DISENTANGLING THE GEOMETRIES OF PREDICTION AND PLANNING IN LATENT WORLD MODELS THROUGH TRAJECTORY-GRAPH CONTROL CALIBRATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Joint-Embedding Predictive Architectures (JEPAs) provide an efficient foundation for latent world models by predicting future representations rather than reconstructing pixels. However, Model Predictive Control (MPC) planners built on these models typically select action sequences by minimizing the terminal latent distance to the goal, which implicitly assumes that latent similarity measures the remaining cost to the goal. In contrast, goal-reaching cost can be directional, violating the symmetry imposed by latent-distance planning and potentially causing the planner to prematurely discard dynamically preferable action sequences. To address this limitation, we propose Trajectory-Graph Control Calibration (TGCC), a lightweight post-hoc method that disentangles the geometries of prediction and planning over the same frozen latent representations. Concretely, TGCC constructs a directed graph over offline trajectories and distills its temporal relations into a control geometry network. During planning, the network's graph-derived control score is combined with terminal latent similarity to rank candidate action sequences, thereby defining a planning geometry beyond the prediction geometry. Without retraining the world model, TGCC achieves superior planning performance across multiple datasets and latent world-model backbones, with higher success rates and fewer steps to success.

1 INTRODUCTION

Joint-Embedding Predictive Architectures (JEPAs) provide a general representation-learning framework that predicts target embeddings from contextual observations rather than reconstructing pixels (Assran et al., 2023; Bardes et al., 2024). Built upon this framework, most latent world models incorporate actions into representation prediction, enabling efficient and generalizable planning from high-dimensional observations (Zhou et al., 2025; Maes et al., 2026; Sobal et al., 2025; Hansen et al., 2024). Latent world models therefore have broad potential to advance long-horizon decision-making, model-based control, and embodied intelligence (Hafner et al., 2025; Hansen et al., 2024; Wu et al., 2024; Gao et al., 2025).

However, latent world models are trained to predict future representations rather than generate action sequences that reach a specified goal. Model Predictive Control (MPC) is therefore commonly paired with these models to search over candidate action sequences. During planning, a typical Cross-Entropy Method (CEM) planner (Rubinstein & Kroese, 2004) samples candidate sequences, rolls them out through the latent world model, and uses terminal latent similarity to the goal as the scalar objective for selecting elites and guiding subsequent sampling. Research on latent world-model planning has largely focused on whether learned representations capture sufficient state information and structure to support planning, and many representative latent planners directly use native latent similarity as the objective for ranking candidate action sequences. However, representational sufficiency does not imply that the geometry learned for prediction is suitable for planning. For example, the control cost of transitioning from one state to another can differ from the cost of returning to the original state, whereas commonly used latent similarity measures are symmetric and cannot express this directional difference. The **LeWM** panel in Figure 1 visualizes LeWM's remaining-cost field on the Two-Room benchmark (Maes et al., 2026). The field is obtained by encoding the observation at each

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

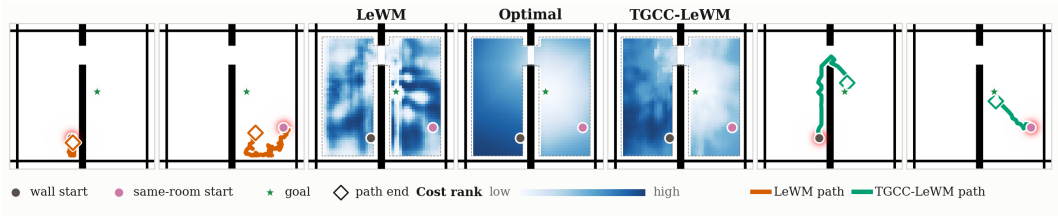

Figure 1: Visualization of the mismatch between prediction and planning geometries. The two trajectories planned by LeWM on the left both fail, as its latent-similarity cost field in the LeWM panel deviates substantially from the ground-truth remaining-cost field shown in the Optimal panel. TGCC calibrates the planning geometry to better align with the ground-truth structure in the TGCC-LeWM panel, enabling successful planning from both start states, as shown by the two trajectories on the right.

physical position and computing its latent distance to a fixed goal. The resulting spatially irregular field exposes a structural mismatch between native latent similarity and the remaining control cost.

To address this mismatch, we introduce **Trajectory-Graph Control Calibration** (TGCC), a lightweight post-hoc method that disentangles the geometries of prediction and planning over the same frozen latent representations. TGCC preserves the predictive geometry learned by the world model and complements native latent similarity with directed control structure distilled from an offline trajectory graph, thereby defining a planning geometry, as visualized in the **TGCC-LeWM** panel of Figure 1. Specifically, offline trajectories are first organized into a directed graph, on which bounded breadth-first search is used to derive forward-reachability relations before deployment. These precomputed relations are then distilled into a network over ordered pairs of latent states, transferring the directed geometry of the offline graph into a continuous estimator of control reachability. During planning, the estimator evaluates control reachability from each predicted rollout state to the task goal, and its output is incorporated into the MPC candidate-ranking objective, where it is combined with terminal latent similarity after candidate-wise normalization. The resulting planning geometry combines native predictive similarity with the trajectory-graph-derived control cost to rank candidate action sequences, thereby improving goal-directed planning without altering the world model's predictive behavior.

Our contributions can be summarized as:

- We identify a common but insufficiently examined assumption in latent world-model planning, namely that terminal latent similarity serves as a surrogate for remaining goal-reaching cost despite a potential mismatch between prediction and planning geometries.
- We propose TGCC, a lightweight and plug-and-play calibration framework applicable to different latent world-model backbones. With the encoder and predictor unchanged, TGCC improves success rate and trajectory efficiency with minor planning overhead.
- To our knowledge, TGCC is the first framework to explicitly disentangle prediction and planning geometries in a predictive latent world model using a directed trajectory graph that encodes control reachability.

2 Related Work

Joint-embedding world models for visuomotor control. Joint-embedding predictive architectures learn by predicting the target representations rather than reconstructing observations. I-JEPA established this principle for images, V-JEPA extended feature prediction to video, and V-JEPA 2 connected self-supervised video representations to prediction and planning (Assran et al., 2023; Bardes et al., 2024; Assran et al., 2025). Action-conditioned world models bring related predictive representations to visuomotor control: DINO-WM predicts dynamics in pretrained visual features, LeWM learns an end-to-end latent world model from pixels, and PLDM studies planning with reward-free offline latent dynamics (Zhou et al., 2025; Maes et al., 2026; Sobal et al., 2025). Building

