Neurosymbolic World Models for Sequential Decision Making

Leonardo Hernández Cano[1], Maxine Perroni-Scharf[1], Neil Dhir[2], Arun Ramamurthy[2], Armando Solar-Lezama[1]

[1] MIT [2]Siemens

What this work is about

SWMPO is a machine-learning system that uses **Finite State Automata and Neural Networks to model hybrid dynamical systems** (i.e., continuous with discrete components).

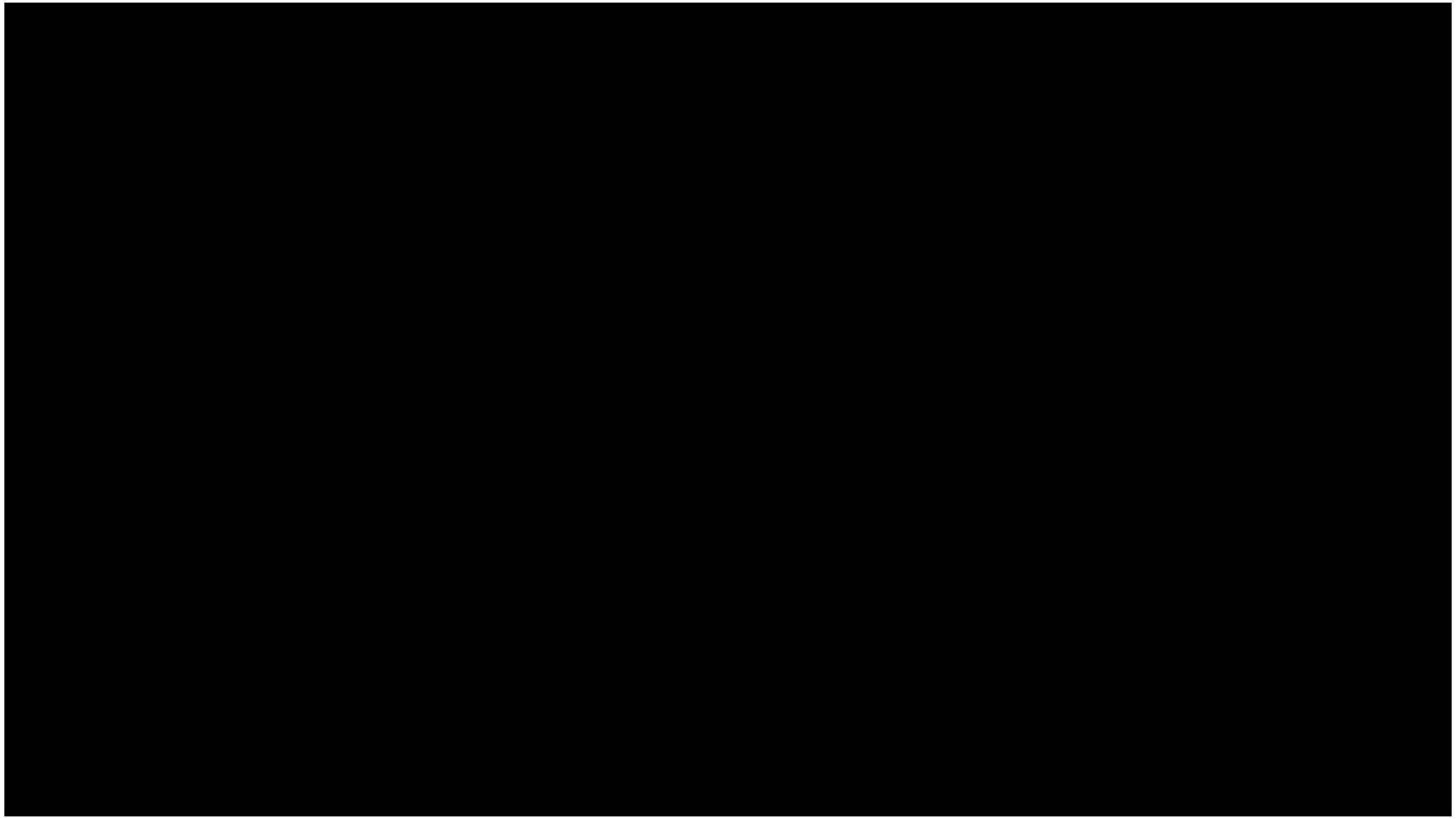
Why learn models

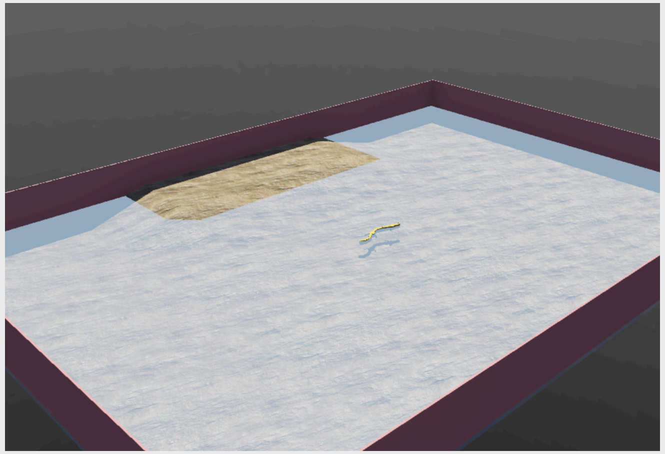
Models are good for planning and learning.

