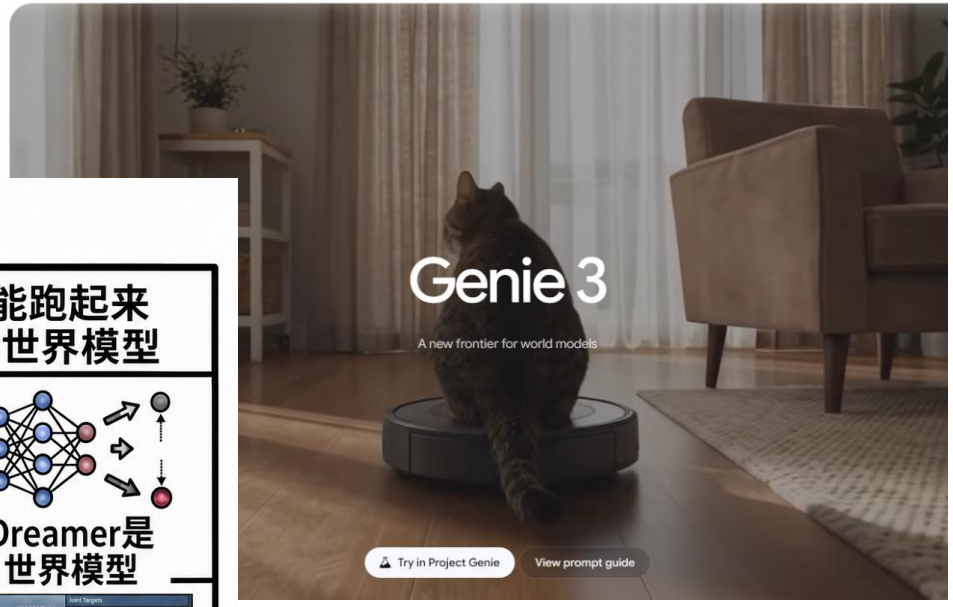
World Models for Planning: From Code World Models to Temporal-Distance JEPA

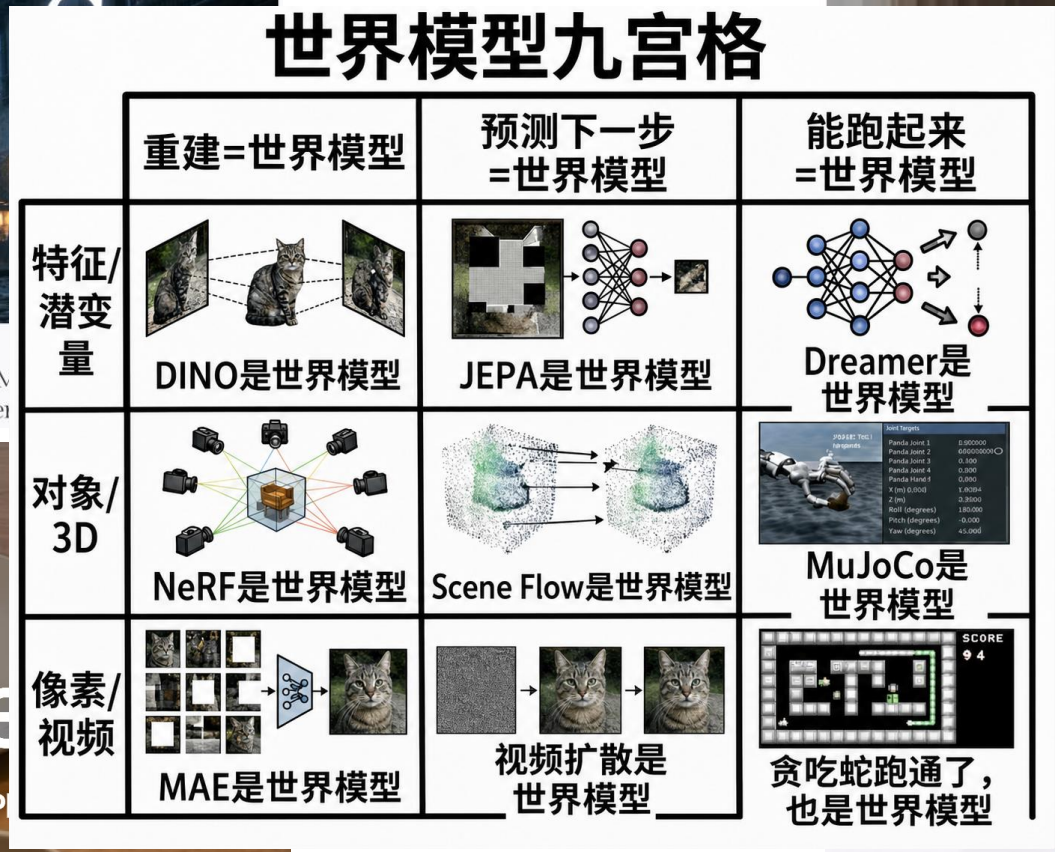
Jiaxin Bai
Assistant Professor, Hong Kong Baptist University

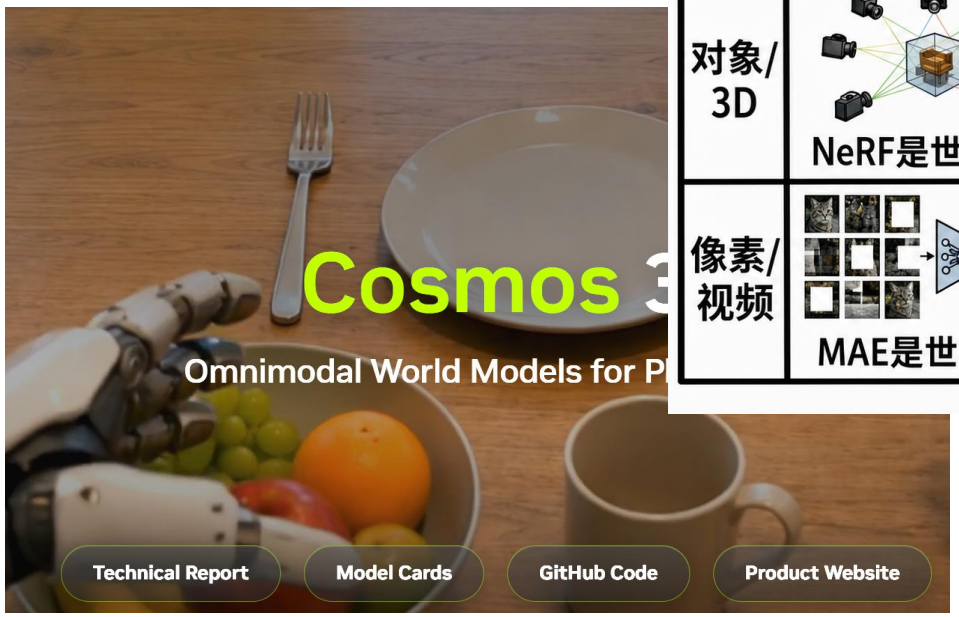
HKUST (GZ) · Nov 2026

What Is a World Model?