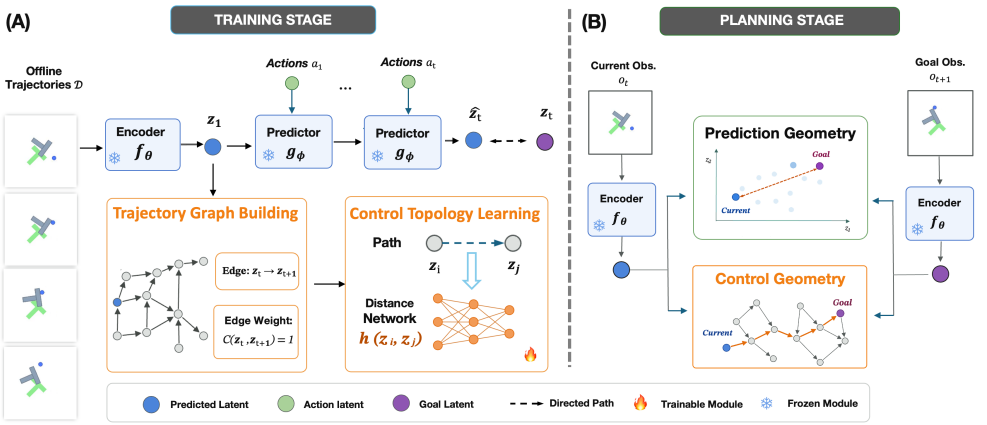

Figure 2: **Pipeline of our TGCC.** (A) In the training stage, the frozen encoder maps recorded observations from offline trajectories into latent states. Their directed temporal transitions are organized into a trajectory graph, whose reachability structure is distilled into a lightweight control-geometry network. (B) During planning, candidate latent rollouts are evaluated relative to the goal under both the native prediction geometry and the learned control geometry. The graph-derived control cost is calibrated with the native prediction cost to define a dedicated planning geometry for candidate ranking.

on these predictive models, TGCC disentangles prediction and planning geometries while keeping the encoder and predictor frozen.

Sampling-based latent planning and candidate objectives. Learned dynamics support planning over action sequences in methods such as PETS and PlaNet (Chua et al., 2018; Hafner et al., 2019). In sampling-based MPC, CEM samples candidate action sequences, the world model predicts their rollouts, and the candidate objective selects elites that update the next sampling distribution (Rubinstein & Kroese, 2004; Gieselmann & Pokorny, 2022). The objective therefore affects both the action sequence selected for execution and the regions of action space explored in subsequent iterations. Dreamer and TD-MPC2 instead represent broader approaches to behavior and value learning with compact latent models (Hafner et al., 2020; Hansen et al., 2024). TGCC focuses on the MPC candidate-ranking objective, combining native latent similarity with graph-derived control reachability to evaluate predicted action sequences.

Graph-based reachability and topological planning. A complementary line of work represents long-horizon control structure with graphs. SPTM and ViNG construct topological graphs over visual observations or previously collected experience for waypoint-based navigation (Savinov et al., 2018; Shah et al., 2021). SoRB searches over replay-buffer states using a learned goal-conditioned distance, L3P constructs a graph-structured world model from latent landmarks and reachability-weighted edges, and Bagatella et al. learn latent graphs from pixels for combinatorial planning (Eysenbach et al., 2019; Zhang et al., 2021; Bagatella et al., 2021; Beker et al., 2022). Mezghani et al. (2023) derive asymmetric goal-reaching signals from the temporal ordering and shortest-path structure of offline trajectories for goal-conditioned policy learning. TGCC instead distills graph relations into a continuous scorer for arbitrary model-predicted latent states and uses this scorer to calibrate MPC while keeping the predictive world model frozen.

3 Methodology

To disentangle the geometry used for future prediction from that required for goal-directed planning, we propose **Trajectory-Graph Control Calibration** (TGCC), which preserves the native predictive geometry of a latent world model while introducing a trajectory-informed control geometry over the same latent space. As depicted in Figure 2, TGCC consists of an offline training stage and an online planning stage. The training stage constructs a directed trajectory graph and distills its temporal topology into a continuous control-geometry network, as detailed in the *Directed Trajectory Graph* and *Graph-Distilled Control Geometry* subsections. During planning, the predicted control cost is

combined with terminal latent similarity to rank MPC candidates, as described in the *Dual-Geometry Planning* subsection. Moreover, TGCC can be trained jointly with the world model while introducing negligible additional training overhead.

3.1 Latent World Model and Prediction Geometry

We consider an offline trajectory dataset

$$\mathcal{D} = \{\tau^n\}_{n=1}^{N_{\text{ep}}}, \qquad \tau^n = (o_0^n, a_0^n, o_1^n, \ldots, a_{T_n-1}^n, o_{T_n}^n), \tag{1}$$

where o_t^n denotes an observation and a_t^n the action applied at time t. A latent world model consists of an encoder f_θ and an action-conditioned predictor g_ϕ:

$$z_t^n = f_\theta(o_t^n), \qquad \hat{z}_{t+1} = g_\phi(\hat{z}_t, a_t). \tag{2}$$

TGCC is compatible with either joint training or post-hoc attachment. In our controlled evaluation, f_θ and g_ϕ are frozen to isolate the contribution of the newly introduced graph-derived control metric from changes in representation learning or dynamics prediction.

Given a current observation o_0 and a goal observation o_g, their representations are

$$z_0 = f_\theta(o_0), \qquad z_g = f_\theta(o_g). \tag{3}$$

A sampling-based MPC planner proposes a population of candidate action sequences

$$\mathcal{A} = \{A_i\}_{i=1}^{M}, \qquad A_i = (a_0^i, \ldots, a_{H-1}^i), \tag{4}$$

and rolls each candidate forward through the predictor:

$$\hat{z}_0^i = z_0, \qquad \hat{z}_{k+1}^i = g_\phi(\hat{z}_k^i, a_k^i), \quad k = 0, \ldots, H-1. \tag{5}$$

The native planner evaluates candidate A_i through terminal proximity in the learned representation space:

$$c_{\text{pred}}^i = \left\| \hat{z}_H^i - z_g \right\|_2^2. \tag{6}$$

We refer to the relational structure induced by this distance as the *prediction geometry*. It preserves the appearance- and prediction-related structure learned by the world model.

3.2 Directed Trajectory Graph

TGCC constructs a directed control structure from the temporal transitions observed in the offline trajectories. Each encoded observation is associated with a graph node

$$v_{n,t} \in V, \qquad z_{v_{n,t}} = f_\theta(o_t^n), \tag{7}$$

and every observed transition contributes a directed edge

$$e_{n,t} = (v_{n,t}, v_{n,t+1}) \in E. \tag{8}$$

To avoid restricting the graph to disjoint episode-level chains, we additionally connect matched states from different episodes. We assign each one-step transition unit cost,

$$w(v_{n,t}, v_{n,t+1}) = 1, \tag{9}$$

forming a directed trajectory graph

$$\mathcal{G} = (V, E, w). \tag{10}$$

For two graph nodes v_i and v_j, we define their directed transition distance as

$$d_{\mathcal{G}}(v_i, v_j) = \min_{\pi : v_i \rightsquigarrow v_j} \sum_{e \in \pi} w(e), \tag{11}$$

and set $d_{\mathcal{G}}(v_i, v_j) = \infty$ when no observed directed path exists. The graph topology determines whether two states are connected, whereas the directed path distance equips this topology with a discrete control geometry.

For a set of transition budgets $\mathcal{B}$, we define graph-derived reachability labels

$$y_{ijb} = \mathbf{1}\left[d_{\mathcal{G}}(v_i, v_j) \le b\right], \qquad b \in \mathcal{B}. \tag{12}$$

A positive label indicates that the offline graph contains a directed path from v_i to v_j within budget b. An absent path indicates only a lack of support in the observed trajectories and is not interpreted as proof of environmental unreachability. In practice, we use breadth-first search to compute the directed graph distance $d_{\mathcal{G}}(v_i, v_j)$. For each sampled source node v_i, the search is truncated at the maximum budget $B_{\max} = \max \mathcal{B}$. If v_j is first reached at depth k, then $d_{\mathcal{G}}(v_i, v_j) = k$; nodes not reached within $B_{\max}$ are treated as unreachable under the considered budgets.

