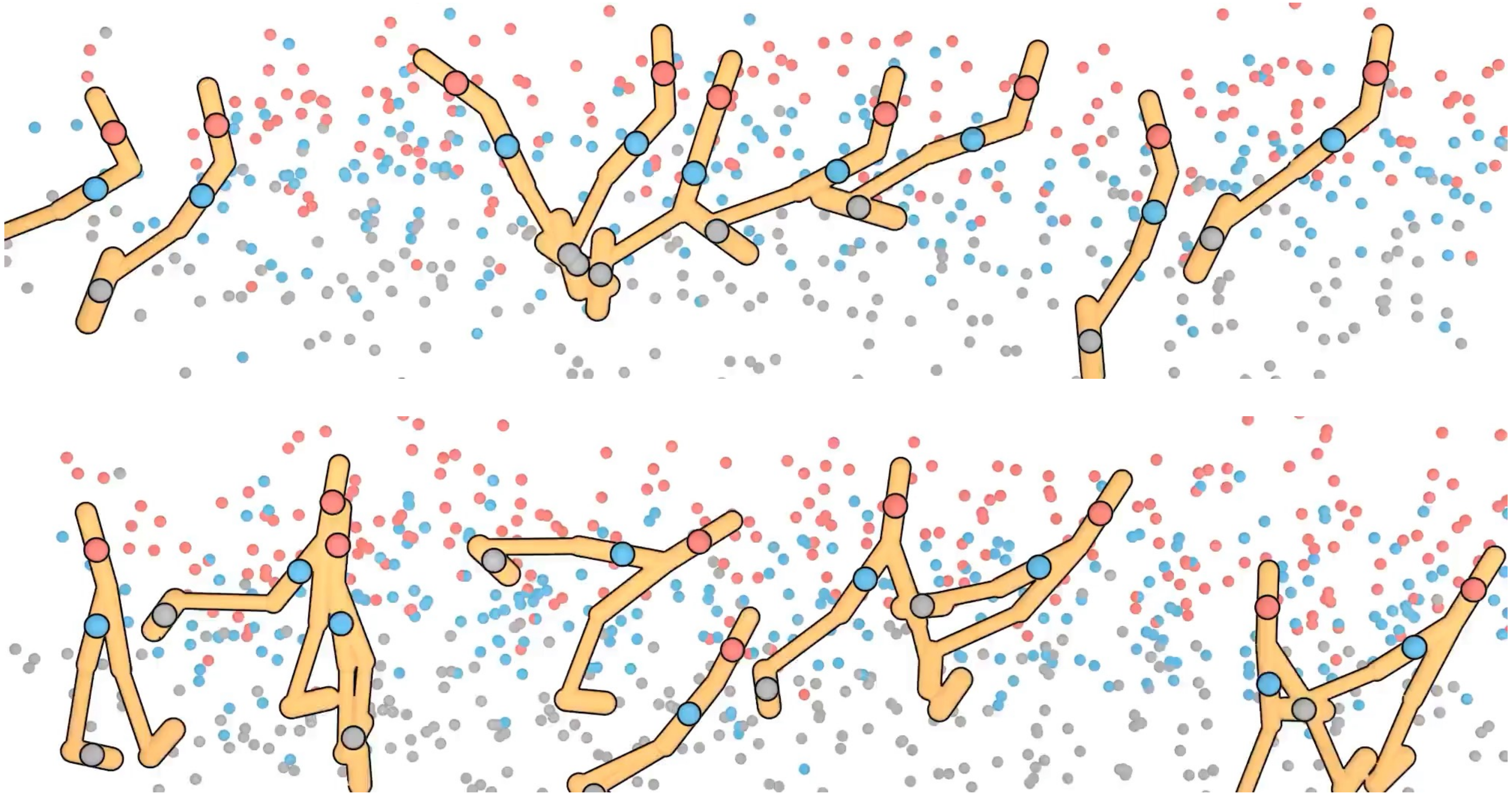
Non-autoregressive prediction

- Prediction is **non-autoregressive**: entire trajectory is predicted simultaneously