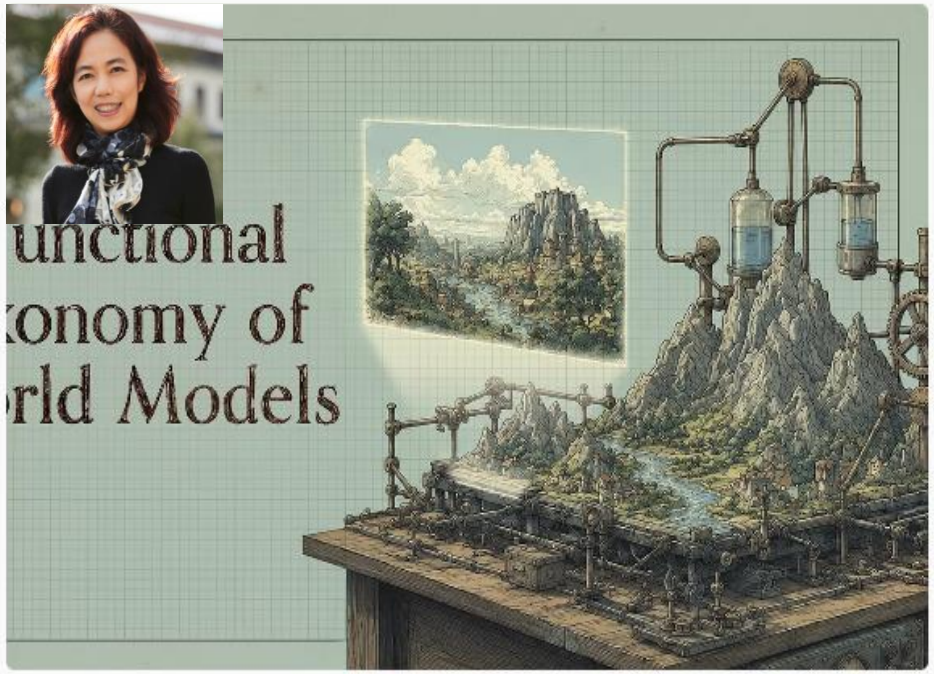




Credit: Internet



What Is a World Model? Renderer, Simulator, Planner



A Functional Taxonomy of World Models

Renderers, Simulators, Planners, and the Loop That Connects Them

JUN 4 • FEI-FEI LI

1,823 123 285



Functional roles:

- Render observations
- **Simulate**
- **Plan actions**

- **Simulation is the key:** explicit state and dynamics make observations and consequences computable.

Source: drfeifei.substack.com

Two Directions: Code Simulation × Latent-Space Planning

Simulate | Render observations | **Plan actions**

(not the focus today)

Code simulation

$$s_{t+1} = f_{\text{code}}(s_t, a_t)$$

+ inspectable, exact rollouts
– needs a parsed state; hand-written cost

Parts I–II

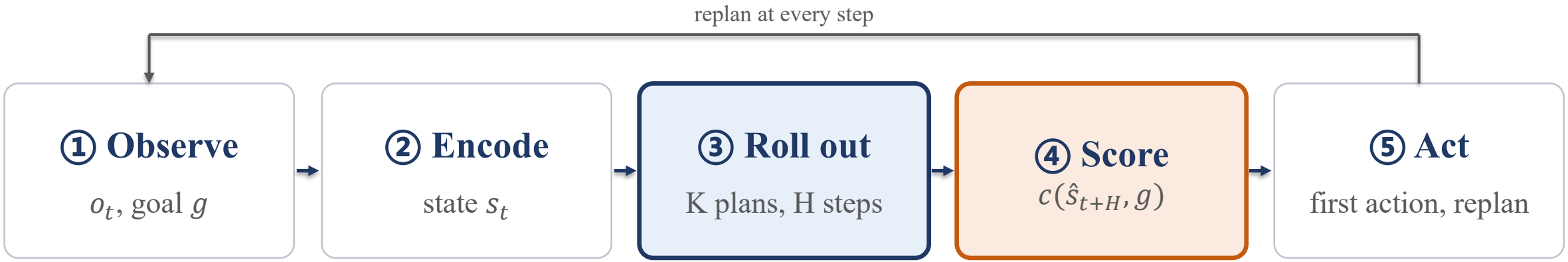
Planning in latent space

$$z_t = \text{Enc}(o_t), \qquad \hat{z}_{t+1} = \text{Pred}(z_t, a_t)$$

+ end-to-end from pixels, scalable
– opaque; ranks futures by plain ℓ_2

Part III

The Planner Loop: Model-Predictive Control



③ Dynamics: what will happen?

④ Cost: which future is closest to the goal?

The world model is used only at ③ and ④: multi-step rollouts and ranking.

Combining the Two in One Planning Loop

	② Encode	③ Roll out	④ Score
Code simulation	object state s_t	**induced program** $f(s_t, a_t)$	hand-written distance
Latent-space planning	latent $z_t = \mathrm{Enc}(o_t)$	neural predictor	**mined cost** d_ψ

Roadmap

Code simulation

Part I

PatchWorld

text environments

③ Dynamics

Part II

VisualPatchWorld

visual control

③ Dynamics

Latent-space planning

Part III

TD-JEPA

visual control

④ Cost

Part I

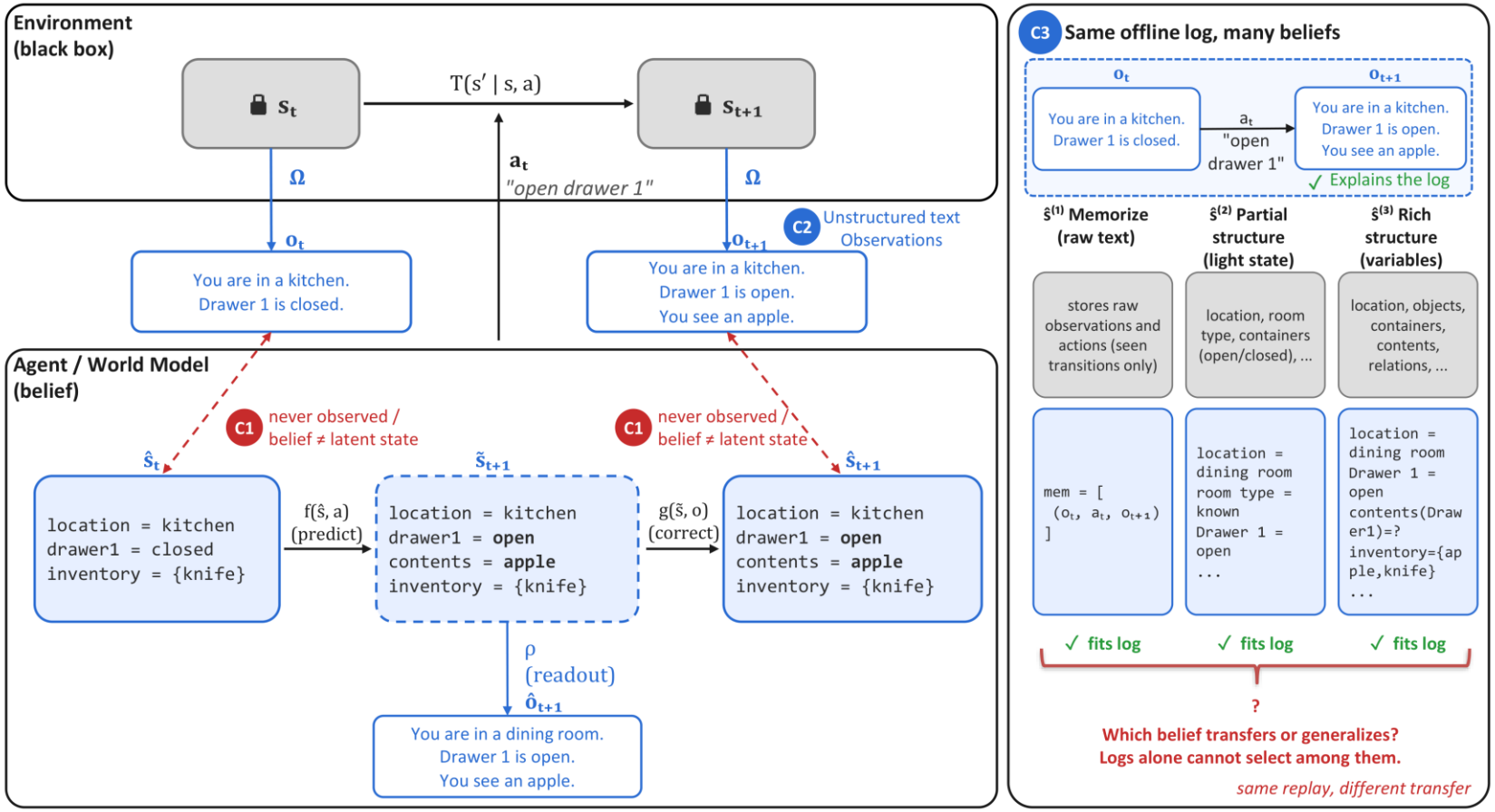
Belief Programs from Black-Box Text Environments