3.3 GRAPH-DISTILLED CONTROL GEOMETRY

The directed graph defines control relations only between observed trajectory nodes, whereas MPC must score arbitrary model-predicted states. TGCC therefore distills the graph-derived reachability relations into a continuous ordered-pair control-cost estimator:

$$h_\eta(z_i, z_j) \in \mathbb{R}_{\geq 0}, \tag{13}$$

where a smaller value indicates stronger directed connectivity from z_i to z_j.

Before pairwise scoring, layer normalization is applied independently to each latent representation:

$$\bar{z}_i = \mathrm{LN}(z_i), \qquad \bar{z}_j = \mathrm{LN}(z_j), \tag{14}$$

where $\mathrm{LN}(\cdot)$ denotes layer normalization over the latent feature dimension. The directed relation between the two states is then parameterized as

$$h_\eta(z_i, z_j) = \mathrm{softplus}\left(q_\eta(\bar{z}_i, \bar{z}_j)\right), \tag{15}$$

where q_η denotes a lightweight pairwise network. By assigning distinct source and target roles to the two inputs, this formulation can represent directed transition relations that cannot be captured by symmetric latent distance.

For transition budget b, the score induces the reachability probability

$$P_\eta(i, j, b) = \sigma\left(\frac{b - h_\eta(z_i, z_j)}{\tau}\right). \tag{16}$$

Based on the graph-derived budget relations in Eq. (12), the graph-budget loss is defined as

$$\begin{aligned}\mathcal{L}_{\mathrm{budget}} = \mathbb{E}_{i,j,b}\big[w_{ij}\big(&- y_{ijb} \log P_\eta(i, j, b) \\ &- (1 - y_{ijb}) \log(1 - P_\eta(i, j, b))\big)\big],\end{aligned} \tag{17}$$

where w_{ij} reflects the confidence associated with each ordered state pair. Observed forward pairs are assigned full weight, whereas reverse or cross-trajectory pairs receive a smaller weight because the absence of an observed path does not necessarily imply environmental unreachability. Implementation details are provided in Appendix B. Minimizing this objective distills the directed temporal relations encoded by the trajectory graph into the continuous scorer h_η.

3.4 DUAL-GEOMETRY PLANNING

During planning, TGCC disentangles the planning geometry from the native prediction geometry by introducing trajectory-graph control calibration. Terminal latent proximity preserves the predictive structure of the frozen world model, whereas the learned control cost characterizes directed progress toward the goal. The two signals jointly define a dedicated planning geometry for MPC candidate ranking.

The prediction-geometry score remains the native terminal cost in Eq. (6). To evaluate the learned control geometry, TGCC applies h_η to selected states along the predicted rollout. Let $\mathcal{K} \subseteq \{1, \ldots, H\}$ denote the set of evaluated prediction steps within planning horizon H. The specific choices of H and $\mathcal{K}$ are given in Appendix C. With soft-min temperature $\beta > 0$, the rollout-level control cost is

$$c^i_{\mathrm{ctrl}} = -\beta \log \sum_{k \in \mathcal{K}} \exp\left(-\frac{h_\eta(\hat{z}^i_k, z_g)}{\beta}\right). \tag{18}$$

This soft-min readout assigns a low control cost when a candidate rollout enters a predicted state with low remaining directed transition cost to the goal.

To place the prediction and control costs on a comparable scale, each score is standardized within the current candidate population:

$$\mathcal{N}_{\mathcal{A}}(c^i) = \frac{c^i - \mu_{\mathcal{A}}(c)}{\sigma_{\mathcal{A}}(c) + \epsilon}. \tag{19}$$

The final calibrated planning objective is

$$\begin{aligned}C_{\mathrm{TGCC}}(A_i) = (1 - \alpha)&\mathcal{N}_{\mathcal{A}}(c^i_{\mathrm{pred}}) \\ &+ \alpha \mathcal{N}_{\mathcal{A}}(c^i_{\mathrm{ctrl}})\end{aligned} \tag{20}$$

Equation (20) defines the *planning geometry*. The prediction term preserves the native visual and local transition sensitivity of the world model, whereas the graph-distilled term introduces directed reachability and trajectory-level control structure.

CEM uses $C_{\mathrm{TGCC}}(A_i)$ to select elites and update its action distribution. After the final CEM iteration, the first action or action block of the lowest-cost candidate is executed, and planning is repeated from the next observation.

4 EXPERIMENTS

We organize the experiments around five questions concerning control performance, planning geometry, component contributions, trajectory efficiency, and computational overhead:

1. Does disentangling planning geometry from prediction geometry improve closed-loop control?
2. Does the learned control geometry recover transition topology that is absent from native latent distance?
3. Are prediction and control geometries complementary, and does graph directionality improve closed-loop performance?
4. Does TGCC improve path efficiency and closed-loop behavior?
5. What computational overhead does TGCC introduce during offline TGCC fitting and online planning?

4.1 EXPERIMENTAL SETUP

Tasks and backbones. TGCC is evaluated on four goal-conditioned vision-based control environments: Two-Room (Sobal et al., 2025), Reacher (Tassa et al., 2018), PushT (Chi et al., 2025), and Cube (Park et al., 2025). LeWM is adopted as the primary backbone owing to its top-tier performance on these benchmarks and its fast and stable end-to-end training from pixels (Maes et al., 2026). PLDM provides a second latent-world-model backbone with a different representation architecture and planning protocol (Sobal et al., 2025), allowing us to examine whether TGCC transfers across predictive models.

Baselines. The comparison covers three families of methods. IRIS and DreamerV3 are model-based RL agents that learn behavior from imagined trajectories (Micheli et al., 2023; Hafner et al., 2025). GCBC, GCIQL, and GCIVL are offline goal-conditioned methods evaluated on OGBench (Park et al., 2025). PLDM, DINO-WM, and LeWM are latent world-model planners that optimize action sequences through predicted latent dynamics (Sobal et al., 2025; Zhou et al., 2025; Maes et al., 2026). TGCC is applied to the frozen PLDM and LeWM backbones without retraining their encoders or predictors.

Metrics. We report success rate and steps-to-success to evaluate goal-reaching performance and path efficiency, respectively. Steps-to-success is computed on episodes solved by both the native planner

Method	**Two-Room** ↑	**Reacher** ↑	**PushT** ↑	**Cube** ↑
IRIS	28	6	2	52
DreamerV3	**100**	64	30	53
GCBC	**100**	78	75	66
GCIQL	**100**	96	20	74
GCIVL	**100**	94	33	76
PLDM	77	41	6	32
DINO-WM	**100**	79	74	**80**
LeWM	87	86	92	68
TGCC-PLDM (Ours)	82	61	6	37
TGCC-LeWM (Ours)	**100**	**98**	**96**	**80**

Table 1: Success rate (%) across four environments.

Variant	Two-Room	Reacher	PushT	Cube
TGCC	**96.8±3.2**	**96.4±1.5**	**91.2±4.5**	**74.0±6.0**
w/o Control Geometry	88.0±4.9	84.4±8.4	86.4±3.9	61.2±8.0
w/o Prediction Geometry	**99.2±1.0**	89.2±2.3	68.0±6.5	57.6±5.7
w/o Graph Directionality	95.6±2.2	93.6±3.0	88.0±3.7	64.0±7.6

Table 2: **Ablation study of TGCC-LeWM.** Results are reported as success rate (mean ± standard deviation over five seeds). *w/o Control Geometry* retains only the native prediction geometry, whereas *w/o Prediction Geometry* uses only the graph-derived control geometry. *w/o Graph Directionality* retains both geometry terms but removes directional information from the graph-derived term.