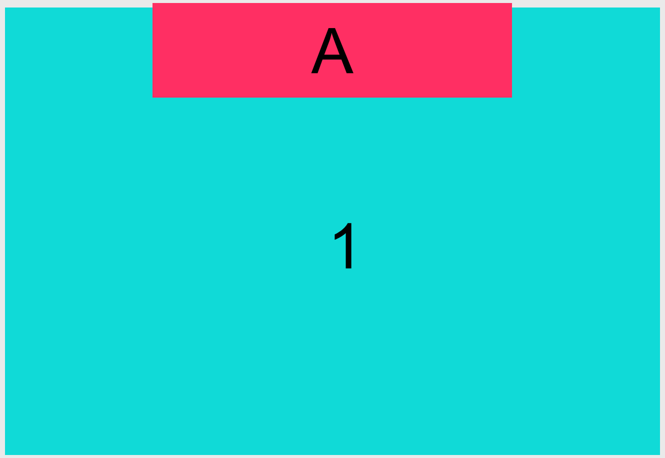
Photo by: Nicholas Larsen
"Pilot training innovation: First successful remote simulator training"
https://www.aetc.af.mil
Accessed: May 26, 2025

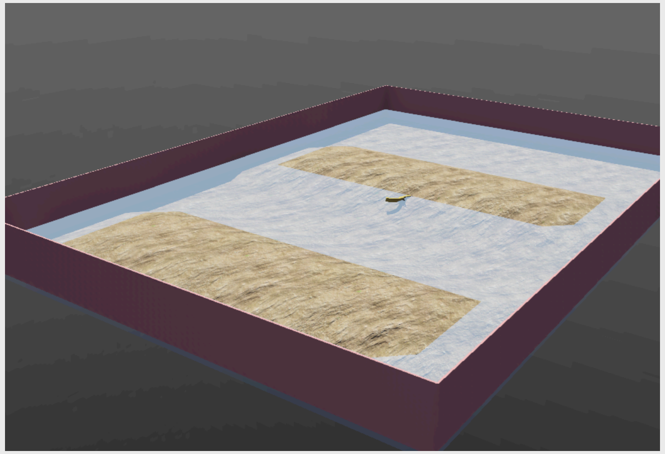
Why SWMPO

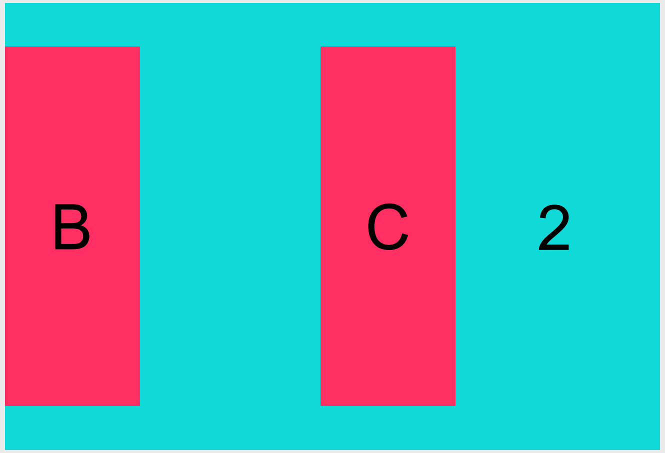
Discover high-level structure and reuse components.