PatchWorld: Gradient-Free Optimization of Executable World Models for Agent Environments

Jiaxin Bai, Yue Guo, Yifei Dong, Jiaxuan Xiong, Tianshi Zheng, Yixia Li, Tianqing Fang, Zihao Wang, Lihui Liu, Jeff Z. Pan, Yangqiu Song, et al.

arXiv:2605.30880 *https://github.com/HKBU-KnowComp/PatchWorld*

Text-Agent Environments Are Black-Box POMDPs



- **Challenge 1** Latent facts may never be directly observable to the agent.
- **Challenge 2** One latent fact can be phrased in many surface forms.
- **Challenge 3** Many programs replay the same logs; logs alone cannot tell which one transfers.

Figure 1, PatchWorld. Environments from AgentGym (Xi et al., 2025).

Data Structures as the Latent, Code as the Transition Law

State is a data structure; the transition law is source code.

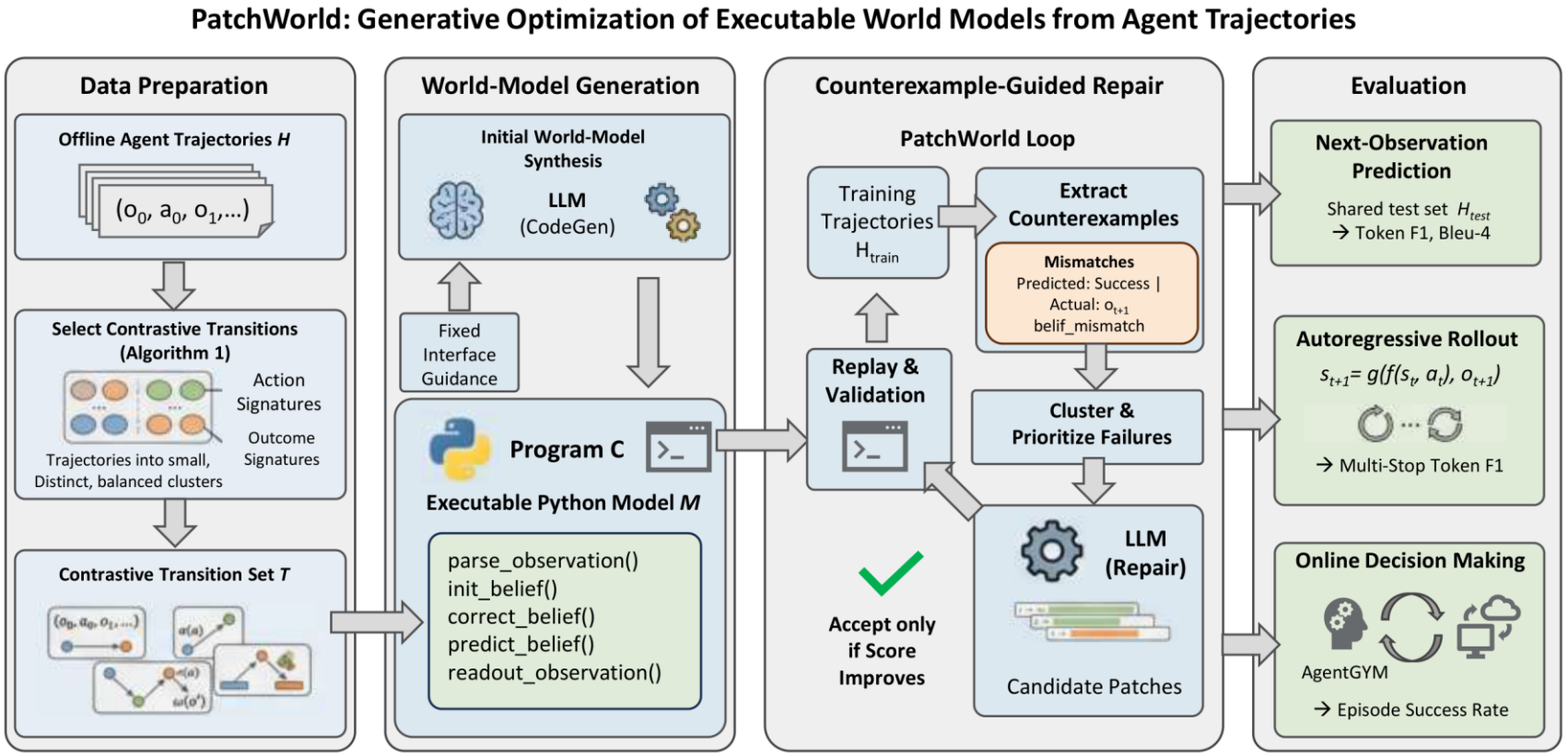
```
class WorldModel:   # induced once, frozen
  def parse_observation(o)      -> State
  def init_belief(o)            -> State
  def predict_belief(s, a)      -> State
  def readout_observation(s, a) -> str
  def correct_belief(s, o)      -> State
```

- **Replay.** Execute on logged trajectories and localize where the disagreement occurs.
- **Repair.** Rewrite the faulty branch instead of retraining a black box.
- **Reuse.** Roll forward in any planner, with no model calls at decision time.
- **Inspect.** States are named fields; any belief can simply be printed.

Separate belief update from readout: the program need not reproduce every phrasing.

Lookahead needs only state-level differences; wording lives in a separate readout function.

Repairing an Executable Model with Counterexamples



- **One patch per round, accepted only if held-out validation replay strictly improves.** Induction costs 17–28 LLM calls once; prediction afterwards is plain Python.

A Repair Example: Counterexample and Patch in Maze

Counterexample found in replay

```
obs:  "... position 1, 2. Walls
       above you, below you."
act:  "move right"
true: "... position 1, 3.
       There is a wall above you."
pred: "Success"
```

Accepted patch: two local edits

```
- if "Success" in obs:
+ if obs.strip() == "Success":
      status = "success"
  if move_blocked:
-     pass
+     walls = belief.walls.copy()
```

- **Errors are concrete.** Failed transitions become typed counterexamples.
- **Edits are local.** The LLM sees the program and the top-16 counterexamples, and edits one branch.
- **Acceptance is global.** The patch must improve replay on the whole held-out validation set.