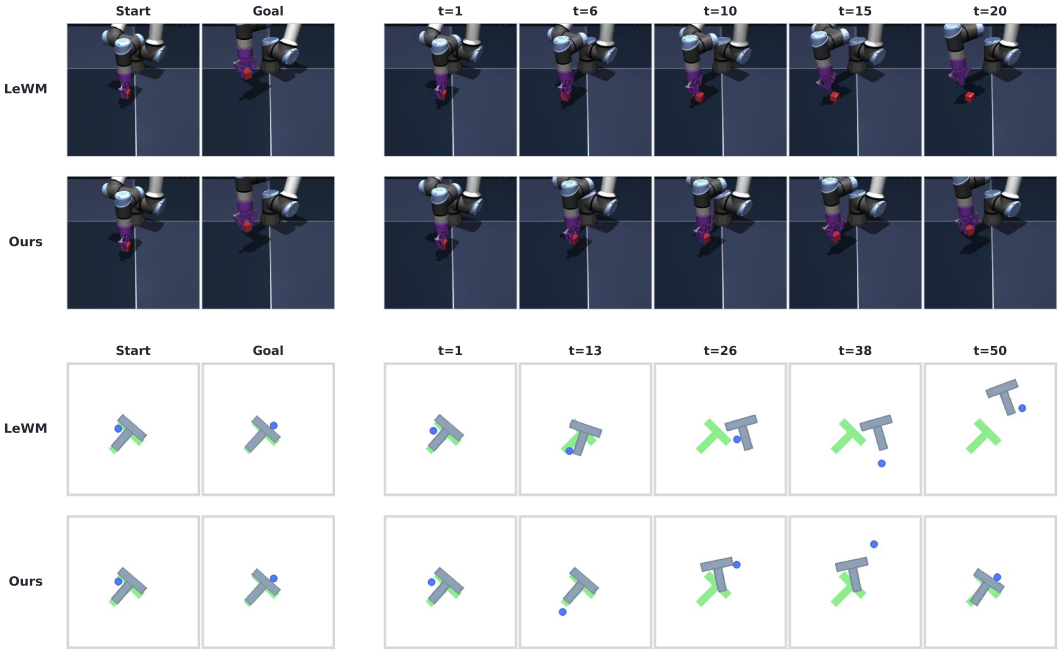

Figure 3: **Qualitative comparison on Cube and PushT.** Under matched start and goal configurations and the same execution budget, TGCC-LeWM successfully completes both Cube and PushT, whereas LeWM fails to reach the target states.

and TGCC. Complete implementation details, hyperparameter settings, graph-construction rules, and evaluation protocols are provided in the supplementary material (Appendices B, C, E, and F).

4.2 Quantitative Results

To answer Q1, we compare TGCC with representative baselines across four visual-control environments. As shown in Table 1, TGCC-LeWM matches or exceeds all compared baselines across the four tasks, achieving the highest results on Reacher and PushT and tied-best performance on Two-Room and Cube. TGCC improves PLDM on three of the four tasks and matches it on PushT. In particular, TGCC raises PLDM success from 77 to 82 on Two-Room and from 41 to 61 on Reacher. These gains are obtained without retraining the predictive model, demonstrating that disentangling planning geometry from prediction geometry improves action-sequence selection and closed-loop control. Figure 3 further illustrates this improvement on Cube and PushT. For further experimental comparisons, please refer to Appendix Table S9.

4.3 Geometry Analysis

To answer Q2, we examine whether the planning cost orders states according to the remaining path required to reach the goal. In Two-Room, this reference can be computed directly from the

Evaluation set	Metric	LeWM	TGCC
Environment states	Spearman ↑	0.009	**0.949**
	Inversion ↓	0.499	**0.096**
CEM rollouts	Spearman ↑	0.442±0.031	**0.603±0.030**
	Inversion ↓	0.347±0.012	**0.285±0.012**

Table 3: **Planning-cost alignment with the ground-truth path cost in Two-Room.** For *Environment states*, we evaluate states sampled throughout the map with $h_\eta(z, z_g)$, whereas for *CEM rollouts*, we evaluate identical model-predicted candidate populations generated during planning with C_{TGCC}. Spearman correlation measures agreement between the cost ranking and the path-length ranking; pairwise inversion rate measures the fraction of sampled pairs ordered differently by the two quantities.

environment layout as the shortest feasible path through the doorway. We compare the native LeWM cost and TGCC-LeWM with this reference, first on states sampled throughout the environment and then on the same predicted rollouts generated during CEM planning. As shown in Table 3, TGCC improves rank agreement and reduces ordering inversions in both settings, indicating that its control geometry better captures the environment's transition structure.

4.4 Ablation Study

To answer Q3, we ablate each component of the TGCC planning geometry while keeping all other settings unchanged. As shown in Table 2, a single geometry can be advantageous for a particular task, but neither variant performs consistently across environments. TGCC instead provides the strongest overall cross-task performance.

Without control geometry, planning relies only on latent proximity and lacks directed information about goal-reaching progress. Control geometry alone performs particularly well in the topology-dominated Two-Room environment, yet degrades substantially on PushT and Cube, where fine-grained appearance, pose, and contact information becomes critical. This contrast demonstrates that the graph-derived control cost supplies directionality and reachability, whereas the native prediction cost retains the local state discrimination required for contact-rich manipulation. Their calibrated combination therefore provides a more general planning objective than either metric alone.

With both geometry terms retained, retaining graph directionality yields higher mean success in all four tasks than removing it. Notably, the directed variant has only a small mean-success advantage on the relatively simple Two-Room task, but a much larger advantage on Cube, indicating that directional reachability information can provide additional benefit beyond the underlying graph-derived control structure.

We additionally examine the effect of cross-episode graph stitching on Two-Room and PushT (Appendix F.3), showing that connecting compatible states across trajectories provides additional control information beyond isolated episode-level transition chains.

Backbone	Task	Goal-offset setting		
		n25	n50	n75
PLDM	Two-Room	10.5→**6.6**	34.7→**31.6**	68.4→**54.6**
	Reacher	22.38→**17.14**	25.0→**22.7**	44.3→**39.0**
	Cube	4.5→**4.1**	3.1→**2.2**	2.2→**2.0**
LeWM	Two-Room	17.7→**13.7**	31.9→**28.9**	46.0→**41.9**
	Reacher	27.2→**20.0**	36.2→**27.0**	37.7→**29.1**
	Cube	5.7→**5.2**	6.9→**5.6**	7.7→**6.5**

Table 4: **Comparison of successful path length.** It's measured as the number of executed environment steps required to reach the goal. Each entry reports the native planner→its TGCC-calibrated counterpart on episodes solved by both methods; lower is better and bold marks the shorter trajectory. The n25, n50, and n75 columns denote three predefined start–goal offset levels of increasing difficulty, with larger values corresponding to larger goal offsets.