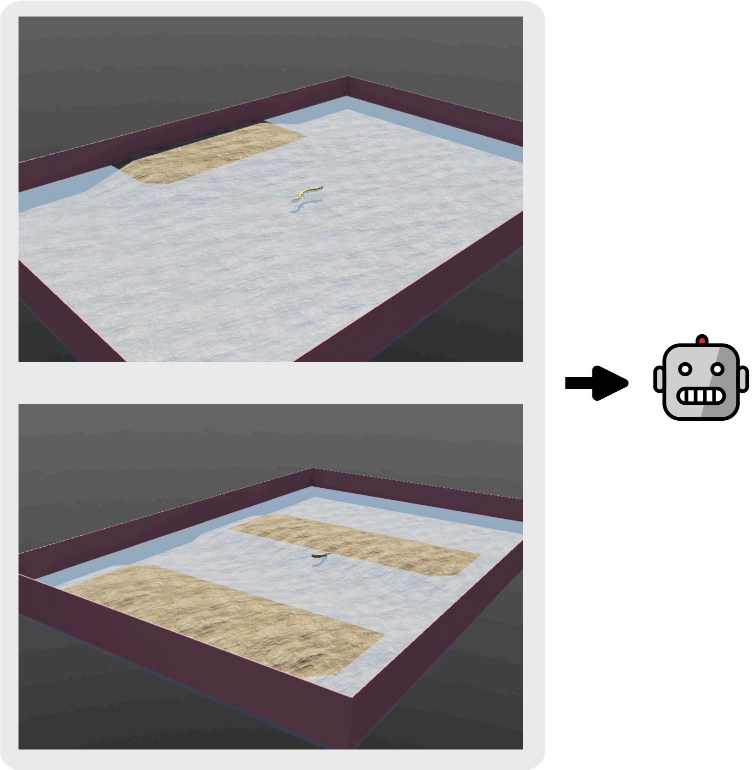


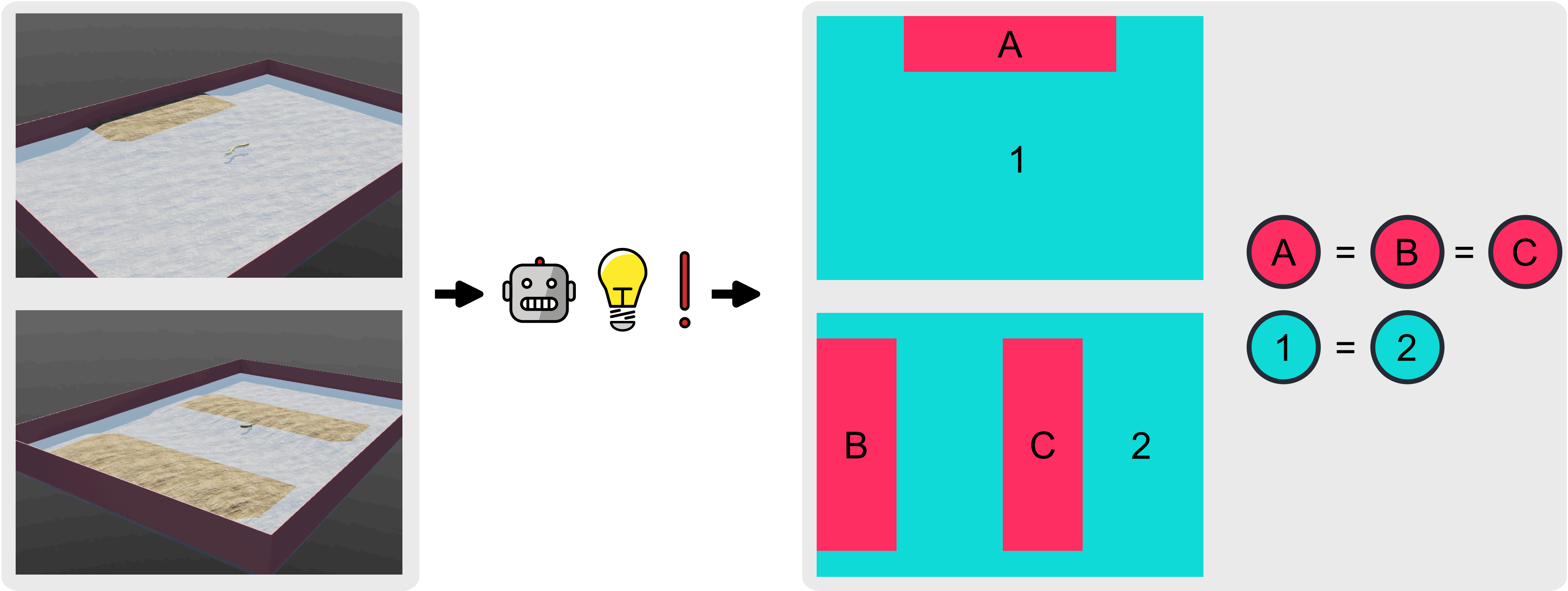


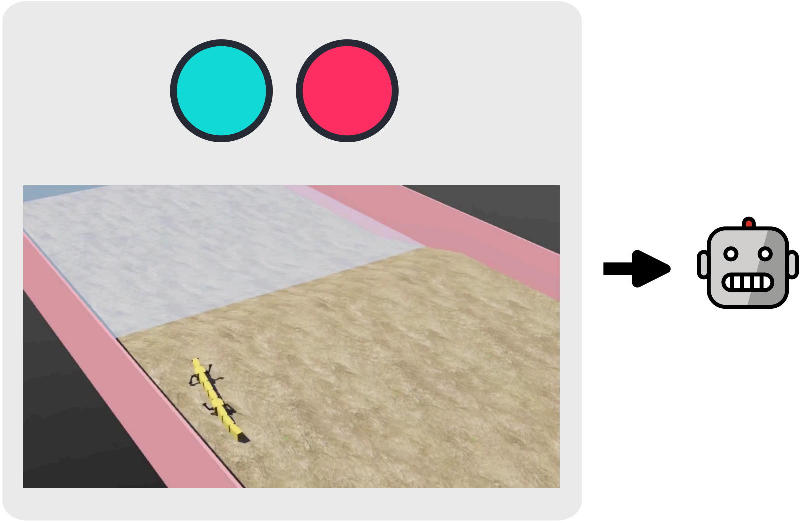


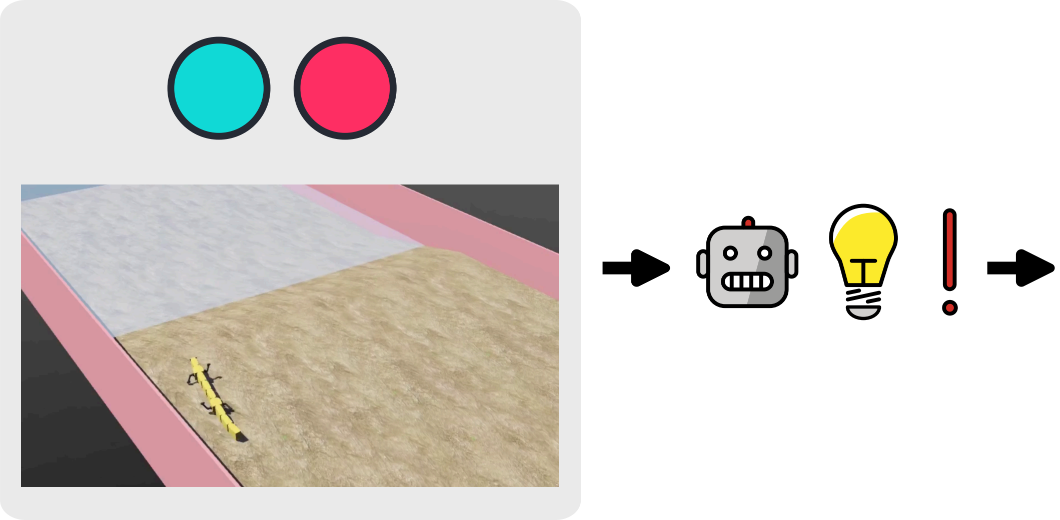
A = B = C

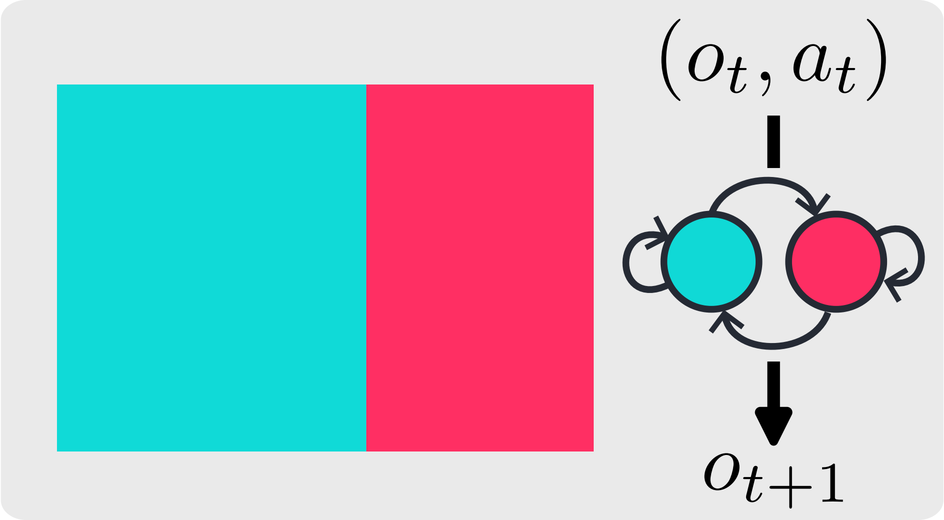
1 = 2

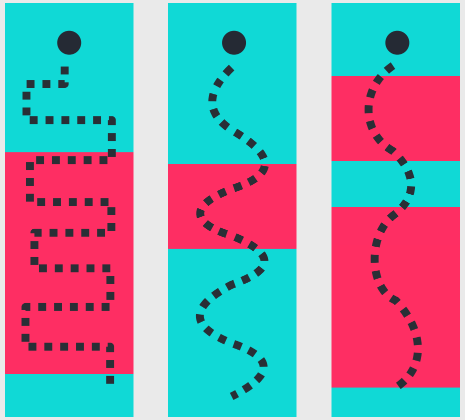



A
1
B
C
2
A = B = C
1 = 2



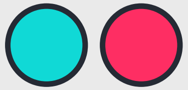



(o_t, a_t)
o_{t+1}



Offline data

NeuralPrimitives()