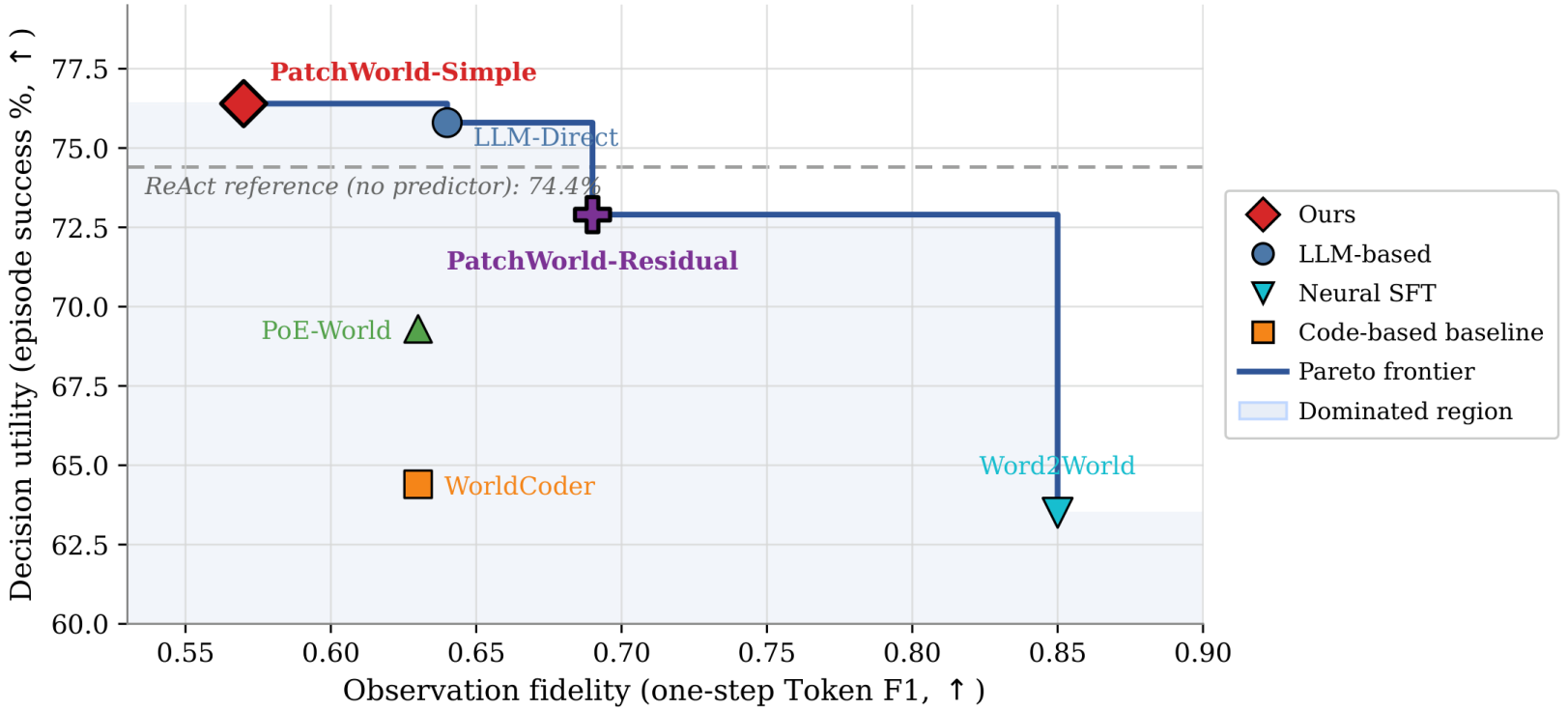
Best Code-Based Decisions at Zero Lookahead Cost

Seven AgentGym environments, one shared one-step lookahead procedure, Qwen3-Coder-480B backbone.

Method	Model class	LLM at inference	Token F1	Success
Word2World	Neural (SFT)	Yes	0.85	63.5%
LLM-Direct	In-context prediction	Yes	0.64	75.7%
ReAct (no world model)	Policy only	Yes	—	74.4%
WorldCoder	Program	No	0.63	64.4%
PoE-World	Program	No	0.63	69.3%
POMDP Coder	Program	No	0.50	74.5%
PatchWorld-Residual	Program + residual memory	No	**0.69**	72.9%
PatchWorld-Simple	**Program**	**No**	0.57	**76.4%**

- **76.4% macro success.** +7.1 over PoE-World and +12.0 over WorldCoder, with an identical candidate pool and selector.
- **0 prediction tokens.** LLM-Direct matches on success but spends about 64k lookahead tokens per task.

Fidelity and Utility Are Different Objectives



- **Residual memory.** F1 rises 0.57 → 0.69, but success falls 76.4% → 72.9%.
- **Word2World.** Highest fidelity in the study (0.85 F1); worst decisions (63.5%).
- **Ablations.** Removing the repair loop costs 7.9 points of success, contrastive mining 12.9, structured belief parsing 24.5.

Lookahead needs action-relevant contrast, not closer surface wording.

Figure 3, PatchWorld. Held-out test split, macro-average over seven environments.

Lessons from PatchWorld

- **Executable beliefs can be learned from text alone.**

  No privileged simulator state is needed; induced programs are inspectable and locally patchable.

- **An LLM is more useful as an optimizer than as a predictor.**

  Moving it out of the inference path improved decisions and removed decision-time model calls.

- **Fidelity is not utility.**

  Report both axes; they form a frontier, not a single optimum.

- **Still open.**

  Beliefs are deterministic; readout still causes ~53% of remaining errors; programs are induced per environment.

PatchWorld searches free-form code. In continuous control that is not enough — a program can parse and replay, and still get the physics wrong.

Part II

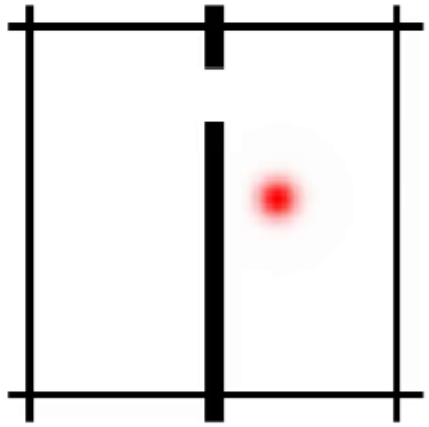
Inducing Dynamics Programs from Visual Environments

VisualPatchWorld: Code World Models as Latent Structured Representations for Planning

Jiaxin Bai, Jiaxuan Xiong · Hong Kong Baptist University

arXiv:2607.25236 https://github.com/HKBU-KnowComp/VisualPatchWorld

Four Control Environments (LeWM Benchmark)