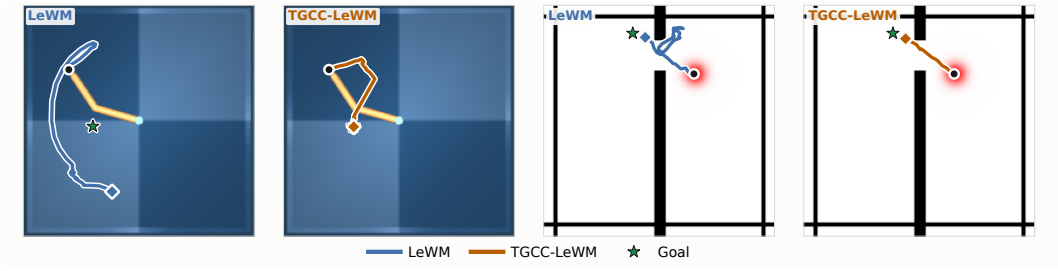

Figure 4: **Matched trajectory comparison on Reacher and Two-Room.** Left: LeWM follows a long exploratory trajectory and fails on Reacher, whereas TGCC-LeWM reaches the goal quickly. Right: both methods succeed on Two-Room, but TGCC-LeWM follows a substantially shorter route while LeWM incurs considerable unnecessary motion. All comparisons use matched start-goal settings and planning budgets.

4.5 Qualitative Analysis

To answer Q4, we combine quantitative path-efficiency evaluation with matched qualitative trajectories. As shown in Table 4, TGCC reduces steps-to-success in all 18 directly comparable settings across both PLDM and LeWM. Figure 4 further shows that TGCC-LeWM reaches the goal quickly where LeWM fails after prolonged exploration on Reacher, and follows a substantially shorter route when both methods succeed on Two-Room. Together, these results demonstrate that the calibrated planning geometry improves both closed-loop effectiveness and path efficiency under matched planning conditions.

4.6 Efficiency Analysis

To answer Q5, we separately measure the one-time offline fitting cost and online planning overhead of the fixed TGCC configuration. Across the four environments, encoding approximately 200K observations takes 112-136 s, while fitting the network for 50 epochs takes 9.7–10.2 s. The control-geometry network adds 755,201 parameters, equivalent to 4.2% of LeWM. During online planning, TGCC increases the median latency of a complete 30-iteration CEM solve by 8.8–10.7 ms, corresponding to 2.4%–3.1% across environments. Detailed results are provided in Appendix D.

5 Discussion and Limitations

The matched LeWM and PLDM results identify the mismatch between native latent similarity and directional control cost as a distinct source of planning error. TGCC mitigates this error by constructing a trajectory-graph control-cost estimator and incorporating it into the planning objective.

However, because the control-cost estimator operates on backbone latents and predicted rollouts, it inherits their observability and prediction limitations. TGCC cannot reconstruct task-relevant variables discarded by the encoder or correct inaccurate dynamics predictions. Extending TGCC toward jointly adaptive representations and more globally connected trajectory graphs remains an important direction.

6 Conclusion

TGCC disentangles the geometry used for planning from that learned for prediction while keeping the world model frozen, by constructing a directed trajectory graph from offline trajectories and distilling its temporal structure into a latent control geometry. By combining this graph-derived control geometry with the native prediction geometry, TGCC substantially strengthens latent-world-model planning and consistently shortens successful trajectories across multiple datasets and backbones.

AI USE STATEMENT

AI tools were used to polish the writing and check the manuscript. The human authors reviewed the AI-assisted work and take responsibility for the final content, including all text, claims, code, and artifacts.

REFERENCES

Mahmoud Assran et al. Self-supervised learning from images with a joint-embedding predictive architecture. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.

Mido Assran et al. V-JEPA 2: Self-supervised video models enable understanding, prediction and planning. arXiv preprint arXiv:2506.09985, 2025. URL https://arxiv.org/abs/2506.09985.

Marco Bagatella, Miroslav Olšák, Michal Rolínek, and Georg Martius. Planning from pixels in environments with combinatorially hard search spaces. In *Advances in Neural Information Processing Systems*, volume 34, pp. 24707–24718. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper_files/paper/2021/hash/cf708fc1decf0337aded484f8f4519ae-Abstract.html.

Adrien Bardes, Quentin Garrido, Jean Ponce, Xinlei Chen, Michael Rabbat, Yann LeCun, Mido Assran, and Nicolas Ballas. Revisiting Feature Prediction for Learning Visual Representations from Video. *Transactions on Machine Learning Research*, 2024. URL https://openreview.net/forum?id=QaCCuDfBk2.

Onur Beker, Mohammad Mohammadi, and Amir Zamir. PALMER: Perception–action loop with memory for long-horizon planning. In *Advances in Neural Information Processing Systems*, volume 35, pp. 34258–34271, 2022.

Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. *The International Journal of Robotics Research*, 44(10–11):1684–1704, 2025. doi: 10.1177/02783649241273668.

Kurtland Chua, Roberto Calandra, Rowan McAllister, and Sergey Levine. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

Ben Eysenbach, Ruslan Salakhutdinov, and Sergey Levine. Search on the replay buffer: Bridging planning and reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 32, pp. 15220–15231, 2019. URL https://proceedings.neurips.cc/paper/2019/hash/5c48ff18e0a47baaf81d8b8ea51eec92-Abstract.html.

Shenyuan Gao, Siyuan Zhou, Yilun Du, Jun Zhang, and Chuang Gan. AdaWorld: Learning adaptable world models with latent actions. In *Proceedings of the International Conference on Machine Learning (ICML)*, volume 267 of *Proceedings of Machine Learning Research*, pp. 18744–18771, 2025. URL https://proceedings.mlr.press/v267/gao25u.html.

Robert Gieselmann and Florian T. Pokorny. Latent planning via expansive tree search. In *Advances in Neural Information Processing Systems*, volume 35, pp. 16821–16835, 2022.

Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2019.

Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2020.

Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering diverse control tasks through world models. *Nature*, 640(8059):647–653, 2025. doi: 10.1038/s41586-025-08744-2.

Nicklas Hansen, Hao Su, and Xiaolong Wang. TD-MPC2: Scalable, robust world models for continuous control. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2024. URL https://openreview.net/forum?id=Oxh5CstDJU.

Lucas Maes, Quentin Le Lidec, Damien Scieur, Yann LeCun, and Randall Balestriero. LeWorld-Model: Stable end-to-end joint-embedding predictive architecture from pixels. arXiv preprint arXiv:2603.19312, 2026. URL https://arxiv.org/abs/2603.19312.

Lina Mezghani, Sainbayar Sukhbaatar, Piotr Bojanowski, Alessandro Lazaric, and Karteek Alahari. Learning goal-conditioned policies offline with self-supervised reward shaping. In *Proceedings of The 6th Conference on Robot Learning*, pp. 1401–1410, 2023.

Vincent Micheli, Eloi Alonso, and François Fleuret. Transformers are sample-efficient world models. In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=vhFu1Acb0xb.

Seohong Park, Kevin Frans, Benjamin Eysenbach, and Sergey Levine. OGBench: Benchmarking offline goal-conditioned RL. In Y. Yue, A. Garg, N. Peng, F. Sha, and R. Yu (eds.), *International Conference on Learning Representations*, volume 2025, pp. 94937–94982, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/ecd92623ac899357312aaa8915853699-Paper-Conference.pdf.

Reuven Y. Rubinstein and Dirk P. Kroese. *The Cross-Entropy Method: A Unified Approach to Combinatorial Optimization, Monte-Carlo Simulation and Machine Learning*. Springer, 2004.

Nikolay Savinov, Alexey Dosovitskiy, and Vladlen Koltun. Semi-parametric topological memory for navigation. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=SygwwGbRW.