(1) Neural world-model primitives

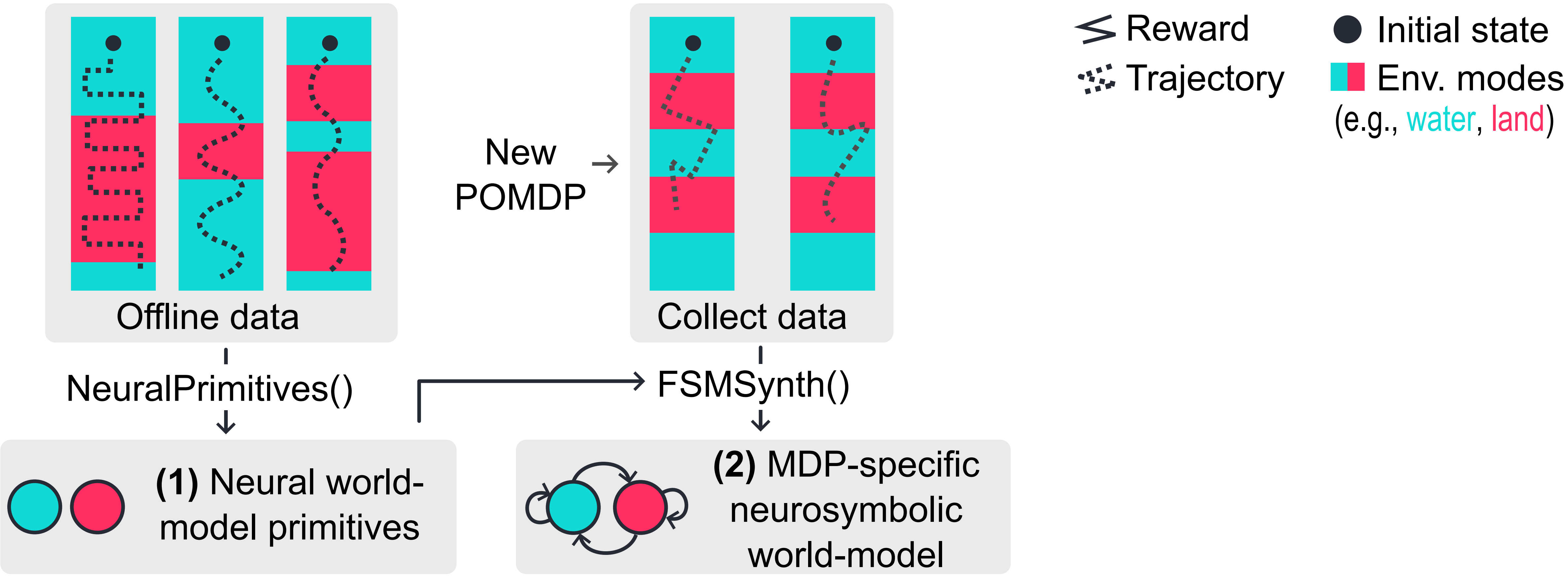
Reward

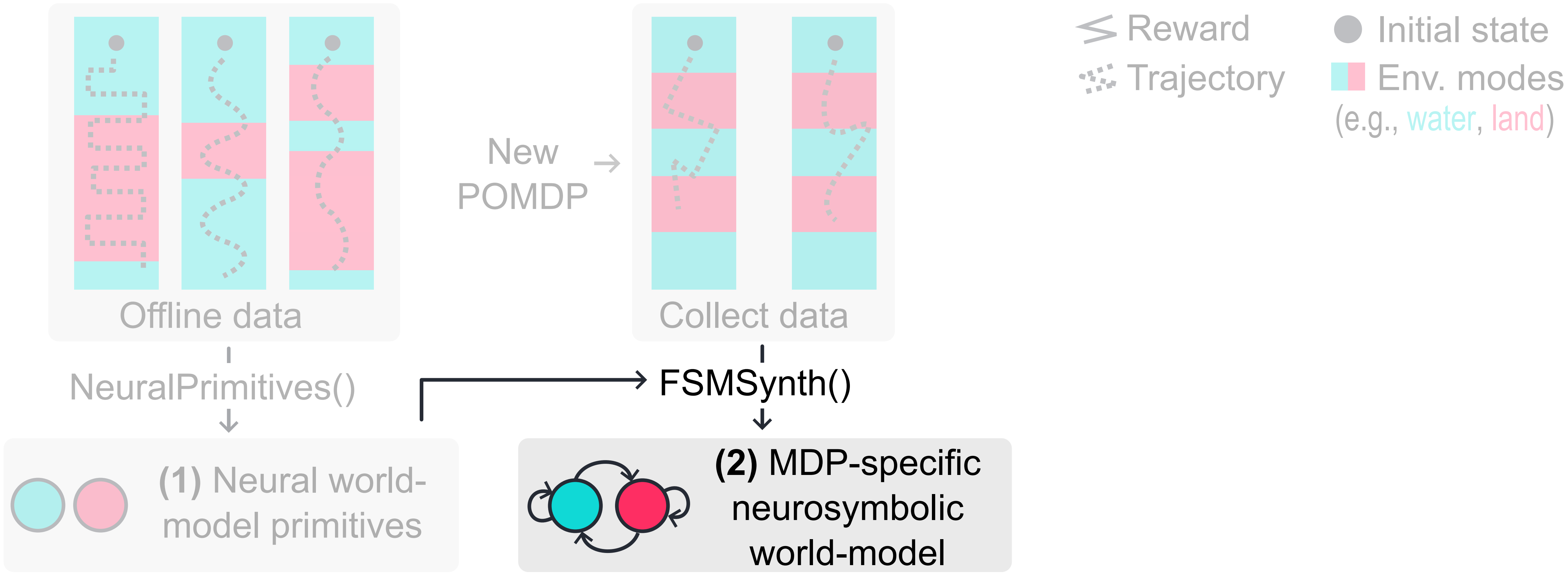
Trajectory

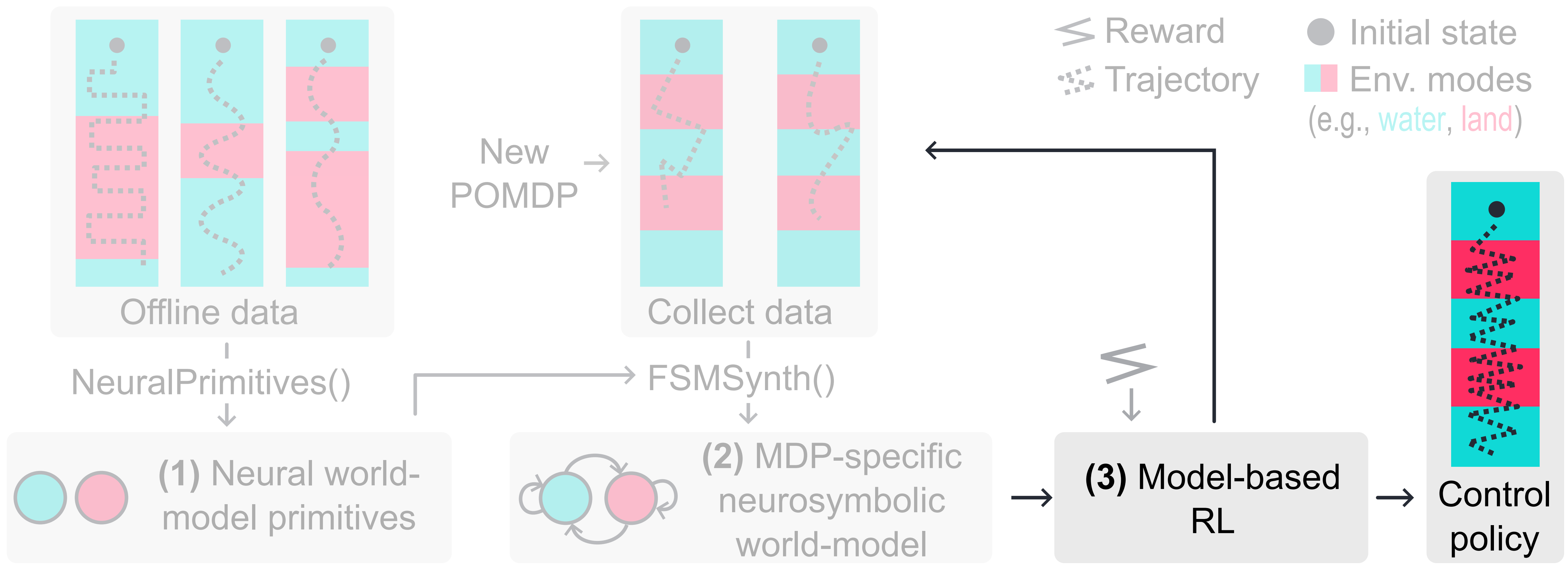
Initial state

Env. modes (e.g., water, land)


Offline data
NeuralPrimitives()
(1) Neural world-model primitives
Reward
Initial state
Trajectory
Env. modes
(e.g., water, land)


Offline data
New POMDP
Collect data
NeuralPrimitives()
FSMSynth()
(1) Neural world-model primitives
(2) MDP-specific neurosymbolic world-model
Reward
Initial state
Trajectory
Env. modes
(e.g., water, land)