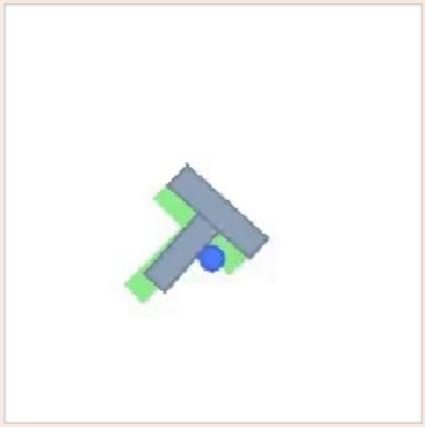
Two-Room

walls and a door



Reacher

joint-driven arm



Push-T

hardest: contact



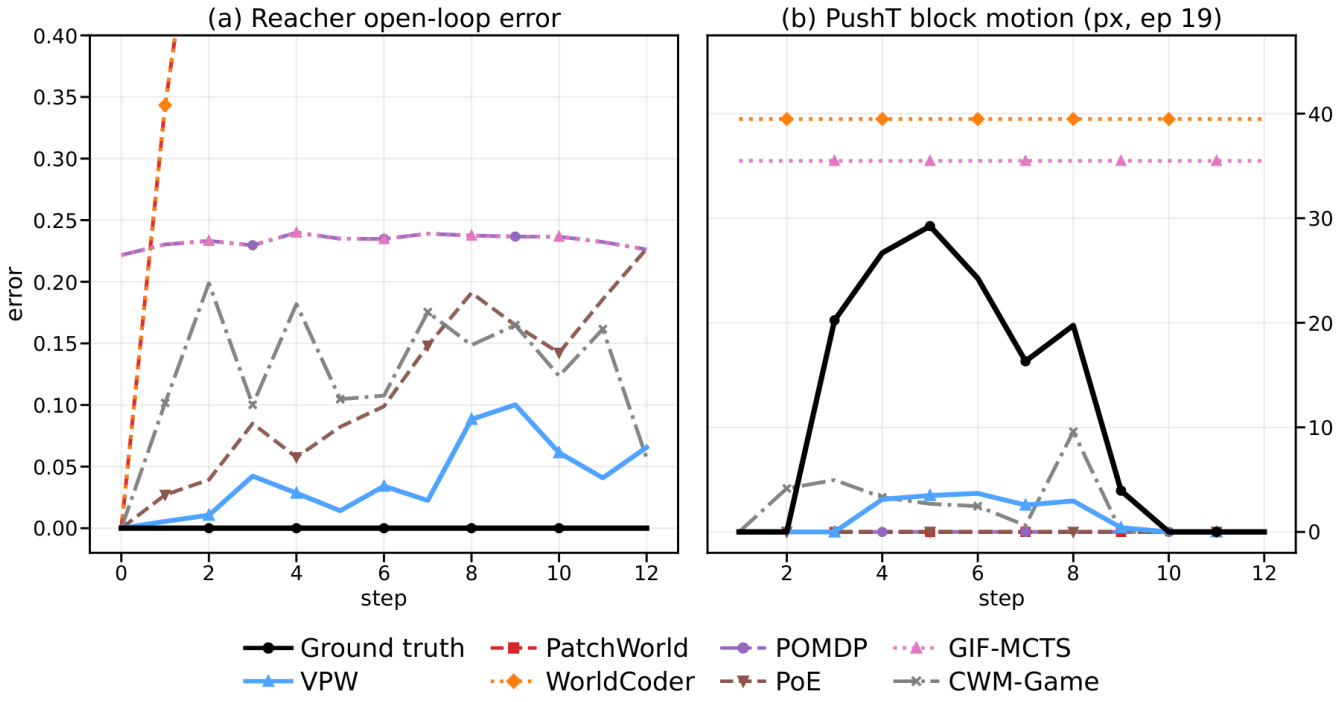
OGB-Cube

moves only once grasped

Same four tasks, same planner: code and latent world models can be **compared directly**.

Environments and demonstrations from LeWM (Maes et al., 2026).

Parses, Fits One Step, Still Wrong Physics





Reacher

fitted as a free particle ✗

actually joint-driven ✓

success 8% → 72%



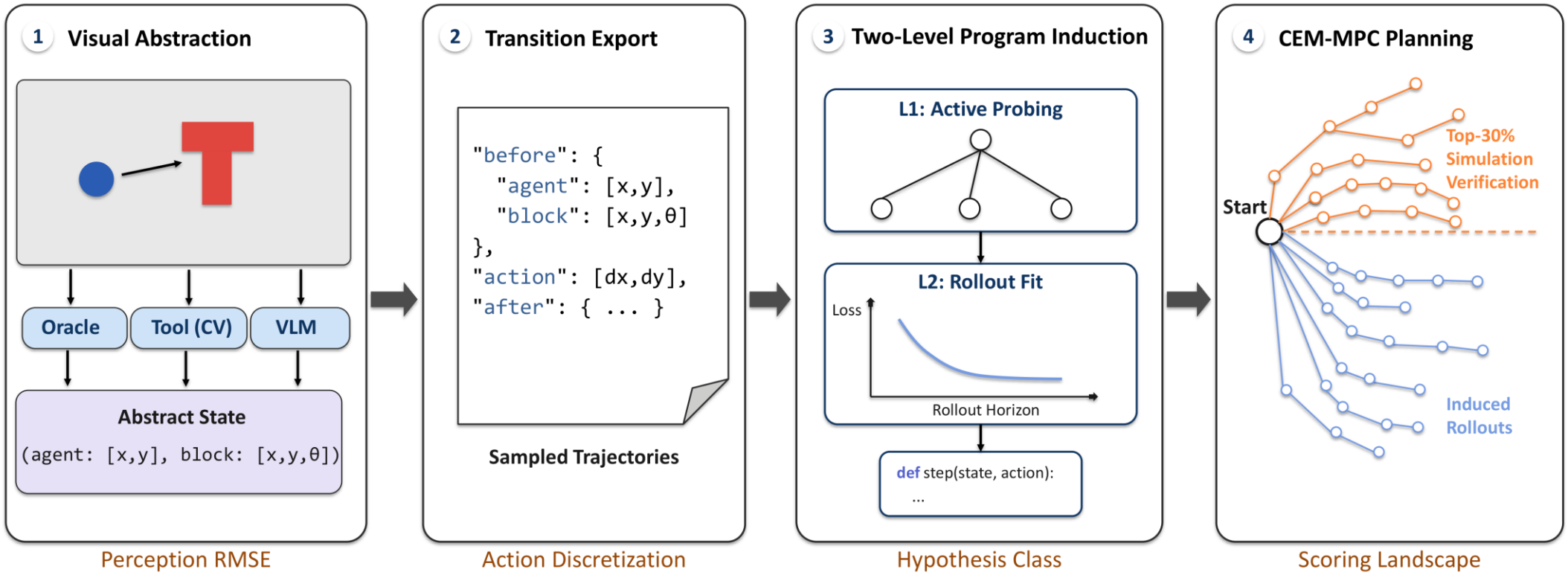
Push-T

block freezes or jumps ✗

contact form is wrong

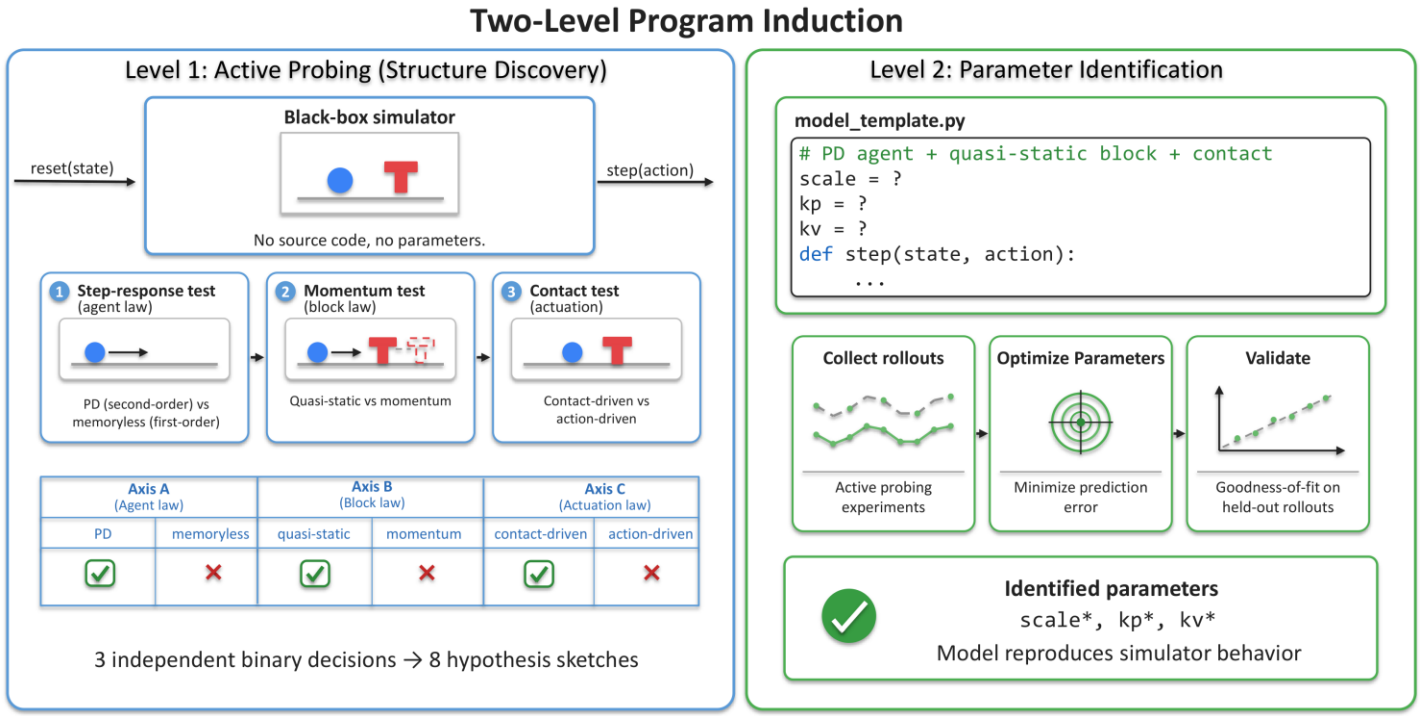
Low one-step error ≠ right physics: the form of the dynamics is what matters.