Dhruv Shah, Benjamin Eysenbach, Gregory Kahn, Nicholas Rhinehart, and Sergey Levine. ViNG: Learning open-world navigation with visual goals. In *2021 IEEE International Conference on Robotics and Automation*, pp. 13215–13222. IEEE, 2021. doi: 10.1109/ICRA48506.2021.9561936.

Uladzislau Sobal, Wancong Zhang, Kyunghyun Cho, Randall Balestriero, Tim G. J. Rudner, and Yann LeCun. Learning from reward-free offline data: A case for planning with latent dynamics models. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen (eds.), *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pp. 43905–43941. Curran Associates, Inc., 2025. doi: 10.52202/085713-1465. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/3e7cf447f21cd11c846463affefce665-Paper-Conference.pdf.

Yuval Tassa, Yotam Doron, Alistair Muldal, Tom Erez, Yazhe Li, Diego de Las Casas, David Budden, Abbas Abdolmaleki, Josh Merel, Andrew Lefrancq, Timothy P. Lillicrap, and Martin Riedmiller. DeepMind Control Suite. arXiv preprint arXiv:1801.00690, 2018. URL https://arxiv.org/abs/1801.00690.

Jialong Wu, Shaofeng Yin, Ningya Feng, Xu He, Dong Li, Jianye Hao, and Mingsheng Long. iVideoGPT: Interactive videogpts are scalable world models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/7dbb5bfab324e3b86af9bd0df15498dd-Abstract-Conference.html.

Lunjun Zhang, Ge Yang, and Bradly C. Stadie. World model as a graph: Learning latent landmarks for planning. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 12611–12620. PMLR, 2021. URL https://proceedings.mlr.press/v139/zhang21x.html.

Gaoyue Zhou, Hengkai Pan, Yann LeCun, and Lerrel Pinto. DINO-WM: World models on pre-trained visual features enable zero-shot planning. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 79115–79135. PMLR, 2025.

APPENDIX

A SUPPLEMENTARY OVERVIEW

This supplementary material provides the implementation and evaluation details of Trajectory-Graph Control Calibration (TGCC). It uses the formulation and terminology introduced in the main paper. TGCC constructs a directed trajectory graph from offline transitions, converts graph connectivity into multi-budget forward-reachability relations, and distills these relations into a continuous ordered-pair control metric. During MPC, this graph-derived control cost is calibrated with the native terminal latent cost to define a planning geometry distinct from the prediction geometry.

The supplement focuses on implementation details, evaluation protocol, reproducibility settings, computational cost, hyperparameter selection, and the additional details that are not included in the main paper.

B ONLINE PAIR CONSTRUCTION AND MULTI-BUDGET LABELING

To avoid enumerating all $|V|^2$ ordered pairs, training uses the pair proposals listed below. Bounded BFS on the graph, including accepted stitching edges, determines reachability under the considered budgets. Forward-reachable neighborhoods are partitioned into budget shells, allowing one source–target pair to contribute labels at all budgets simultaneously.

Six pair classes are used:

1. **Forward-reachable pairs:** targets with $d_{\mathcal{G}}(v_i, v_j) \leq B_{\max}$.
2. **Forward out-of-budget pairs:** targets with $B_{\max} < d_{\mathcal{G}}(v_i, v_j) < \infty$.
3. **Reverse-time pairs:** ordered reversals of sampled forward-reachable pairs.
4. **Different-trajectory pairs:** randomly sampled endpoints from different episodes for which no directed path is found within $B_{\max}$.
5. **Same-episode kNN pairs:** latent-neighbor proposals whose endpoints lie in the same episode; their labels are determined by $d_{\mathcal{G}}$.
6. **Rejected cross-episode kNN pairs:** cross-episode neighbor proposals that do not satisfy the stitching criterion and are retained as low-confidence training pairs; their labels are determined by $d_{\mathcal{G}}$.

Pair sampling. For each of the six pair classes, we request 50,000 training pairs and 10,000 validation/test pairs, corresponding to a nominal uniform allocation of $1/6$ per class. The realized proportions may deviate from $1/6$ when a class contains fewer valid proposals than the requested quota. For example, in Two-Room, the same-episode hard-negative class contains 28,979 training pairs, accounting for 10.39% of the realized training pairs.

Accepted mutual cross-episode neighbors are graph-stitching edges rather than a separate training-pair class. For every pair class, labels follow Eq. (12) on the graph after stitching. In particular, reversing a pair or rejecting a proposed direct edge does not by itself make the pair negative: an alternative directed path within budget yields a positive label. A pair not reached within $B_{\max}$ has zero labels only for the considered budgets, without asserting physical unreachability.

Observed forward relations receive confidence $w_{ij} = 1$. Reverse and cross-trajectory pairs receive $w_{ij} = 0.1$, since a missing graph path may reflect incomplete offline coverage rather than physical impossibility. The confidence affects optimization only and never creates or removes graph edges.

Cross-episode nearest-neighbor pairs serve two distinct purposes. Mutual neighbors satisfying the stitching threshold are added as bidirectional graph edges. Cross-episode proposals that do not satisfy the stitching criterion are retained as low-confidence training pairs and never modify the graph. Their labels are determined by budgeted reachability on the graph after stitching, including paths through intermediate nodes. For stitching, we use mutual k-nearest-neighbor matching with $k = 5$. The retained-distance threshold is the 5th percentile of distances between mutual cross-episode neighbors in the training split. The threshold is computed independently for each environment and then held fixed for validation, test-pair construction, and all downstream evaluations.

Table S1: Transition budgets used to generate graph-derived reachability relations.

Environment	Transition budgets $\mathcal{B}$
Two-Room	$\{1, 2, 3, 4, 5, 8, 10, 20, 64\}$
Reacher	$\{1, 2, 3, 4, 5, 8, 10, 20, 128\}$
PushT	$\{1, 2, 3, 4, 5, 8, 10, 20, 64\}$
Cube	$\{1, 2, 3, 4, 5, 8, 10, 20, 128\}$

On Two-Room, this procedure uses a training-split fifth-percentile distance threshold of 0.0789 and accepts 16,450 cross-episode state pairs. Since accepted matches are inserted bidirectionally, they contribute 32,900 directed stitching edges, increasing the graph from 197,738 within-trajectory edges to 230,638 total directed edges. A physical sanity check of the accepted Two-Room matches finds no stitching edge that directly crosses the separating wall.

Trajectory-level train, validation, and test splits are disjoint. Pair mini-batches are resampled with independent shuffling across epochs, whereas the underlying graph, stitching edges, and bounded-BFS neighborhoods remain fixed. Model selection uses the validation correlation between h_η and finite graph distance; closed-loop test episodes are not used to select the control-geometry checkpoint.

Network implementation. Let $u = \mathrm{LN}(z_i)$ and $v = \mathrm{LN}(z_j)$. The pairwise network takes the ordered feature vector

$$\phi(u, v) = [u;\, v;\, v - u;\, |v - u|;\, u \odot v] \in \mathbb{R}^{5d},$$

and maps $5d \to 512 \to 512 \to 1$ using GELU hidden activations and a softplus output. The ordered features u, v, and $v - u$ allow the scorer to distinguish source and target states, while $|v - u|$ and $u \odot v$ provide symmetric pairwise interactions. For the $d = 192$ LeWM latent, the network contains 755,201 trainable parameters. The head is optimized for 50 epochs using AdamW, batch size 4096, learning rate 10^{-3}, weight decay 10^{-4}, and cosine learning-rate decay. Validation is performed every five epochs, and the checkpoint with the highest validation Spearman correlation to finite graph distance is retained.