Offline data
New POMDP
Collect data
NeuralPrimitives()
FSMSynth()
(1) Neural world-model primitives
(2) MDP-specific neurosymbolic world-model
Reward
Trajectory
Initial state
Env. modes
(e.g., water, land)

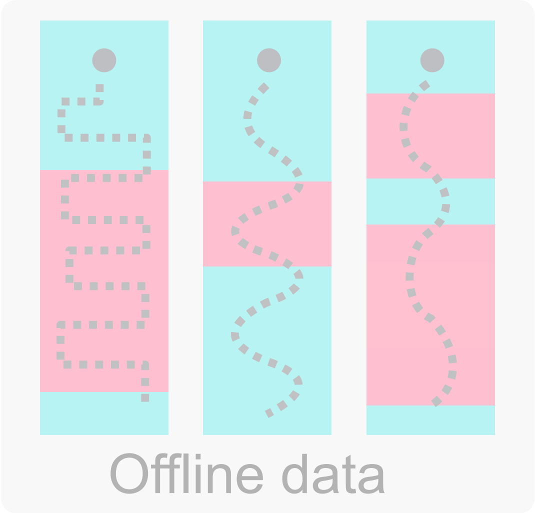


Reward
Trajectory
Initial state
Env. modes
(e.g., water, land)
Offline data
New POMDP
Collect data
NeuralPrimitives()
FSMSynth()
(1) Neural world-model primitives
(2) MDP-specific neurosymbolic world-model
(3) Model-based RL
Control policy

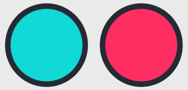
Prior art

Lots of prior work.