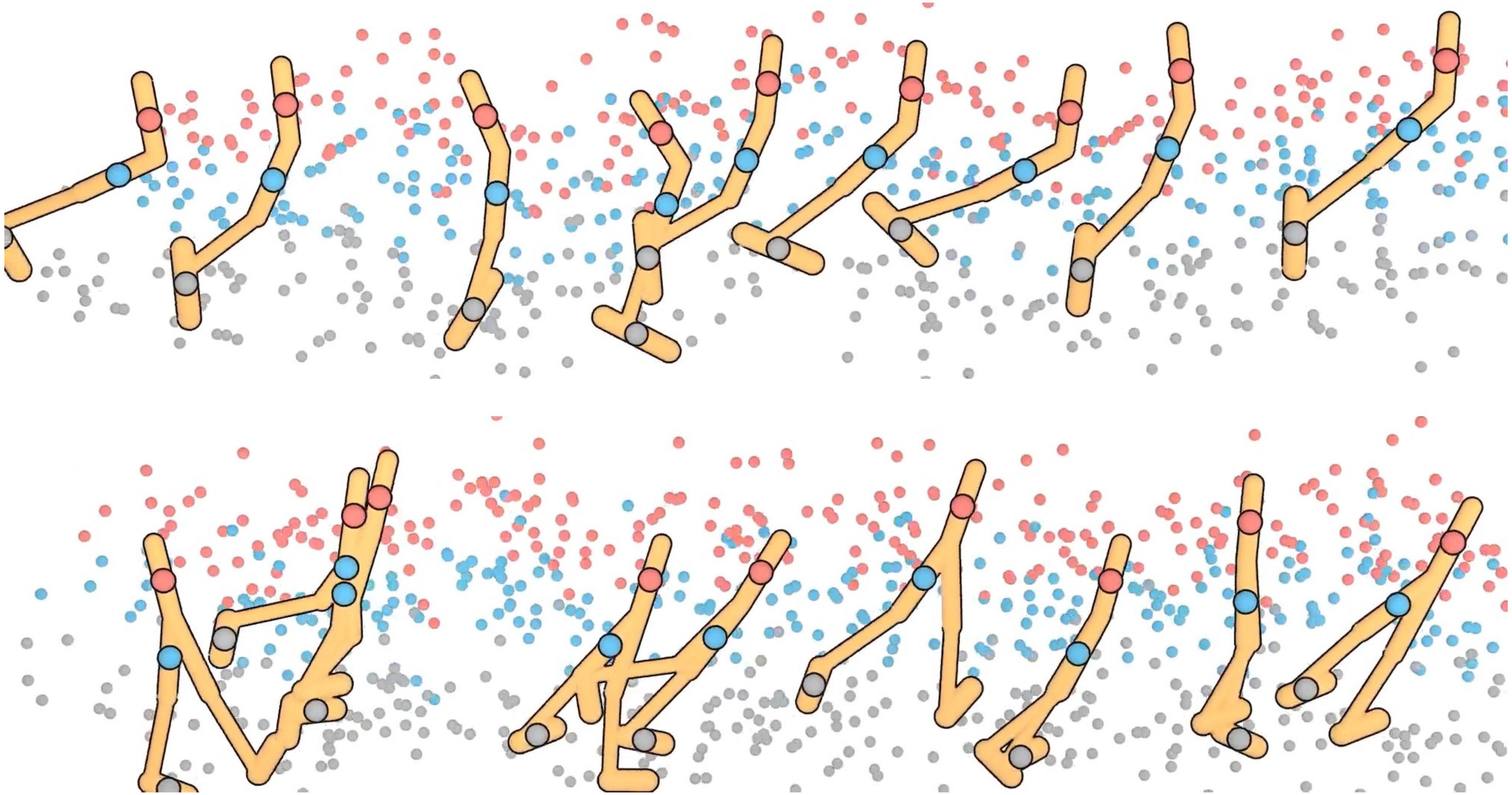
Sampling from Diffuser

- Sampling occurs by iteratively refining randomly-initialized trajectories



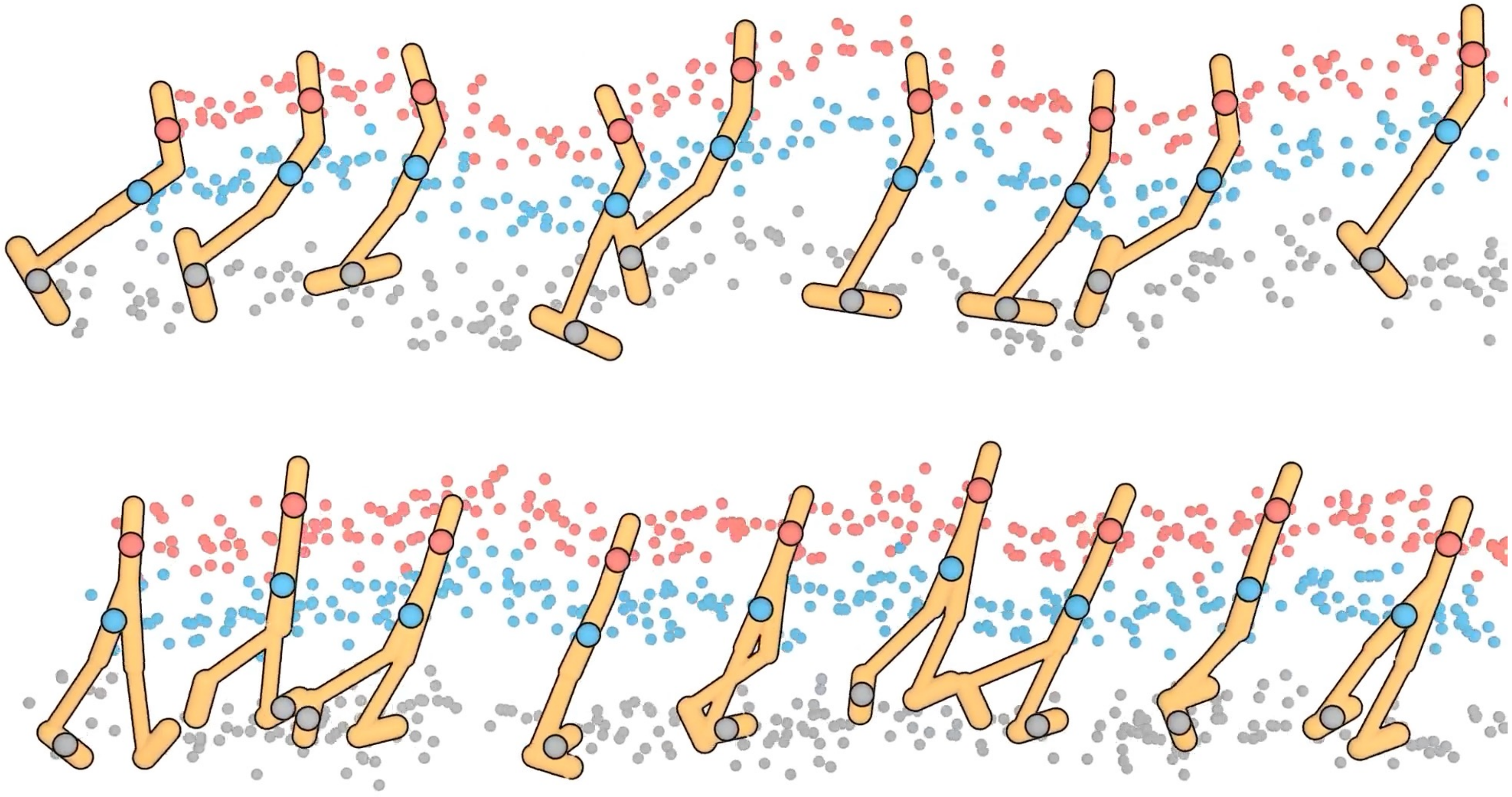
Sampling from Diffuser

- Sampling occurs by iteratively refining randomly-initialized trajectories



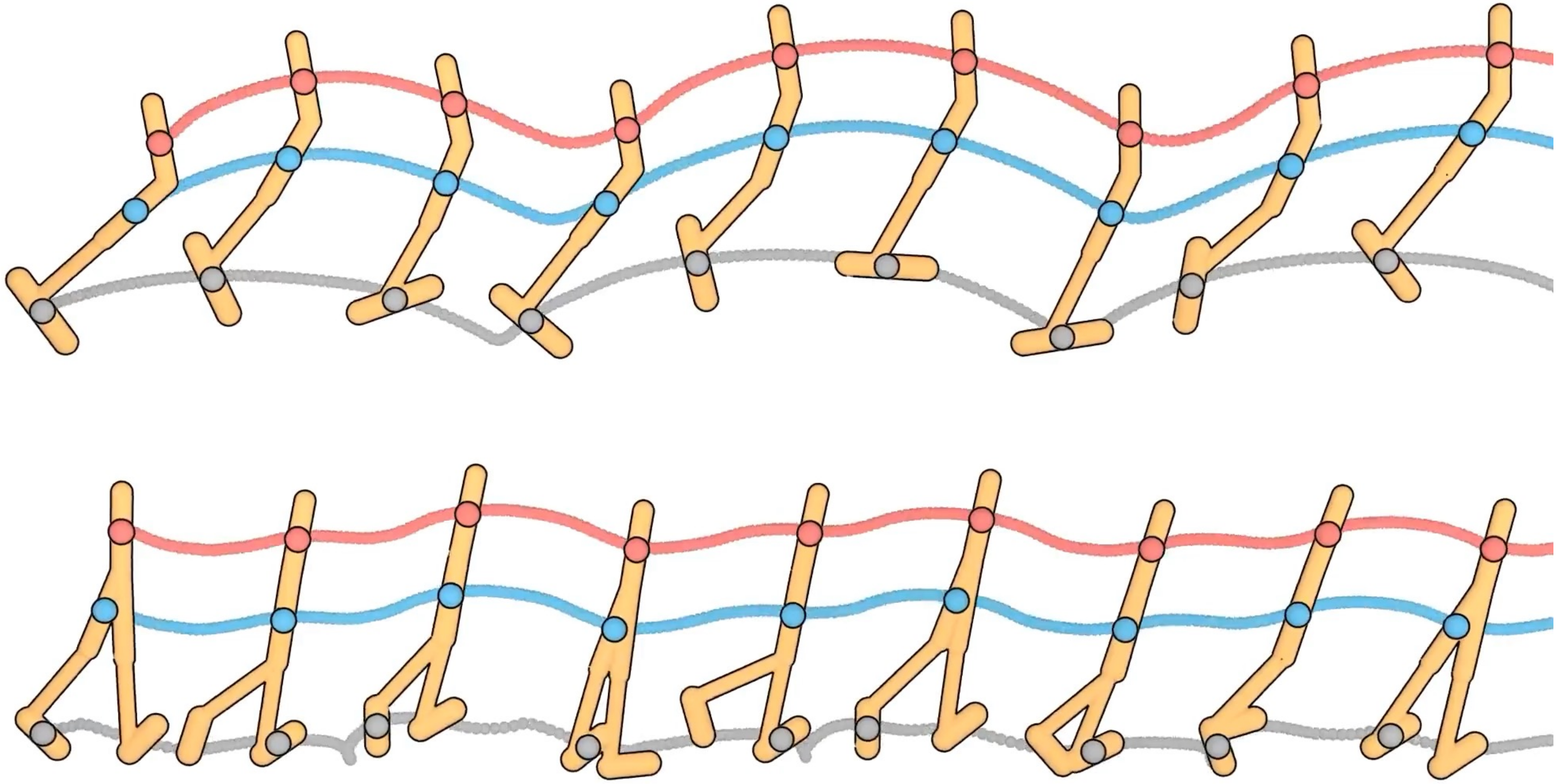
Sampling from Diffuser

- Sampling occurs by iteratively refining randomly-initialized trajectories



Sampling from Diffuser

- Sampling occurs by iteratively refining randomly-initialized trajectories



Flexible Behavior Synthesis through Composing Distributions

Flexible Behavior Synthesis through Composing Distributions

- Synthesize different behaviors through conditional trajectory synthesis with different learned guidance functions:

Flexible Behavior Synthesis through Composing Distributions

- Synthesize different behaviors through conditional trajectory synthesis with different learned guidance functions:

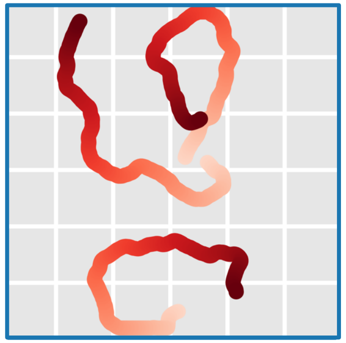
$$\tilde{p}_\theta(\boldsymbol{\tau}) \;\propto\; p_\theta(\boldsymbol{\tau})\, h(\boldsymbol{\tau})$$

See also: Model Based Planning with Energy Based Models: Du at al, CoRL 2019.