The VisualPatchWorld Pipeline



Visual abstraction · trajectory export · structure induction · planning with the world model

Two-Level Induction: Structure First, Parameters Second



- **Level 1 selects the form.** A few active probing experiments choose the best formula from a small library of dynamical sketches.
- **Level 2 fits the constants.** Multi-restart minimization of accumulated error over the planning horizon, not one-step error.

What the Induced Programs Look Like

Reacher: baselines 8% → VPW 72%

```
# baseline: free-particle fingertip
finger += (dx, dy)
# VPW: joint update + kinematics
q = q + B @ [1, dx, dy]
finger = FK(q)
```

Push-T: PatchWorld 0% → VPW 22%

```
# baseline: block never moves
agent += (dx, dy)
# VPW: PD pusher, quasi-static block
a = PD(agent, SCALE*act; KP, KV)
block = push(a, contact_springs)
```

Env	Recovered dynamics	Plan %
Two-Room	Linear navigation	96
Reacher	Joints + forward kinematics	72
Push-T	PD pusher + contact push	22
OGB-Cube	Grip-gated contact	86

Readable and editable

KP ≈ 92, KV ≈ 26, SCALE ≈ 98 are physical constants; an error traces to one line.

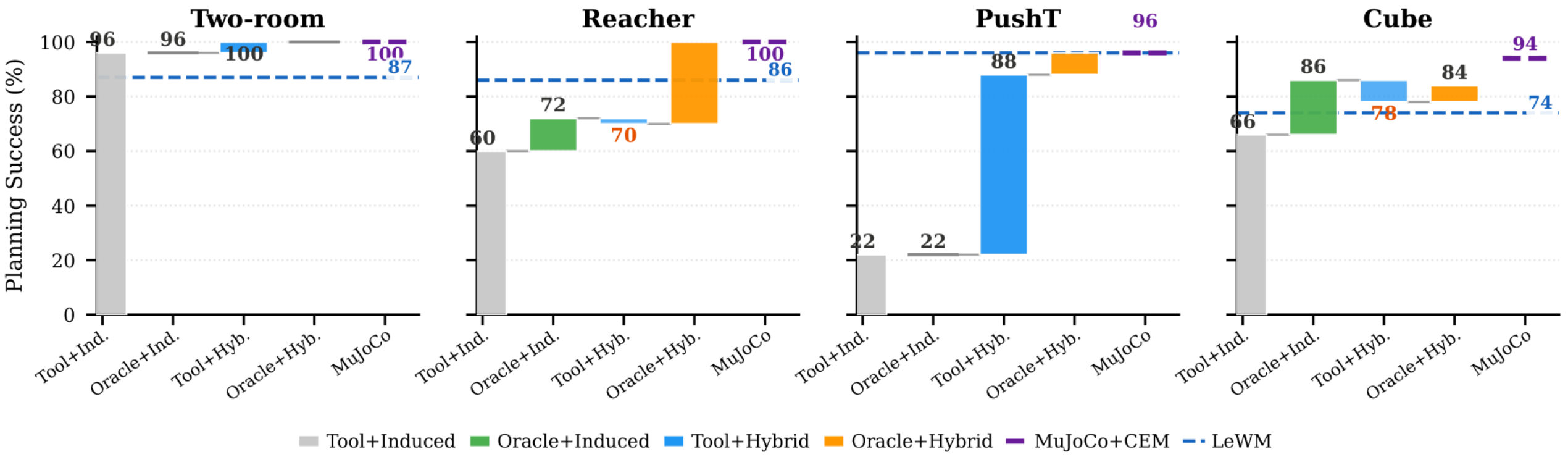
VisualPatchWorld, Tables 5–6 and Appendix D.1.

Same Planner, Right Structure: +23.5 Points

Method	Two-room	Reacher	PushT	Cube	Mean
LeWM (neural, RGB reference)	87	86	96	74	85.8
DINO-WM (neural, RGB reference)	100	79	74	86	84.8
PatchWorld	98	8	0	66	43.0
WorldCoder	60	6	0	74	35.0
POMDP-Coder	98	18	0	66	45.5
PoE-World	6	18	0	84	27.0
GIF-MCTS	6	20	0	66	23.0
VisualPatchWorld	**96**	**72**	**22**	**86**	**69.0**

- **Margin.** +23.5 over POMDP-Coder, the strongest programmatic baseline, and +26.0 over PatchWorld.
- **Where the gains come from.**
  - Reacher 8 → 72 (joint-space dynamics),
  - Cube 66 → 86 (grip-gated contact),
  - PushT 0 → 22 (recovered contact law).

Contact Is the Remaining Bottleneck



- **Where the induced structure fits, it is near the ceiling.** 0.96 of physics-engine success on Two-room and 0.91 on Cube — but only 0.23 on PushT.
- **Hybrid scoring closes the gap.** The induced program still searches the full candidate pool; the engine re-scores only the top 30%: PushT 22 → 96, suite mean 95.0 (ceiling 97.5), with ~70% fewer physics queries.

Lessons from VisualPatchWorld

① **Form is important**

the right structure closes the gap

② **Optimize for the planner**

multi-step, not one-step error

③ **Perception is separable**

image state ≈ oracle state

Still open: richer sketch library · end-to-end from vision · real robots

Related: discovering physical laws from experience, NewtonBench [1]

[1] T. Zheng et al. NewtonBench: Benchmarking Generalizable Scientific Law Discovery in LLM Agents. ICLR 2026.

Code vs. Neural Latent World Models: A Gap Remains

Method	Type	Two-Room	Reacher	Push-T	Cube	Mean
LeWM	Neural (pixels)	87	86	96	74	85.8
DINO-WM	Neural (pixels)	100	79	74	86	84.8
VPW, induced only	**Code program**	**96**	**72**	**22**	**86**	**69.0**
VPW + 30% physics check	Code + engine	100	100	96	84	95.0

Neural latent models still plan better, especially under contact.

Learned end to end from pixels, no hand-written forms: Push-T 96 (LeWM) vs. 22 (VPW, induced only).

Part III: neural latent world models, their background, and the gaps that remain for planning.

Part III

Mining the Planning Cost in Latent Space

Temporal-Distance-JEPA: Plan-Aware Representation Learning for Latent World Model Predictive Control

Jiaxin Bai, Jiaxuan Xiong · Hong Kong Baptist University

arXiv:2607.25337 https://github.com/HKBU-KnowComp/Temporal-Distance-JEPA

Background: JEPA World Models and LeWM



Training: $\mathcal{L}_{\text{pred}} = \|\hat{z}_{t+1} - \text{sg}(\text{Enc}_\theta(o_{t+1}))\|^2 + \lambda \text{SIGReg}(z)$

Predict in representation space; SIGReg keeps latents near an isotropic Gaussian to prevent collapse.