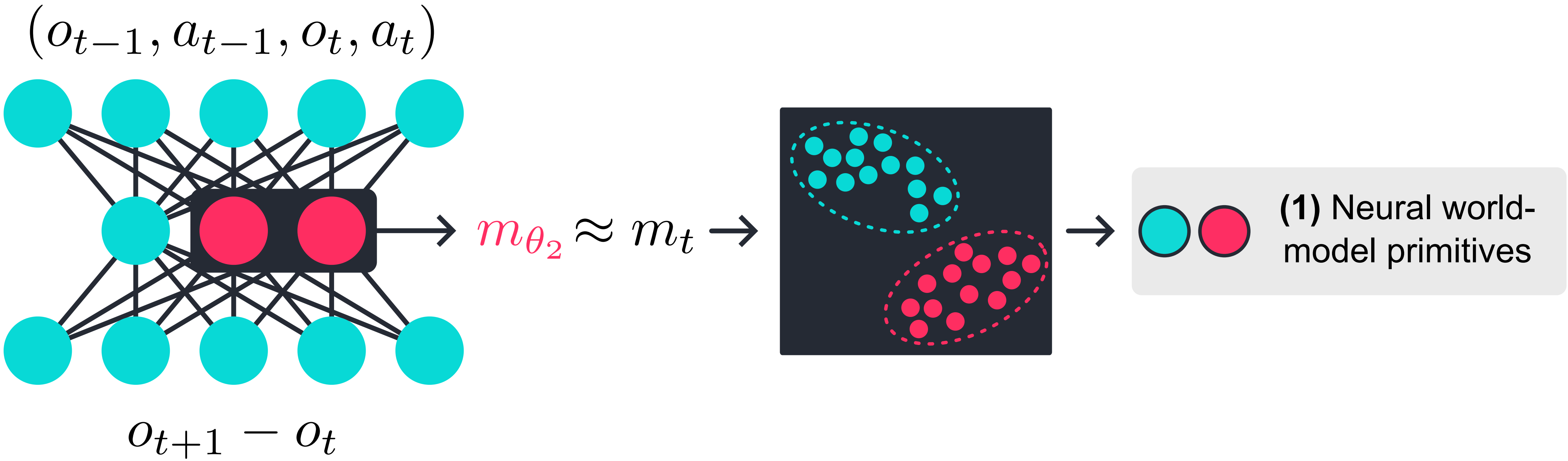
- Automata from pretrained vision models (Hasanbeig et al, 2021)
- Hidden Markov Models or similar (e.g., rSLDS Glaser et al, 2020)
- Automata with Affine Dynamics (Soto et al 2021)