Flexible Behavior Synthesis through Composing Distributions

- Synthesize different behaviors through conditional trajectory synthesis with different learned guidance functions:

$$\tilde{p}_\theta(\boldsymbol{\tau}) \propto \underbrace{p_\theta(\boldsymbol{\tau})}_{\text{Diffusion Model}} h(\boldsymbol{\tau})$$

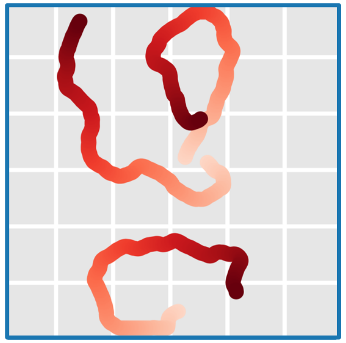


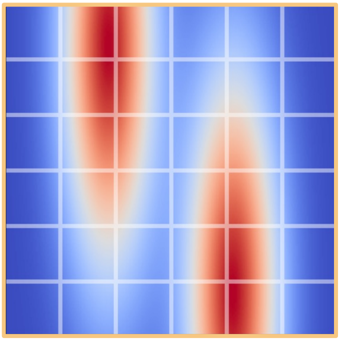
See also: Model Based Planning with Energy Based Models: Du at al, CoRL 2019.

Flexible Behavior Synthesis through Composing Distributions

- Synthesize different behaviors through conditional trajectory synthesis with different learned guidance functions:

$$\tilde{p}_\theta(\boldsymbol{\tau}) \propto \underbrace{p_\theta(\boldsymbol{\tau})}_{\text{Diffusion Model}} \underbrace{h(\boldsymbol{\tau})}_{\text{Guidance Function}}$$



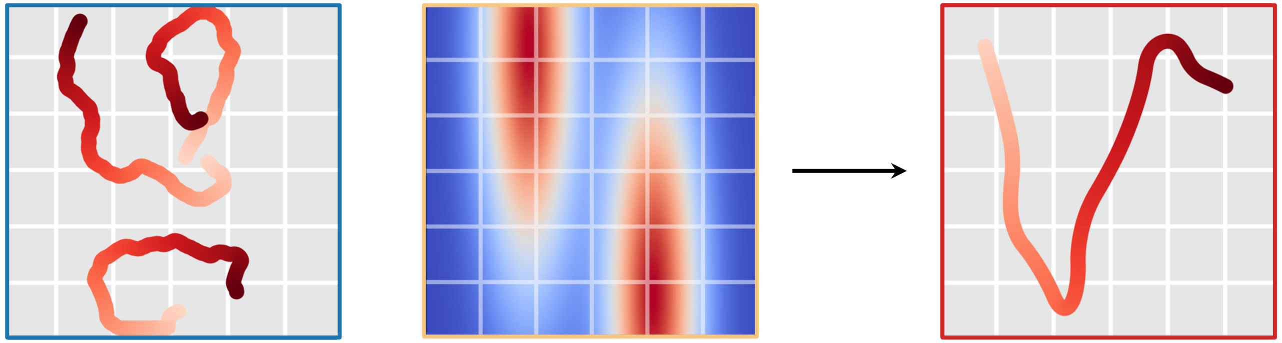


See also: Model Based Planning with Energy Based Models: Du at al, CoRL 2019.

Flexible Behavior Synthesis through Composing Distributions

- Synthesize different behaviors through conditional trajectory synthesis with different learned guidance functions:

$$\underbrace{\tilde{p}_\theta(\boldsymbol{\tau})}_{\text{Behavior Model}} \propto \underbrace{p_\theta(\boldsymbol{\tau})}_{\text{Diffusion Model}} \underbrace{h(\boldsymbol{\tau})}_{\text{Guidance Function}}$$



See also: Model Based Planning with Energy Based Models: Du at al, CoRL 2019.

Flexible Behavior Synthesis through Composing Distributions

- Synthesize different behaviors through conditional trajectory synthesis with different learned guidance functions:

$$\underbrace{\tilde{p}_\theta(\boldsymbol{\tau})}_{\text{Behavior Model}} \propto \underbrace{p_\theta(\boldsymbol{\tau})}_{\text{Diffusion Model}} \; \underbrace{h(\boldsymbol{\tau})}_{\text{Guidance Function}}$$

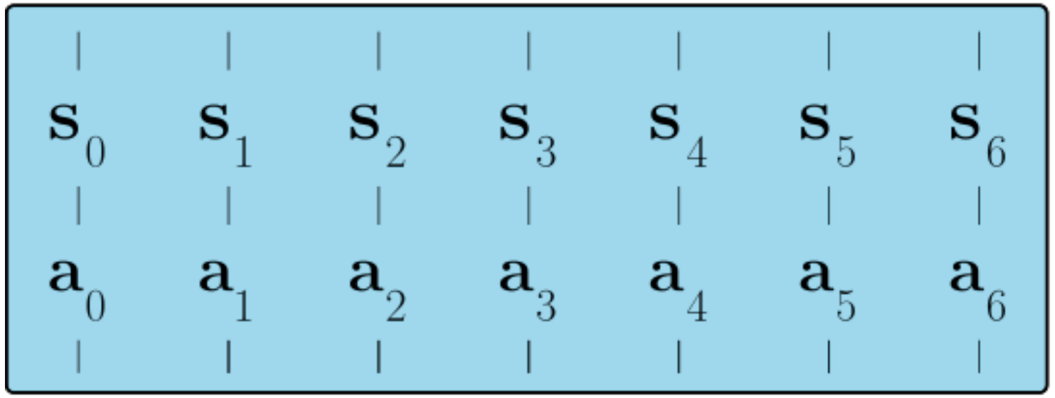
- Guidance functions transforms an unconditional trajectory model into a conditional policy for diverse tasks.

See also: Model Based Planning with Energy Based Models: Du at al, CoRL 2019.

Offline Reinforcement Learning through Value Guidance

- Can use a value function to bias trajectory model to particular task

$p_\theta(\boldsymbol{\tau})$



Offline Reinforcement Learning through Value Guidance

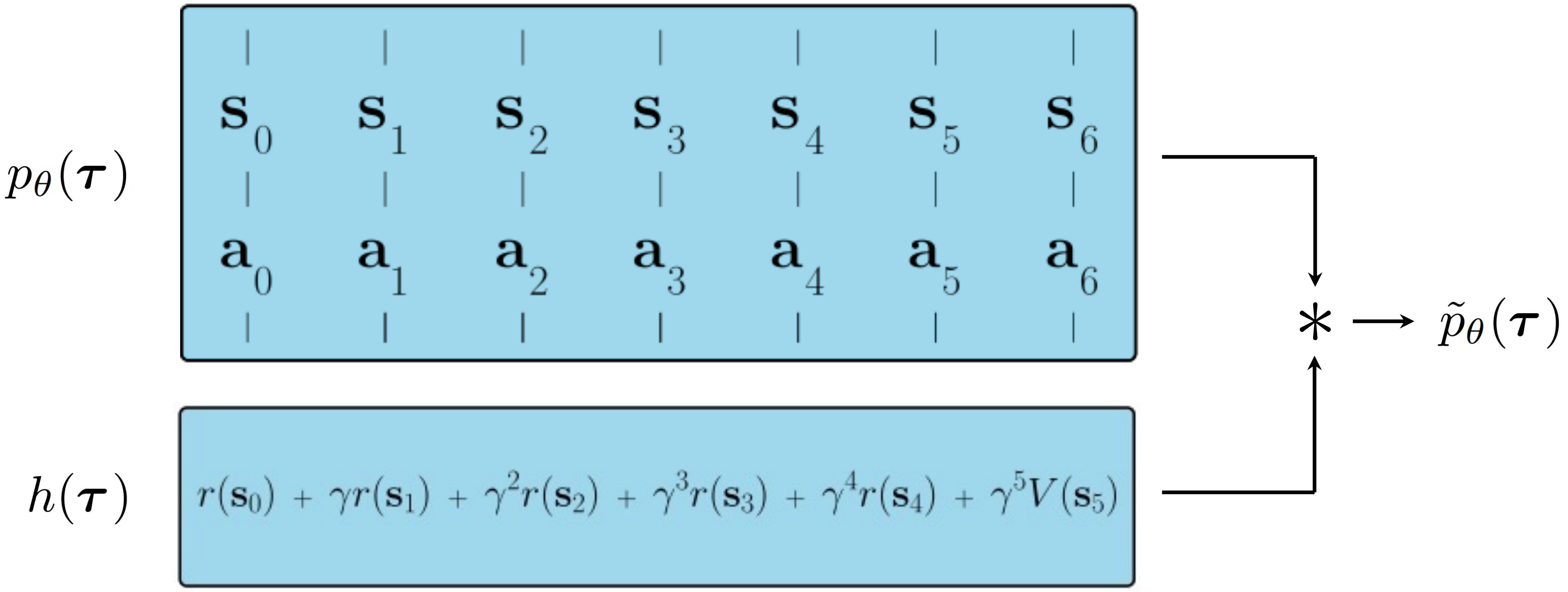
- Can use a value function to bias trajectory model to particular task