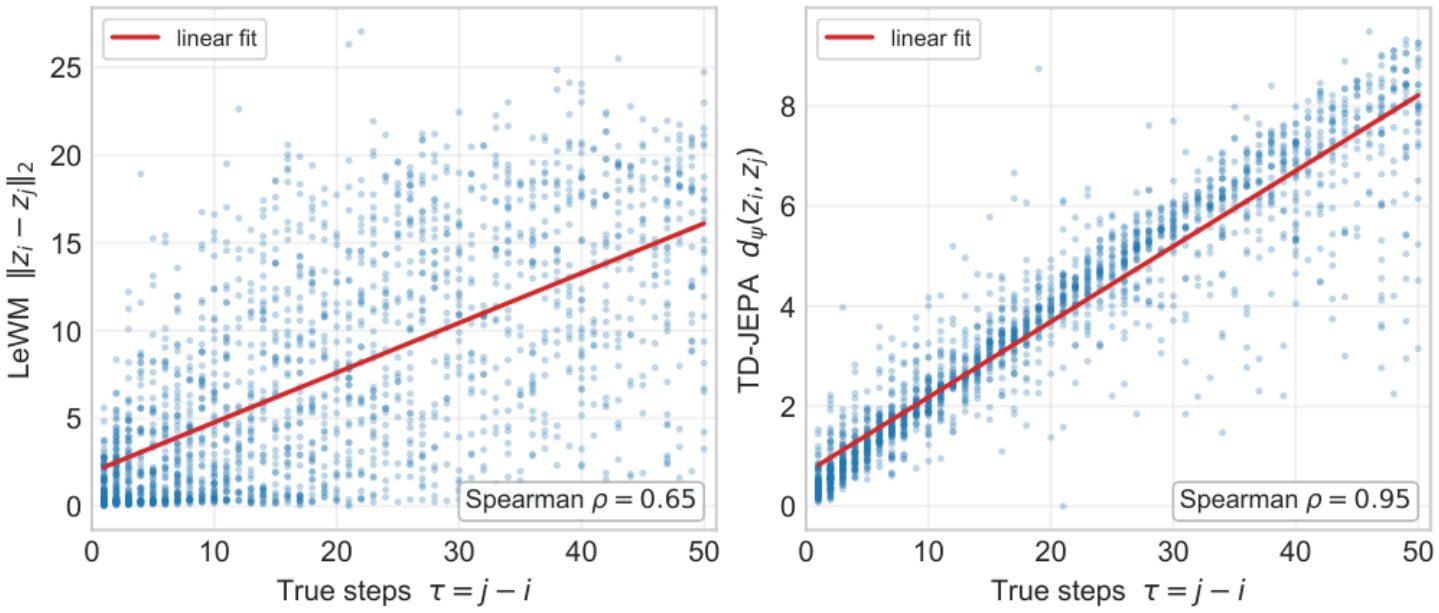
Planning

Sample K plans → roll out H steps with Pred_ϕ → score $\|\hat{z}_H - z_g\|_2$ → act on the best, replan

The planning cost itself is not learned: Euclidean distance in latent space is taken as given.

LeJEPA / SIGReg: Balestriero & LeCun, 2025. LeWM: Maes et al., 2026.

Problem 1: Trained on Trajectories but Does Not Track Progress



LeWM Euclidean distance

Spearman $\rho = 0.65$

States 40 steps apart can look closer than states 5 steps apart.

TD-JEPA cost d_ψ

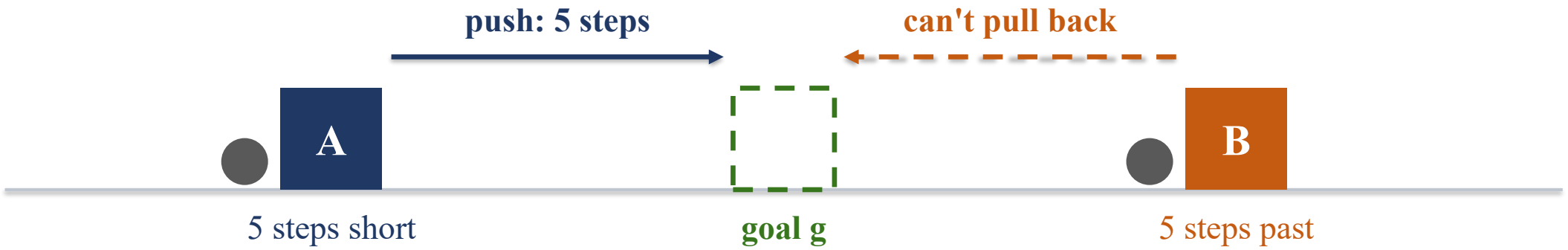
Spearman $\rho = 0.95$

Same backbone, same logs, plus temporal order.

Temporal order in demonstrations is free supervision about which state is closer to the goal.

Problem 2: Euclidean Distance Has No Direction

Push-T, simplified: the pusher (gray) can only push the block to the right.

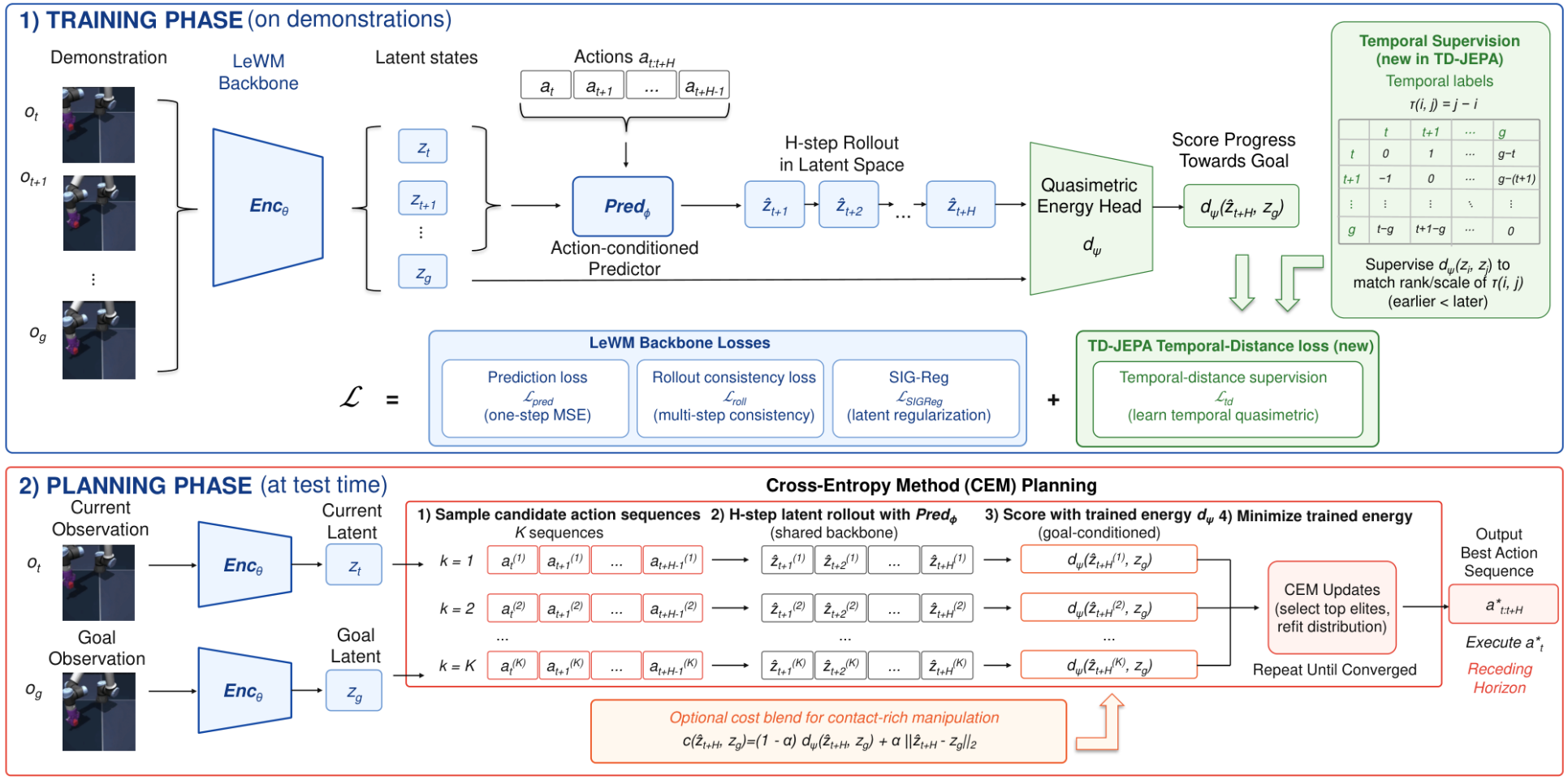


Euclidean: $\|z_A - z_g\| = \|z_B - z_g\|$

Real cost: $d(A \to g) \ll d(B \to g)$

Same temporal labels, direction removed: Push-T success **85% → 57%**.