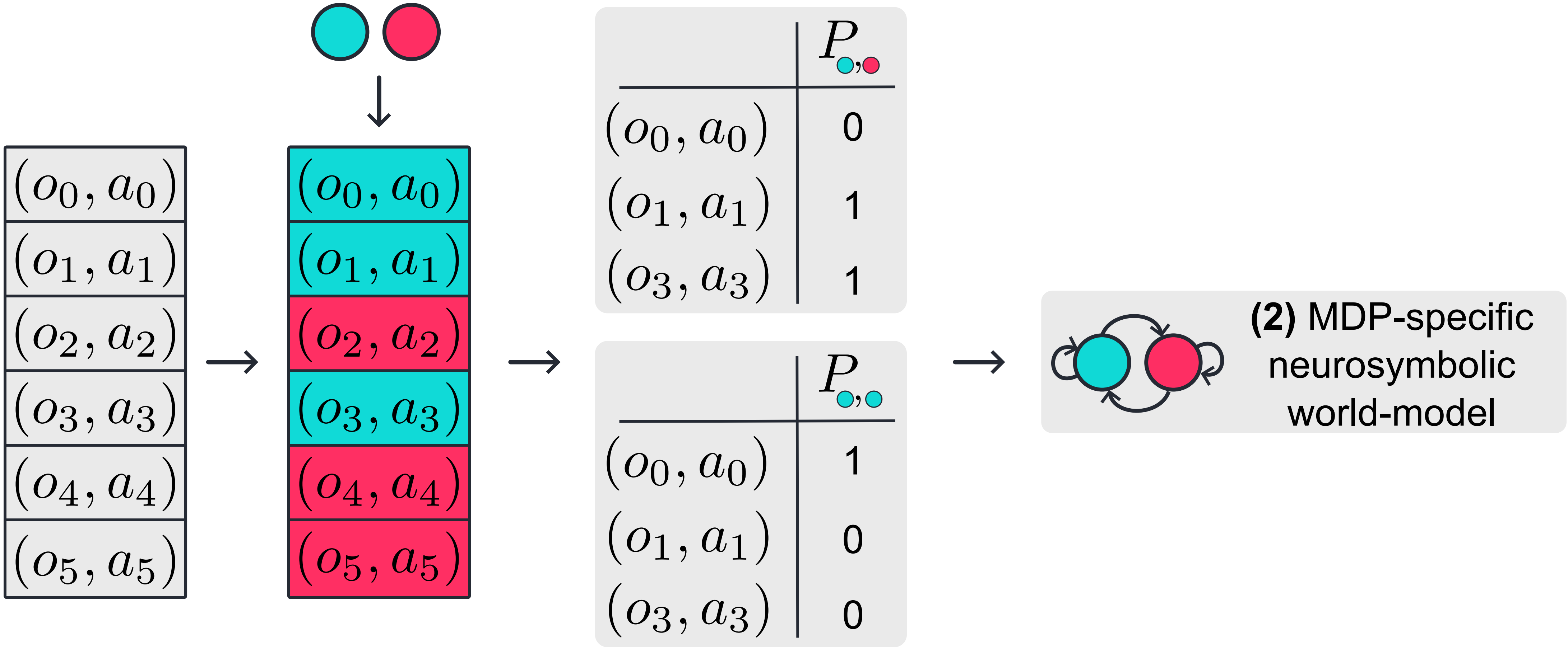


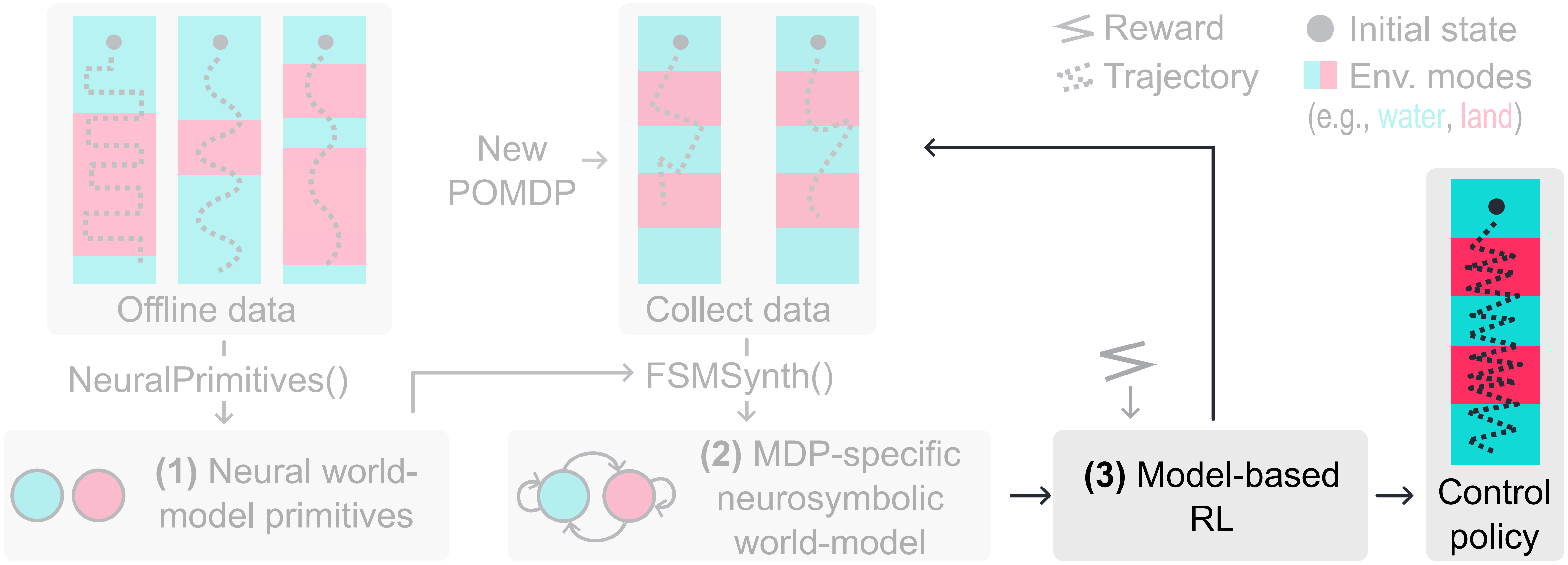
NeuralPrimitives()



(1) Neural world-model primitives

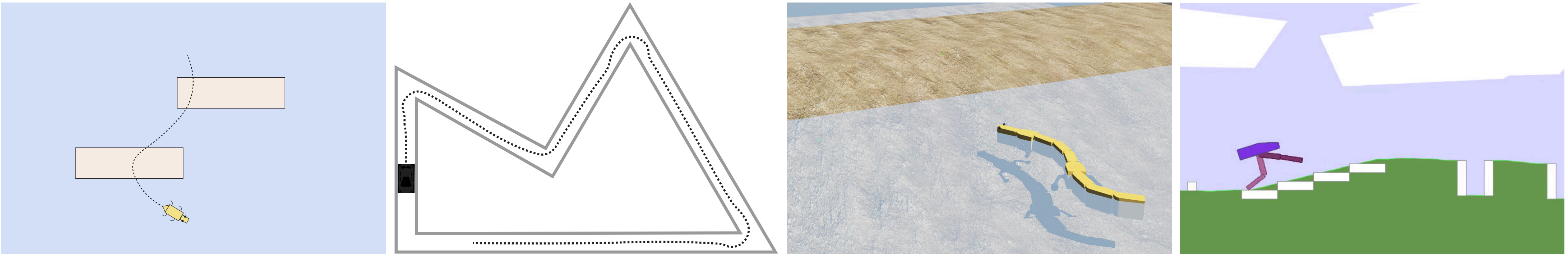

$(o_{t-1}, a_{t-1}, o_t, a_t)$
$o_{t+1} - o_t$
$m_{\theta_2} \approx m_t$
(1) Neural world-model primitives


(o_0, a_0)
(o_1, a_1)
(o_2, a_2)
(o_3, a_3)
(o_4, a_4)
(o_5, a_5)
P
(o_0, a_0) 0
(o_1, a_1) 1
(o_3, a_3) 1
P
(o_0, a_0) 1
(o_1, a_1) 0
(o_3, a_3) 0
(2) MDP-specific neurosymbolic world-model


Reward
Initial state
Trajectory
Env. modes
(e.g., water, land)
Offline data
New POMDP
Collect data
NeuralPrimitives()
FSMSynth()
(1) Neural world-model primitives
(2) MDP-specific neurosymbolic world-model
(3) Model-based RL
Control policy

Experiments

1. How effectively does SWMPO leverage offline data in the synthesis of environment-specific FSMs?
2. Is the resulting FSM accurate enough for model-based RL?



(a) Point Mass (b) LiDAR Racing (c) Salamander (d) Bipedal Walker

Figure 5. The four simulated evaluation environments.