$$p_\theta(\boldsymbol{\tau}) \quad \begin{bmatrix} | & | & | & | & | & | & | \\ \mathbf{s}_0 & \mathbf{s}_1 & \mathbf{s}_2 & \mathbf{s}_3 & \mathbf{s}_4 & \mathbf{s}_5 & \mathbf{s}_6 \\ | & | & | & | & | & | & | \\ \mathbf{a}_0 & \mathbf{a}_1 & \mathbf{a}_2 & \mathbf{a}_3 & \mathbf{a}_4 & \mathbf{a}_5 & \mathbf{a}_6 \\ | & | & | & | & | & | & | \end{bmatrix}$$

$$h(\boldsymbol{\tau}) \quad r(\mathbf{s}_0) + \gamma r(\mathbf{s}_1) + \gamma^2 r(\mathbf{s}_2) + \gamma^3 r(\mathbf{s}_3) + \gamma^4 r(\mathbf{s}_4) + \gamma^5 V(\mathbf{s}_5)$$

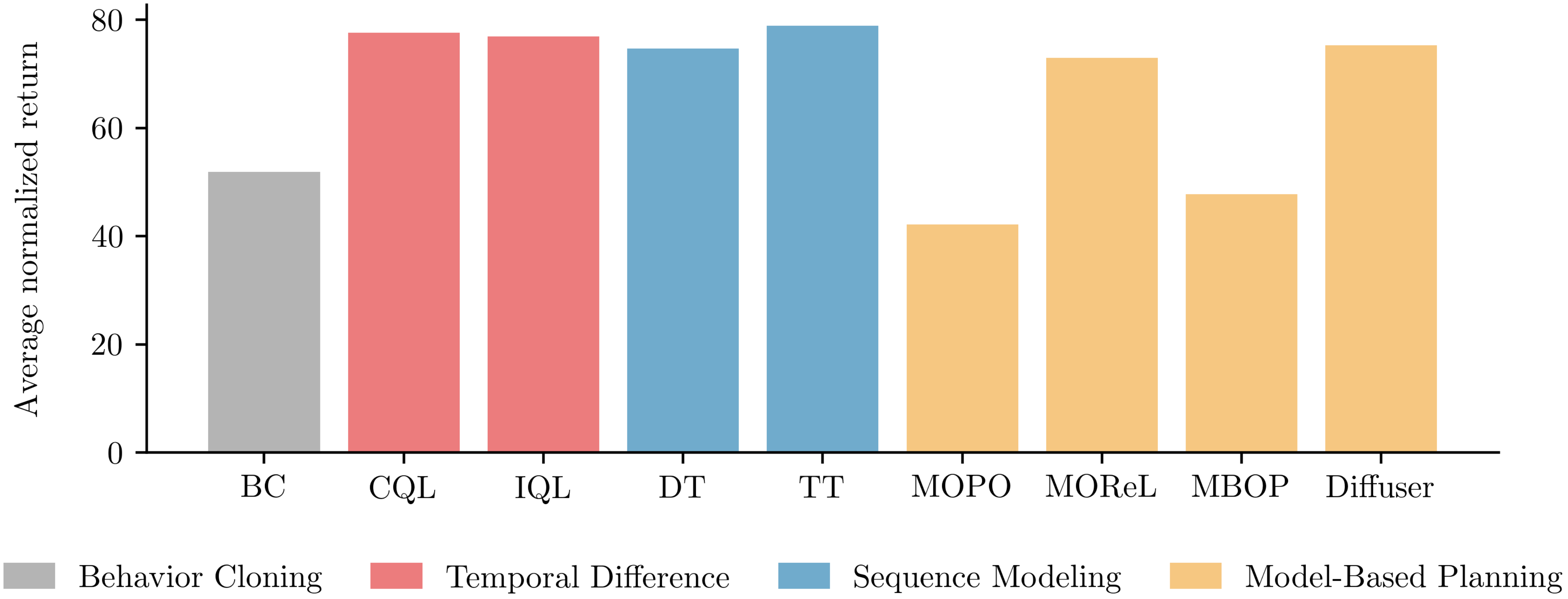
Offline Reinforcement Learning through Value Guidance

- Can use a value function to bias trajectory model to particular task



- Can use a single Diffuser model for multiple different tasks

Offline Reinforcement Learning through Value Guidance



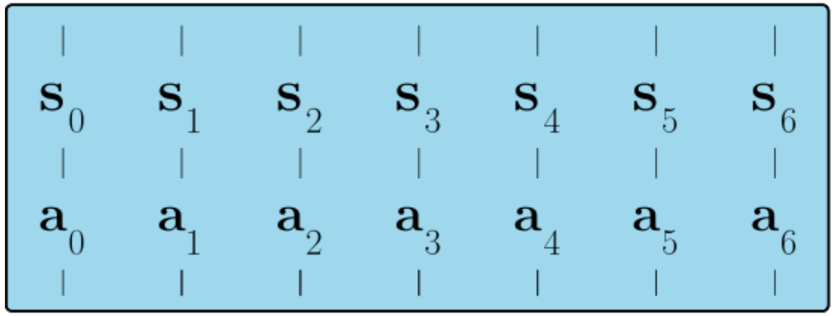
Goal Planning through Inpainting

- Specify a guidance function over the first and goal state of a trajectory

Goal Planning through Inpainting

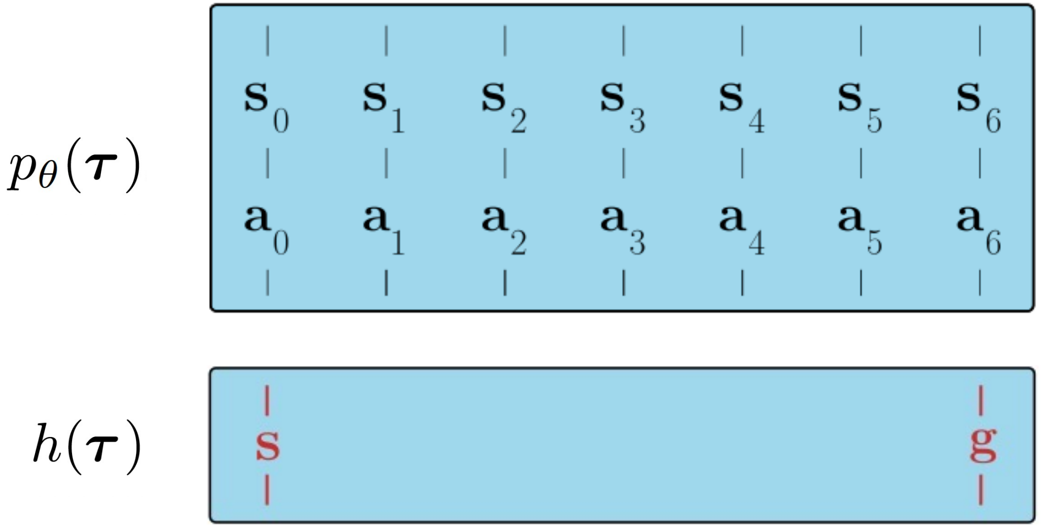
- Specify a guidance function over the first and goal state of a trajectory