Temporal-Distance-JEPA: Overview



New: directed cost head d_ψ · temporal supervision from step counts · rollout consistency. Backbone and SIGReg unchanged.

Component 1: A Directed Cost Head

$$d_\psi(z_s \to z_g) = \|\phi_{\text{sym}}(z_s) - \phi_{\text{sym}}(z_g)\|_2 + \max_k \text{ReLU}\big(\phi_{\text{asym},k}(z_s) - \phi_{\text{asym},k}(z_g)\big)$$

symmetric term: shared configuration · directed residual: positive one way only

Why directed?

- Minimum steps $D^*(s, g)$ obeys the triangle inequality but need not be symmetric: a quasimetric.
- Pushing away from a wall is easy; pushing back is hard.
- Euclidean distance cannot express direction.

Relation to LeWM

- LeWM's $\|z - z_g\|_2$ is the special case: identity ϕ_{sym}, zero residual.

Evidence (Push-T)

- Symmetric head instead of directed: 85.3% → 57.3% success.

Component 2: Temporal Supervision from Step Counts

$$\mathcal{L}_{\text{td}} = \mathbb{E}_{i<j}\left[\rho\left(d_\psi(z_i, z_j) - (j-i)\right)\right] + \mathbb{E}_{i,g'\sim\mathcal{N}}\left[\text{ReLU}\left(m - d_\psi(z_i, z_{g'})\right)\right]$$

same-trajectory positives: regress to step gap · cross-trajectory negatives: above margin

Positives: same trajectory

- Step gap $j - i$ is a free, reward-free label.
- Demos need not be shortest: $D^*(o_i, o_j) \leq j - i$.
- So it is **ranking calibration**, not a true distance.

Negatives: other trajectories

- Block spuriously low costs off the demo manifold.
- Margin $m = \eta(T-1)$ for window length T.
- Heuristic: a few false negatives; no value function.

Component 3: Rollout Consistency

$$\mathcal{L}_{\text{roll}} = \frac{1}{H}\sum_{h=1}^{H} \| \hat{z}_{t+h} - \text{sg}(z_{t+h}) \|^2, H = 5 = \text{planning horizon}$$

$$\mathcal{L} = \mathcal{L}_{\text{pred}} + 0.5\mathcal{L}_{\text{roll}} + 1.0\mathcal{L}_{\text{td}} + 0.09\text{SIGReg}(z)$$

Same principle as VisualPatchWorld's multi-step fit: **optimize the way the planner uses the model.**

Main Results: Matching or Beating LeWM and RC-aux

Method	Two-Room	Reacher	Push-T	OGB-Cube
LeWM	97.4 ± 1.3	96.0 ± 1.9	83.6 ± 3.2	68.0 ± 2.8
RC-aux (concurrent)	98.6 ± 1.0	96.8 ± 1.4	81.4 ± 1.9	81.6 ± 2.8
TD-JEPA (ours)	**100.0 ± 0.0**	**97.0 ± 2.4**	**86.0 ± 4.2**	**82.2 ± 2.9**
Plan cost used by TD-JEPA	d_ψ	d_ψ	ℓ_2	ℓ_2

- **Where d_ψ is the plan cost:** Two-Room 100%, Reacher 97%; an explicit temporal cost beats a reachability auxiliary that still plans with geometry.
- **Where all plan with ℓ_2:** the gain comes from temporal supervision shaping the representation (OGB-Cube +14.2 over LeWM).

Ablations: Co-Design Cost Form and Supervision

Variant	Blend	Pure d_ψ	Pure ℓ_2	iCEM
TD-JEPA (full)	**82.7 ± 6.4**	**69.3 ± 3.1**	**85.3 ± 4.2**	**65.3 ± 6.1**
Symmetric Euclidean head	56.7 ± 4.2	55.3 ± 5.0	57.3 ± 4.2	43.3 ± 1.2
w/o cross-traj. negatives	72.0 ± 3.5	68.7 ± 3.1	77.3 ± 4.2	49.3 ± 1.2
w/o rollout consistency	62.0 ± 3.5	56.7 ± 3.1	60.0 ± 3.5	44.0 ± 2.0

- **Every component matters** under all four planner settings.
- **Form matters more than labels:** a symmetric head drops ℓ_2 planning 85.3% → 57.3%.
- **Match the horizon:** no rollout consistency (60.0%) hurts more than no negatives (77.3%).

Lessons from Temporal-Distance-JEPA

Mine the planning cost from logs; don't inherit it from embedding geometry.

Same backbone and data plus temporal order: ρ 0.65 → 0.95; Two-Room reaches 100%.

Co-design the cost form and its supervision.

Directed head, negatives and rollout consistency all matter.

Better ranking ≠ better planning.

Push-T: d_ψ ranks better (0.91 vs. 0.79) yet plans worse (69% vs. 86%).

Still open.

Post-contact control needs local geometry; only four simulated environments so far.

Two World Models on the Same Environments

	VisualPatchWorld (code)	TD-JEPA (latent)
Dynamics	Python program	Neural network
Learned from data	Physical law	Planning cost
Readable	Yes	No
Hardest case	**Push-T contact**	**Push-T contact**

Opposite routes, same hard case.

The Shared Bottleneck: Contact

Code world model (VPW)

22% → 60% (96%)

Push-T success

Problem: errors at contact

Fix: the physics engine re-checks the top plans

Latent world model (TD-JEPA)

69% → 86%

Push-T success

Problem: d_ψ plans fail after contact

Fix: plan with ℓ_2 (local geometry)

Global structure gets the agent to the block; after contact, plans need a local check.

Summary

	PatchWorld	VisualPatchWorld	TD-JEPA
Question	Belief from text alone?	Right form of dynamics?	What ranks latent futures?
World model	Python belief program	Python dynamics program	JEPA latent + directed cost
Mined from logs	Counterexamples	Form + parameters	Temporal order
Planner	One-step lookahead	CEM-MPC	CEM / iCEM-MPC
Result	76.4%, 0 lookahead tokens	69.0%, +23.5 over code	≥ LeWM, RC-aux on all four
Lesson	Fidelity ≠ utility	Wrong form can't be fit away	Better ranking ≠ better plans

Learn world models for what the planner needs; mine that structure from logs, and evaluate by decisions, not reconstruction.

Outlook: Dynamics × Cost, Explicit × Latent

	Dynamics: what happens?	**Planning cost**: which is better?
Explicit / code	**PatchWorld · VisualPatchWorld** Inspectable transition laws	**Open question** Induce a programmatic notion of progress
Latent	**LeWM · DINO-WM · Dreamer** Learned end-to-end from pixels	**Temporal-Distance-JEPA** Directed cost from temporal order

- **Contact:** executable contact programs inside latent models, or code filters plus local physics checks.
- **Perception and real robots:** induce programs end-to-end from vision; evaluate beyond simulation.

Thank You

Thank you.
Questions welcome.

Jiaxin Bai · Hong Kong Baptist University · baijiaxin@hkbu.edu.hk · https://bjx.fun

arXiv: PatchWorld 2605.30880 · VisualPatchWorld 2607.25236 · TD-JEPA 2607.25337

https://github.com/HKBU-KnowComp/PatchWorld

https://github.com/HKBU-KnowComp/VisualPatchWorld

https://github.com/HKBU-KnowComp/Temporal-Distance-JEPA