C Experimental Protocol

Randomness and runs. All random generators in Python, NumPy, PyTorch, CEM, and the task environment are initialized from the evaluation seed. The five-seed LeWM analyses use $\{0, 1, 2, 7, 42\}$; each seed evaluates 50 episodes. The PLDM Reacher analysis uses $\{0, 1, 2, 3, 7, 13, 42\}$ with 50 episodes per seed.

The integer results in the main comparison table follow the matched benchmark configuration used by the corresponding backbone baselines. The five-seed component analysis reported in Appendix F is a separate robustness evaluation with 50 episodes per seed and therefore does not numerically coincide with the headline benchmark entries.

Control-cost rollout readout. For all LeWM experiments, the CEM prediction horizon is $H = 6$, and the control score in Eq. (18) is evaluated at $\mathcal{K} = \{2, 3, 5, 6\}$ using one-based indexing. For PLDM, $H = 15$, and the soft-min is computed over all predicted states, $\mathcal{K} = \{1, \ldots, 15\}$.

Computing infrastructure. Training and evaluation use an NVIDIA GeForce RTX 4090 with 24 GB memory under Ubuntu 24.04. The implementation uses Python 3.10, PyTorch 2.4, CUDA 12.1, and MuJoCo 3.1. Exact package hashes, CPU and host-memory information, and environment configuration files will accompany the released implementation.

D Detailed Offline and Online Overhead Analysis

TGCC is fitted post hoc from existing offline trajectories and requires neither world-model retraining nor additional environment interaction. Table S2 reports the latent-extraction cost, control-metric fitting time, and full-CEM latency.

Encoding approximately 200K observations requires 112–136 s per task, while fitting the control-metric network for 50 epochs requires 9.7–10.2 s. The network contributes 755,201 parameters, equal

Table S2: Offline fitting and online planning cost of the fixed TGCC configuration. Online latency measures a complete 30-iteration, 300-candidate CEM solve.

Task	Dump (s)	Fit (s)	LeWM (ms)	TGCC (ms)	Overhead
Two-Room	112.368	9.701	352.936	362.081	2.591%
Reacher	136.144	9.948	359.913	368.703	2.442%
PushT	134.133	10.018	351.609	362.351	3.055%
Cube	115.182	10.217	360.261	369.916	2.680%

to approximately 4.19% of LeWM. During planning, TGCC adds 8.8–10.7 ms per complete CEM solve, corresponding to 2.4–3.1% overhead. Peak allocated GPU memory increases by approximately 3 MiB.

The timing benchmark includes model rollout, both cost evaluations, candidate-wise normalization, elite selection, and CEM distribution updates. It excludes simulator stepping, rendering, video encoding, dataset construction, and one-time checkpoint loading.

E Hyperparameter Selection

Table S3 lists the search ranges, final values, and selection criteria used for the reported implementation. Planning hyperparameters are selected on a held-out validation start–goal manifest, whereas graph and control-network hyperparameters are selected using held-out trajectory relations. No test episode is used in either selection procedure.

Table S3: Hyperparameter ranges and final values. Selection uses held-out validation trajectories and does not use closed-loop test episodes.

Parameter	Values considered	Final	Selection criterion
Planning weight α	$\{0.10, 0.25, 0.50\}$	0.25	Mean validation success
Soft-min temperature β	$\{0.5, 1.0, 2.0\}$	1.0	Validation path-cost correlation
Budget temperature τ	$\{1.0, 2.0, 4.0\}$	2.0	Validation graph-relation loss
Hidden width	$\{256, 512\}$	512	Validation correlation and computational cost
Uncertain-pair weight	$\{0.05, 0.10, 0.20\}$	0.10	Validation graph-relation loss
Stitching neighbors k	$\{3, 5, 10\}$	5	Validation reachability correlation
Stitching percentile	$\{2.5, 5, 10\}$	5	Validation reachability correlation
Learning rate	$\{3\times10^{-4}, 10^{-3}\}$	10^{-3}	Validation convergence

F Additional Implementation Details

Consistent with the main paper, all TGCC planning results use the fixed mixing coefficient $\alpha = 0.25$; no environment-specific or candidate-specific mixing coefficient is used.

F.1 Pair-Label Coverage and Graph Statistics

Table S4 summarizes the realized pair-label coverage statistics reported for Two-Room, Reacher, and PushT. Coverage is the number of obtained pairs divided by the requested number across the six pair classes. "Uncertain" denotes the fraction assigned to reverse, different-trajectory, or rejected cross-episode nearest-neighbor pairs.

Nearest-neighbor indices are constructed independently within the training, validation, and test splits. Consequently, neither retained pair endpoints nor pair proposals depend on latent rows from another split. The stitching threshold is estimated on the training split and then held fixed for validation and test. As defined in Appendix B, mutual cross-episode neighbors satisfying the stitching threshold become bidirectional graph edges. Cross-episode proposals that do not satisfy that criterion remain low-confidence training pairs and never modify the graph. Their labels follow Eq. (12), including any alternative paths in the graph after stitching. The absence of observed forward support is not interpreted as proof of physical impossibility.

Table S4: Realized pair-label coverage and the fractions of uncertain pairs and pairs with finite directed graph distance.

Task	Split	Coverage	Uncertain	Finite $d_{\mathcal{G}}$
Two-Room	Train	0.929	0.538	0.411
	Validation	0.925	0.541	0.410
	Test	0.931	0.537	0.411
Reacher	Train	0.997	0.501	0.416
	Validation	0.998	0.501	0.417
	Test	0.998	0.501	0.418
PushT	Train	1.000	0.500	0.416
	Validation	1.000	0.500	0.415
	Test	1.000	0.500	0.417

Table S5: Base trajectory-graph statistics before cross-episode stitching. Edges count only recorded within-episode adjacent transitions.

Task	Nodes	Paths	Edges	Degree
Two-Room	199,910	2,172	197,738	0.9891
Reacher	199,995	995	199,000	0.9950
PushT	199,984	1,610	198,374	0.9919
Cube	199,995	995	199,000	0.9950

F.2 Planning-Geometry Alignment

Table S6 evaluates alignment with an obstacle-aware shortest-path reference on Two-Room. The state-field measurements compare the native terminal latent cost with the graph-derived score $h_\eta(z, z_g)$ on states sampled throughout the map. The CEM-pool measurements compare the native candidate objective with the combined TGCC planning objective C_{TGCC} on five matched model-predicted candidate populations at one planning state; both objectives score the same candidates within each population.

Table S6: Alignment with obstacle-aware control structure in Two-Room. For environment states, the LeWM column reports native terminal latent cost and the TGCC column reports the graph-derived score $h_\eta(z, z_g)$. For CEM rollouts, the LeWM column reports the native candidate objective and the TGCC column reports the combined, candidate-wise-normalized planning objective C_{TGCC}. This comparison uses five matched CEM candidate populations at one planning state; both objectives score the same candidates within each population. Higher Spearman correlation and lower inversion are better.