$p_\theta(\boldsymbol{\tau})$



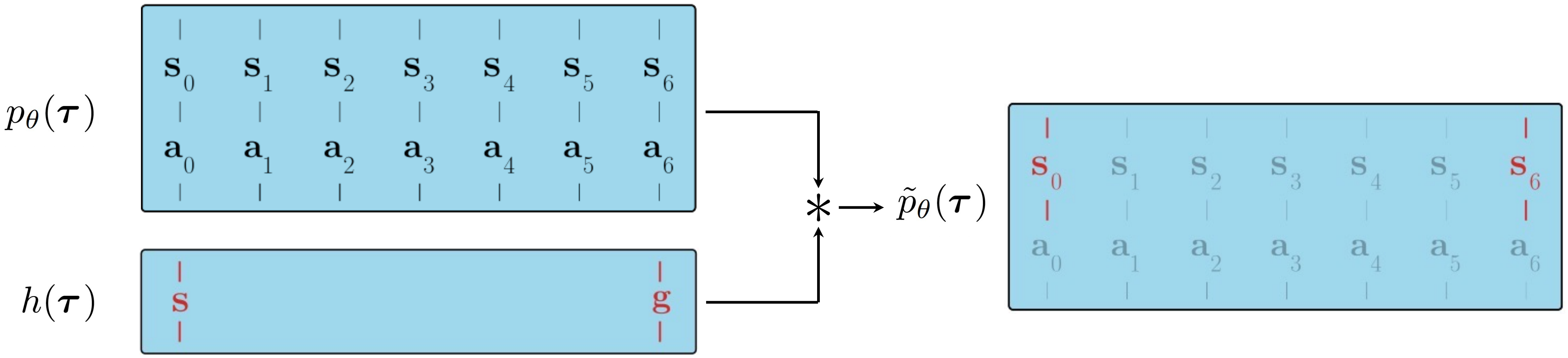
Goal Planning through Inpainting

- Specify a guidance function over the first and goal state of a trajectory



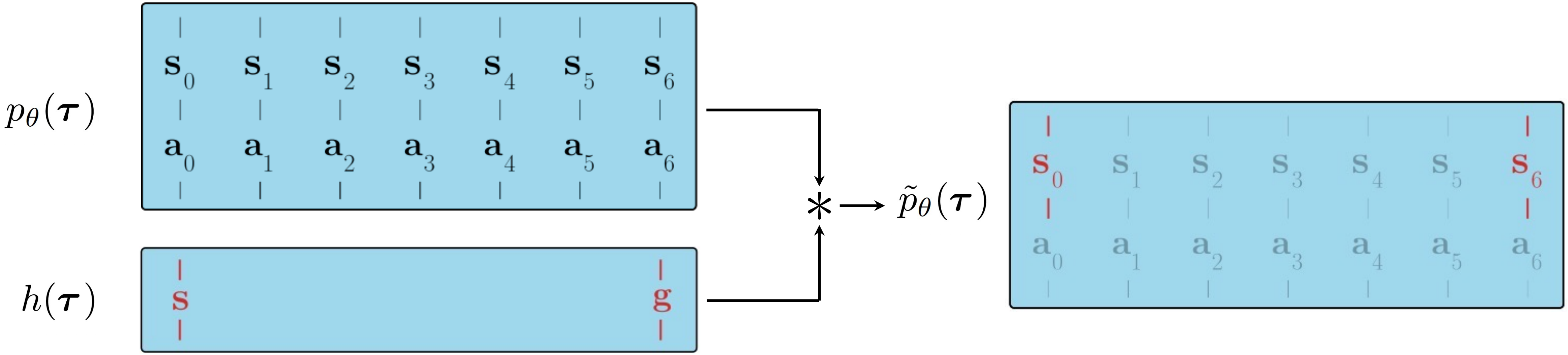
Goal Planning through Inpainting

- Specify a guidance function over the final explicit goal state of a trajectory



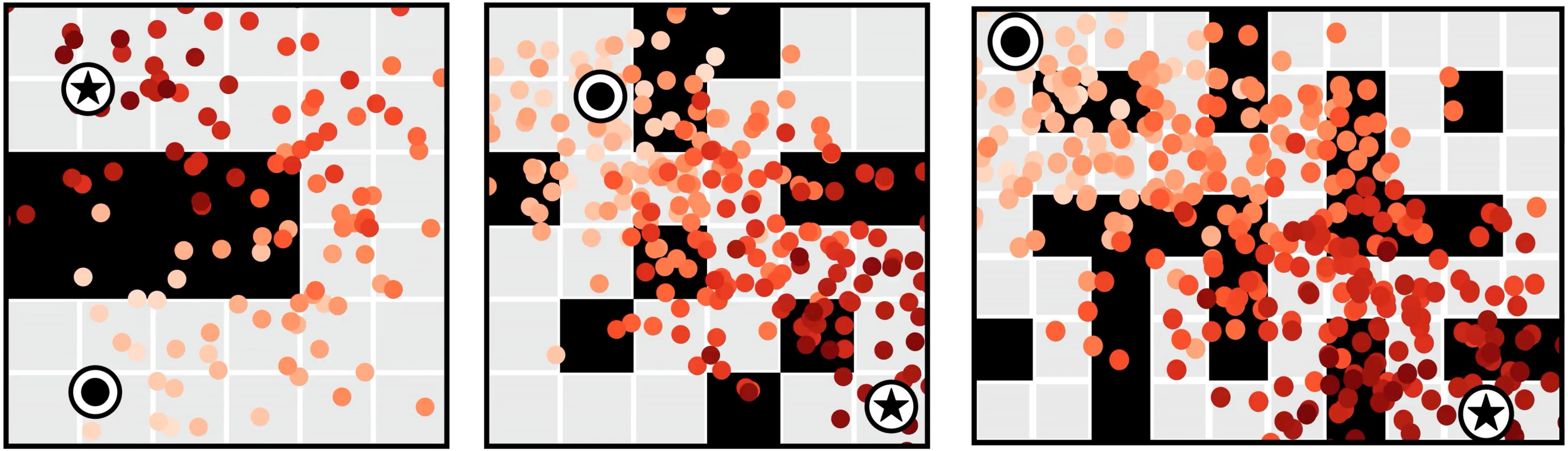
Goal Planning through Inpainting

- Specify a guidance function over the final explicit goal state of a trajectory

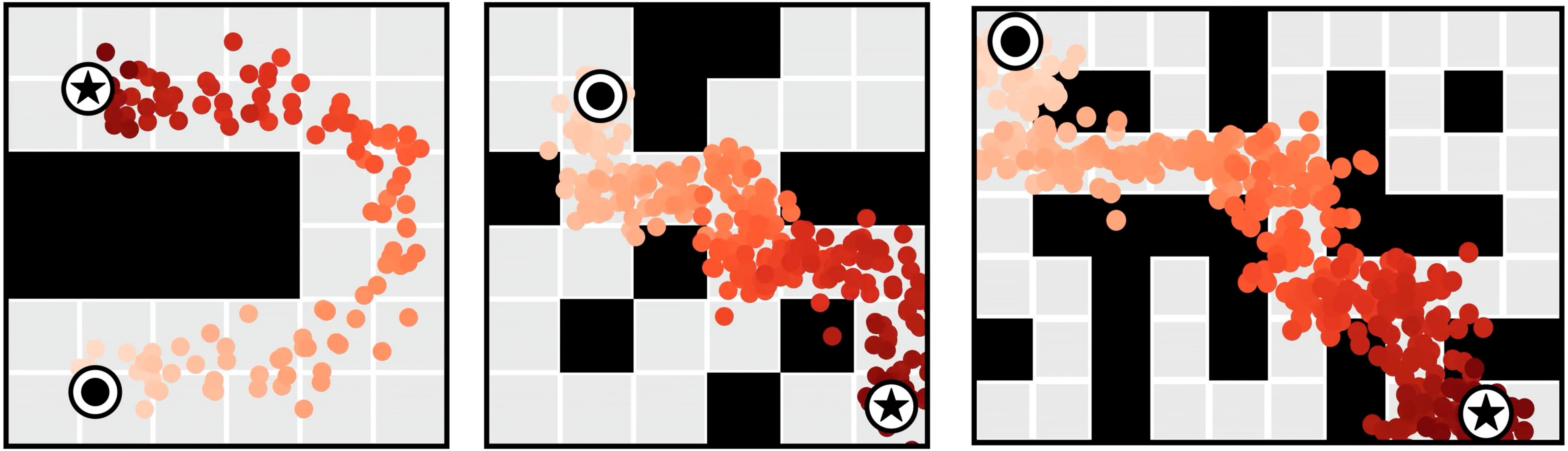


- Construct a goal seeking policy through guidance

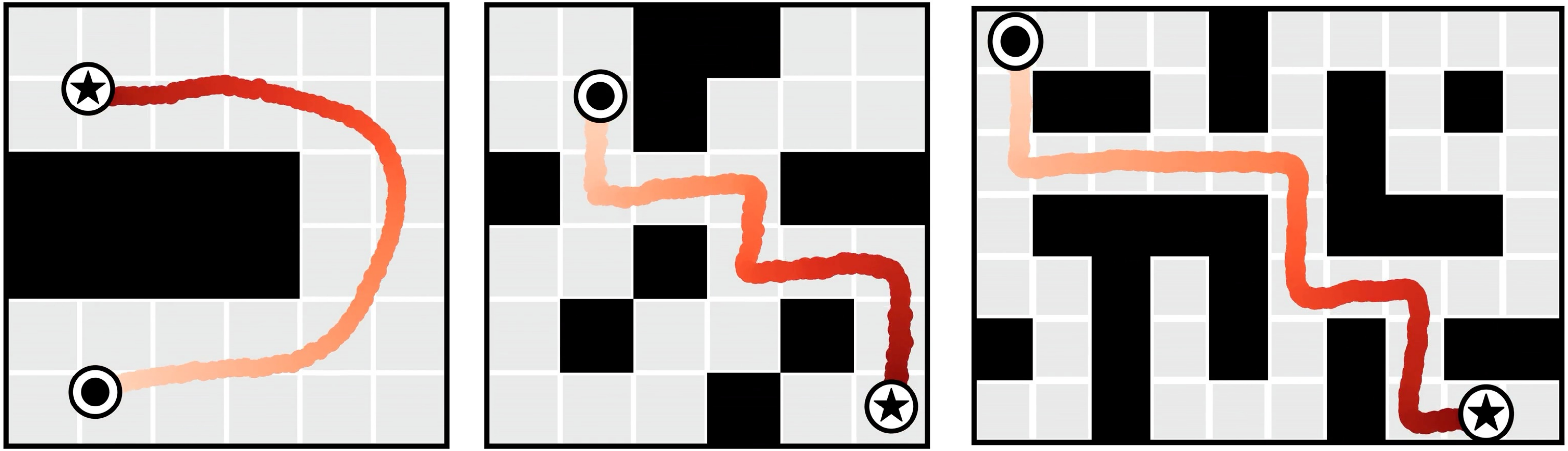