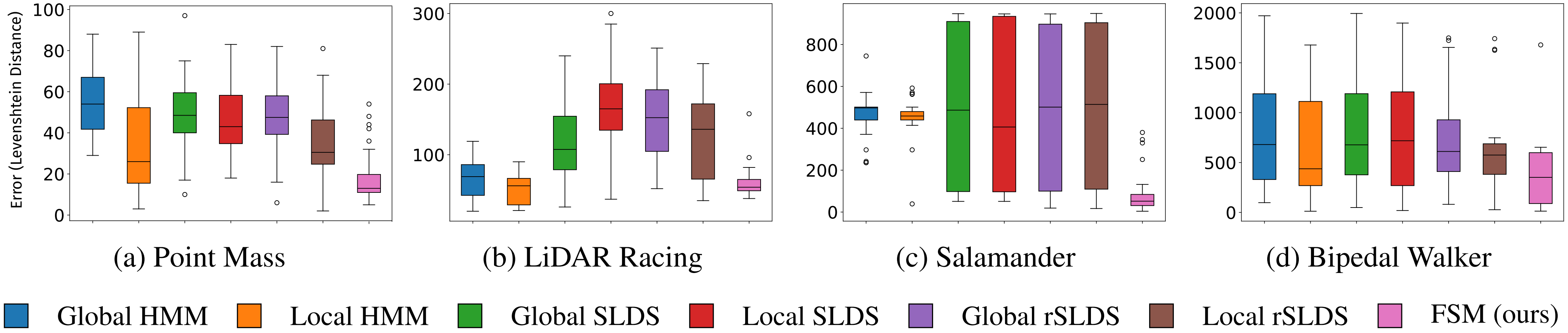


Figure 6. Bottom row: Box plots comparing the Levenshtein distance between predicted and ground truth labels for unseen trajectories (lower is better) for each evaluation environment. Systems that accurately track the latent mode variable of a dynamical system more accurately match the ground-truth labels. The box-plots show that SWMPO (right) outperforms baselines (left) in all environments in tracking the latent mode variable.

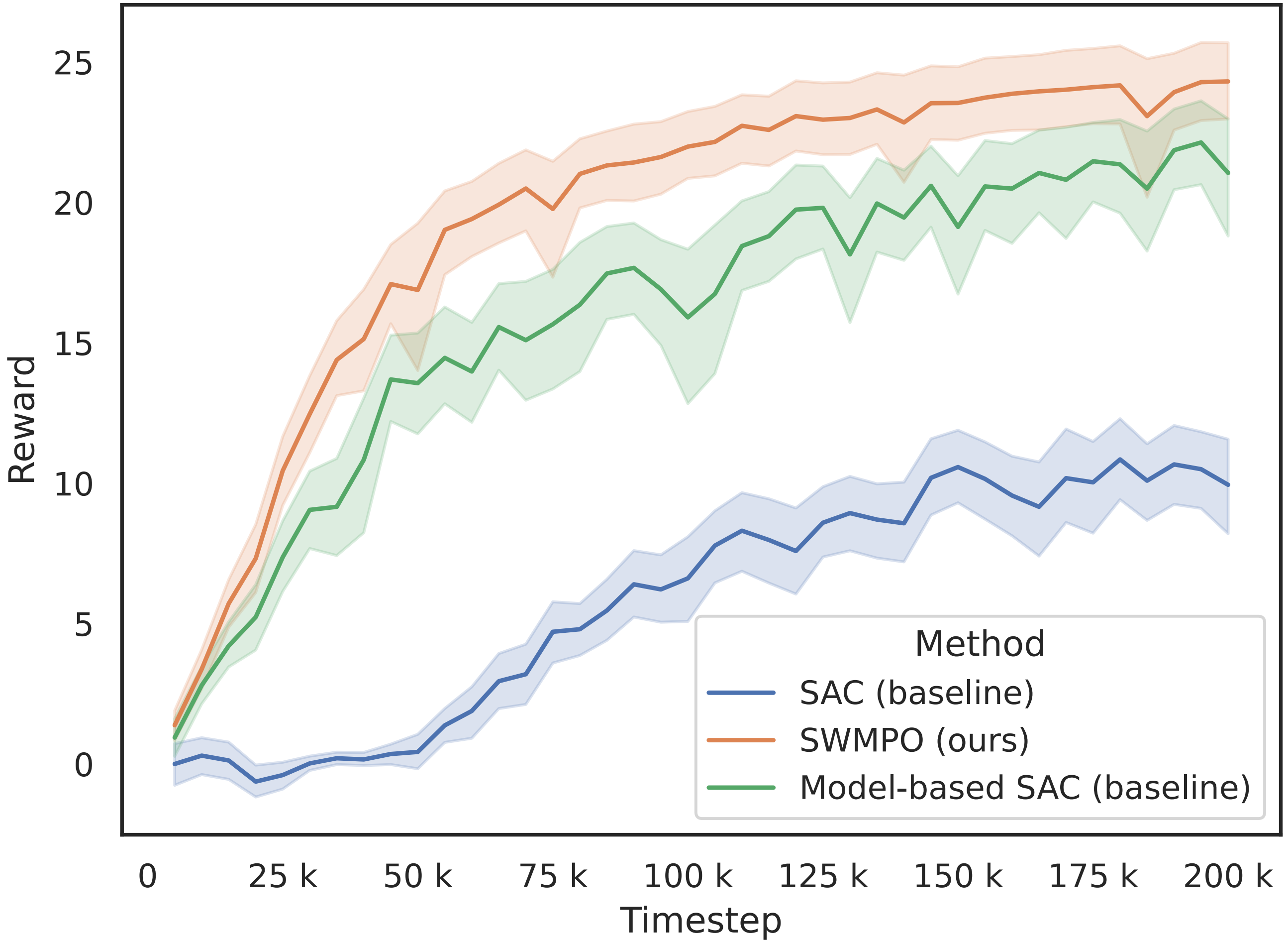


Figure 8. Reward curves in the **Point Mass** environment for SWMPO (ours), baseline model-free RL, and baseline model-based RL with a neural feed-forward model learned online. Shaded area are 95% confidence intervals computed over 64 different random seeds.

Conclusion

- SWMPO: reusable structured world models + policy optimization.
- SWMPO matches or outperforms state of the art in our experiments, but further work is needed to evaluate the framework in more realistic settings.

What's next?

- Apply SWMPO to more environments
- Relax assumptions
- Use more general class of models for the discrete modelling (i.e., Python instead of FSMs)
- Explicitly leverage FSM structure during policy optimization

Thanks! :D