Diagnostic	Native cost	Graph-derived / TGCC cost
Field Spearman ↑	0.0089	0.9492
Field inversion ↓	0.4986	0.0959
Across-wall Spearman ↑	0.1421	0.7705
CEM-pool Spearman ↑	0.4422 ± 0.0306	0.6027 ± 0.0300
CEM-pool inversion ↓	0.3468 ± 0.0117	0.2854 ± 0.0123

F.3 Component Ablation

Cross-episode stitching. We further isolate the effect of trajectory stitching on Two-Room and PushT. Both variants use the same backbone, control-geometry architecture, planning weight $\alpha = 0.25$, evaluation seeds, and matched start-goal episodes; the only difference is whether accepted cross-episode matches are inserted into the trajectory graph before graph-relation labeling and control-geometry fitting.

Table S7: Five-seed component ablation of TGCC-LeWM, reported as closed-loop success rate (%). w/o CG retains only the native prediction-geometry cost, whereas w/o PG retains only the graph-derived control-geometry cost. w/o GD retains both geometry terms but removes directional information from the graph-derived term.

Task	TGCC	w/o CG	w/o PG	w/o GD
Two-Room	96.8 ± 3.2	88.0 ± 4.9	99.2 ± 1.0	95.6 ± 2.2
Reacher	96.4 ± 1.5	84.4 ± 8.4	89.2 ± 2.3	93.6 ± 3.0
PushT	91.2 ± 4.5	86.4 ± 3.9	68.0 ± 6.5	88.0 ± 3.7
Cube	74.0 ± 6.0	61.2 ± 8.0	57.6 ± 5.7	64.0 ± 7.6

Table S8: Effect of cross-episode trajectory stitching on Two-Room and PushT. Success rates are reported as mean $\pm$ standard deviation over five seeds. Two-Room evaluates 50 matched episodes per seed; its steps-to-success is evaluated on the 236 episodes solved by both variants.

Variant	Two-Room Success (%) $\uparrow$	PushT Success (%) $\uparrow$	Two-Room Steps-to-success $\downarrow$
w/o Cross-Episode Stitching	94.8 ± 2.3	88.8 ± 4.1	13.71
TGCC w/ Stitching	$\mathbf{96.8 \pm 3.2}$	$\mathbf{91.2 \pm 4.5}$	**13.28**

On Two-Room, the stitched variant uniquely succeeds on 6 of the 250 matched episodes, whereas the no-stitching variant uniquely succeeds on 1 episode. The remaining 236 episodes are solved by both variants. On PushT, stitching increases mean success by 2.4 percentage points.

Table S9: Five-seed comparison with the native LeWM planner under the separate robustness protocol (50 episodes per seed), reported as closed-loop success rate (%, mean $\pm$ standard deviation). The LeWM row is identical to the w/o Control Geometry variant in Table S7.

Method	Two-Room	Reacher	PushT	Cube
LeWM	88.0 ± 4.9	84.4 ± 8.4	86.4 ± 3.9	61.2 ± 8.0
TGCC-LeWM	96.8 ± 3.2	96.4 ± 1.5	91.2 ± 4.5	74.0 ± 6.0

F.4 Matched Qualitative Comparisons

Figure S1 shows one matched start–goal comparison for each environment under the fixed, ungated TGCC configuration ($\alpha = 0.25$). In every example, the native LeWM planner and TGCC-LeWM use the same environment, dataset episode, start state, goal state, and LeWM checkpoint. The panels report the actual final decoded rollout frames rather than padded visualization frames.

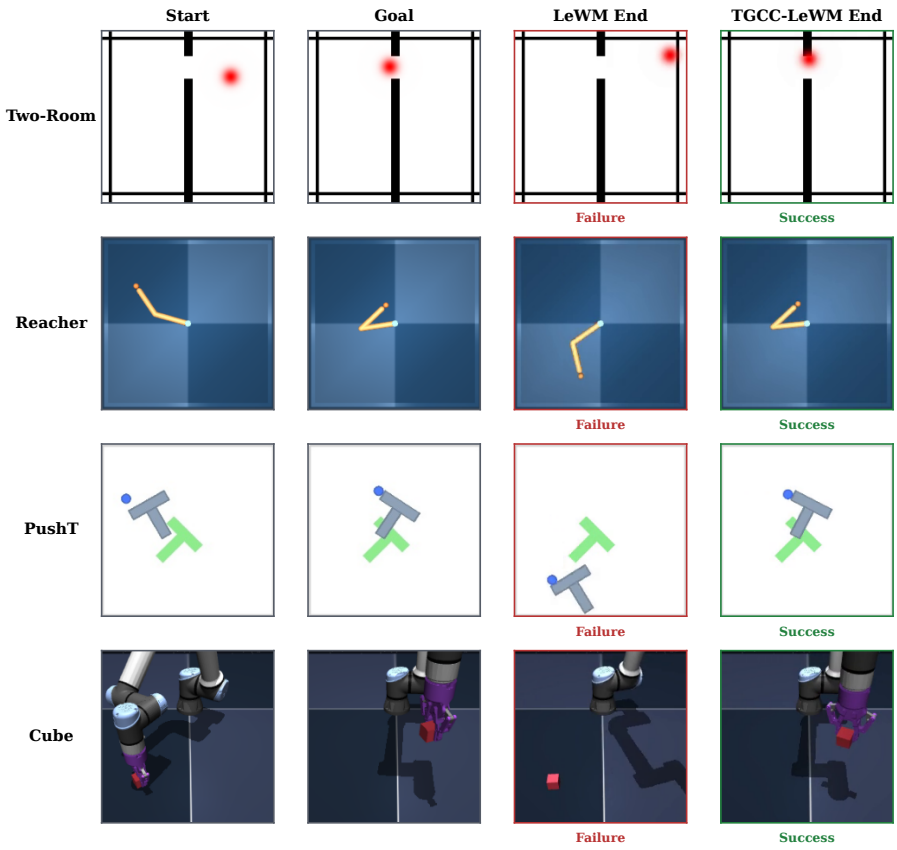

Figure S1: Matched qualitative comparisons across all four environments using fixed, ungated TGCC with $\alpha = 0.25$. Each row shows the exact start and goal observations followed by the final decoded rollout frames under LeWM and TGCC-LeWM. The selected examples are matched cases in which LeWM fails and TGCC-LeWM reaches the goal; they are qualitative examples rather than estimates of outcome frequency.

The accompanying Media Supplement provides planning videos for LeWM and TGCC-LeWM across different datasets and evaluation settings. In each video, the leftmost column shows the model-planned rollout, the middle column shows the ground-truth trajectory from the start state to the goal state, and the final column shows the goal state. The relevant visual criterion is the similarity between the final frame of the model-planned rollout and the goal-state frame; it is independent of the intermediate ground-truth trajectory displayed in the middle column.

F.5 Additional TGCC-PLDM Results

We evaluate TGCC-PLDM on Reacher using seven seeds $\{0, 1, 2, 3, 7, 13, 42\}$ with 50 episodes per seed. The per-seed success rates are $(46, 34, 38, 46, 38, 38, 44)$ for PLDM and $(62, 56, 62, 62, 80, 48, 56)$ for TGCC-PLDM, yielding 40.6 ± 4.7 and 60.9 ± 9.9, respectively. A two-sided paired Student t-test gives $p = 0.002524$.

G Code and Data Availability

The benchmark environments and source datasets are publicly available from their original releases. The TGCC implementation, graph-construction code, training configurations, evaluation manifests, and analysis scripts will be released upon publication under the Apache-2.0 license. The release will include the exact random seeds, software environment, final hyperparameters, and commands required to reproduce every reported table.