Goal Planning through Inpainting



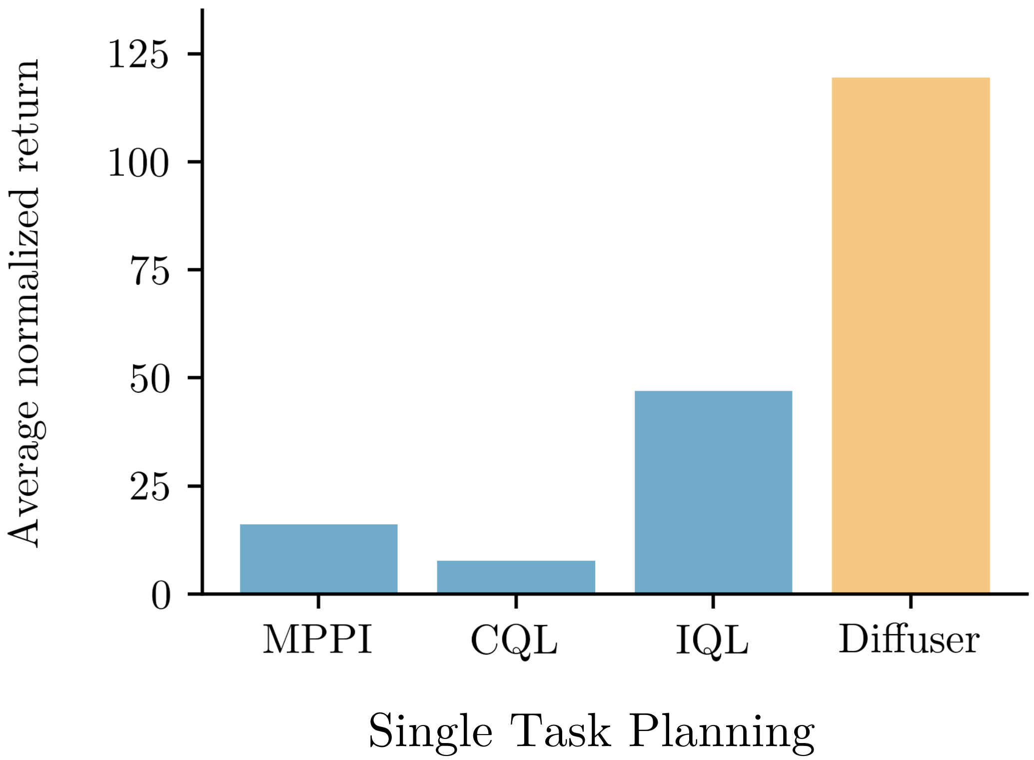
Goal Planning through Inpainting



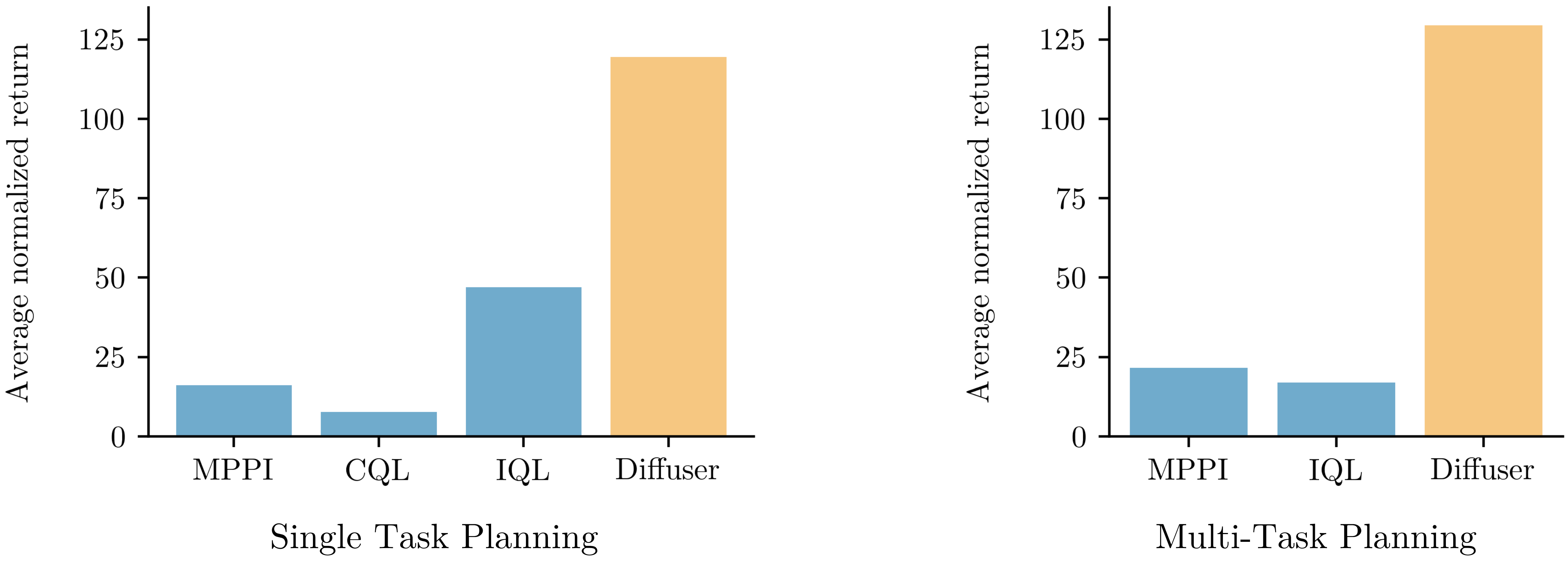
Goal Planning through Inpainting



Goal Planning through Inpainting



Goal Planning through Inpainting



Test-Time Cost Functions

- Can use guidance function to specify arbitrary costs on a trajectory

Test-Time Cost Functions

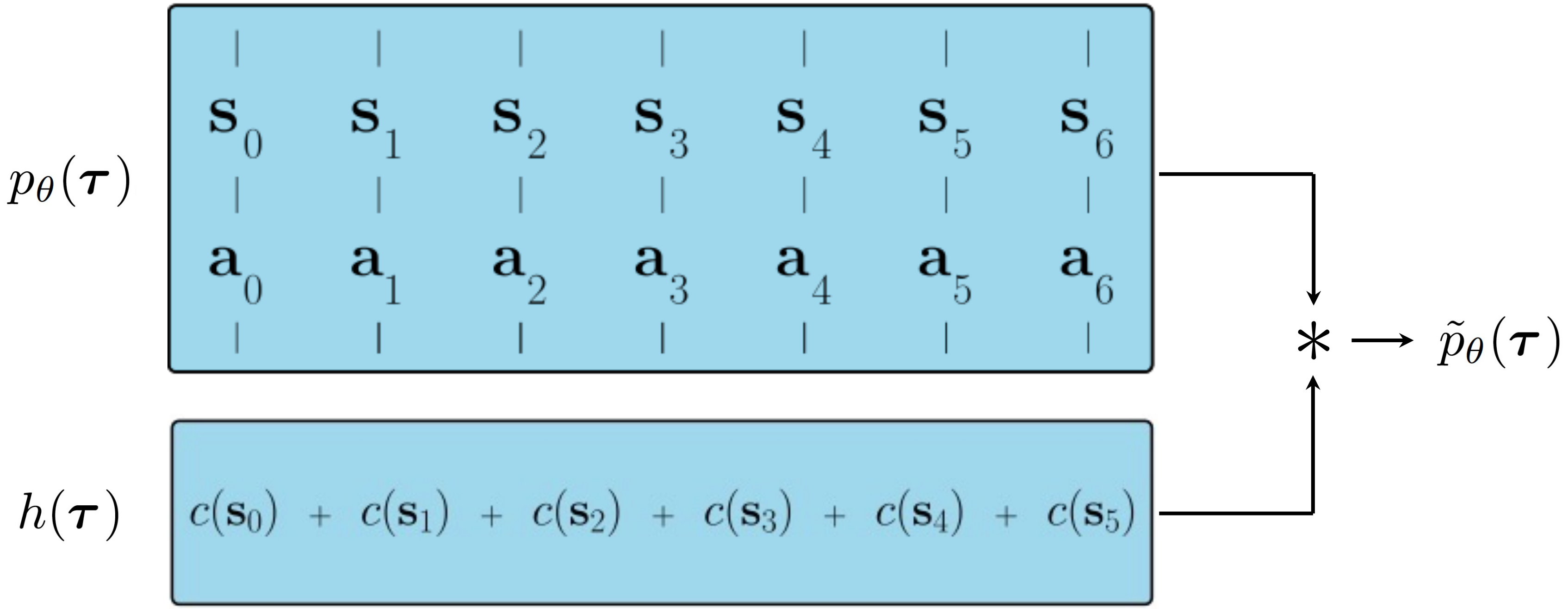
- Can use guidance function to specify arbitrary costs on a trajectory

$$p_\theta(\boldsymbol{\tau}) \quad \begin{bmatrix} \mathbf{s}_0 & \mathbf{s}_1 & \mathbf{s}_2 & \mathbf{s}_3 & \mathbf{s}_4 & \mathbf{s}_5 & \mathbf{s}_6 \\ \mathbf{a}_0 & \mathbf{a}_1 & \mathbf{a}_2 & \mathbf{a}_3 & \mathbf{a}_4 & \mathbf{a}_5 & \mathbf{a}_6 \end{bmatrix}$$

$$h(\boldsymbol{\tau}) \quad \boxed{c(\mathbf{s}_0) + c(\mathbf{s}_1) + c(\mathbf{s}_2) + c(\mathbf{s}_3) + c(\mathbf{s}_4) + c(\mathbf{s}_5)}$$

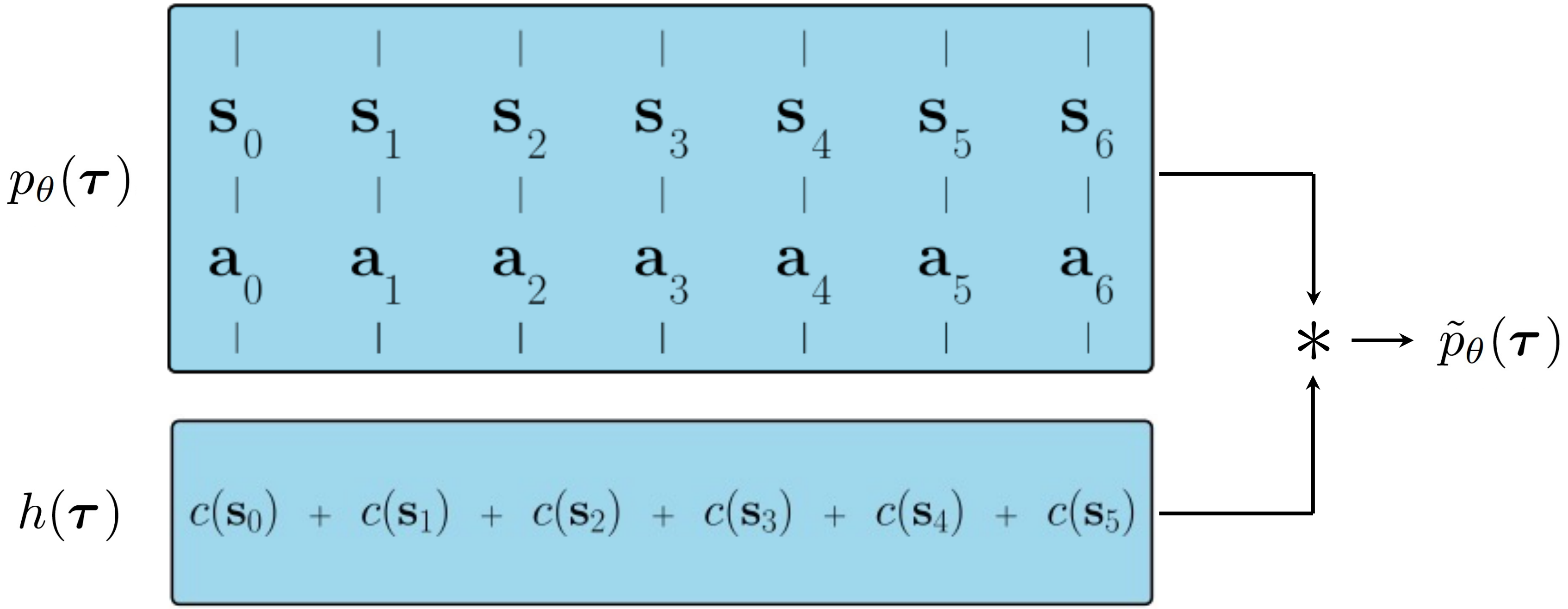
Test-Time Cost Specification

- Can use guidance function to specify arbitrary costs on a trajectory



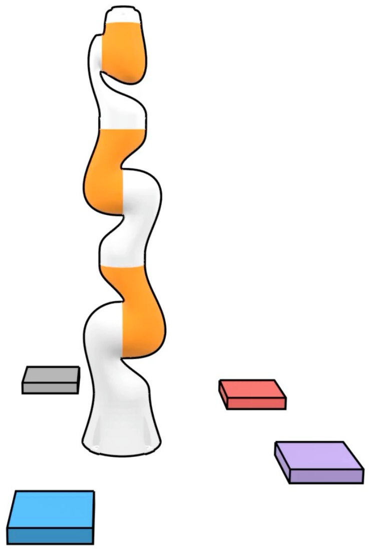
Test-Time Cost Specification

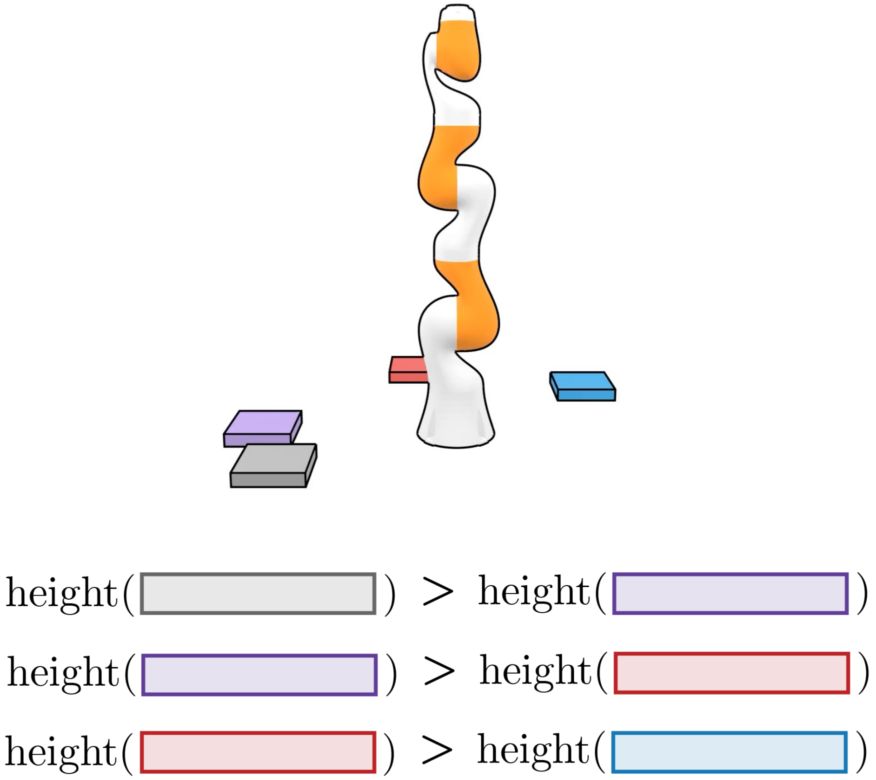
- Can use guidance function to specify arbitrary costs on a trajectory



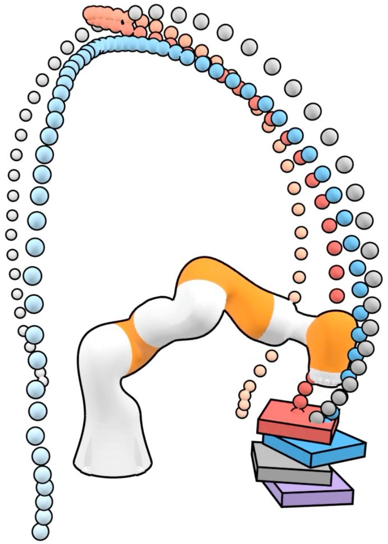
- Can be seen as a learned analogue of trajectory optimization

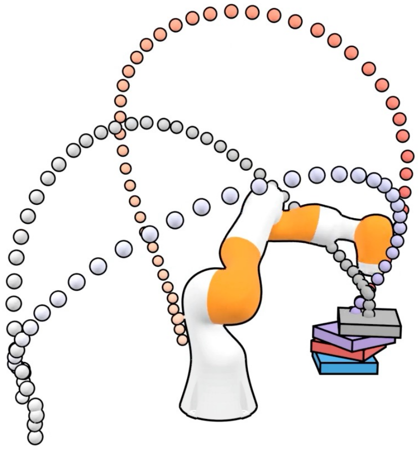
Test-Time Cost Specification





Test-Time Cost Specification



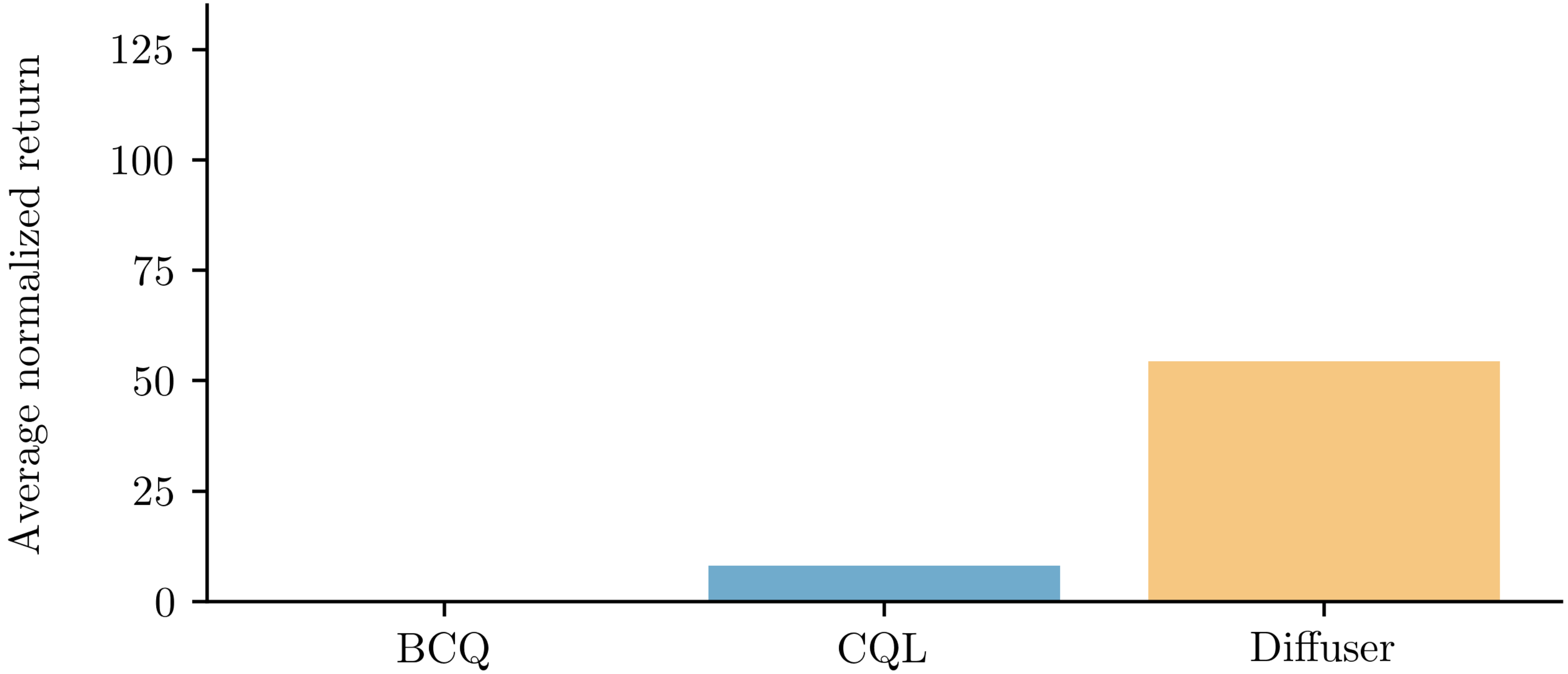


height( ) > height( )

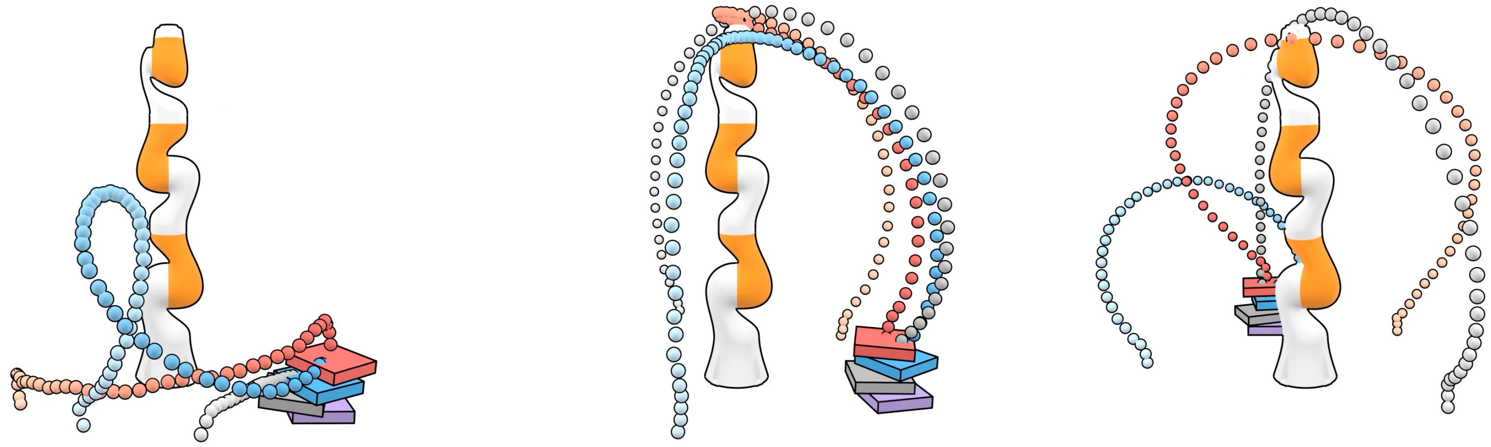
height( ) > height( )

height( ) > height( )

Test-Time Cost Specification



Thanks!



diffusion-planning.github.io