EXECUTABILITY, NOT OPTIMALITY: OFFLINE PLANNING ON LEARNED COGNITIVE MAPS

Anonymous authors
Paper under double-blind review

ABSTRACT

Animals reach places they were never shown how to reach, and hippocampal place cells (O'Keefe & Dostrovsky, 1971) are thought to support this by encoding a predictive map of where the animal can get to next (Stachenfeld et al., 2017). One computational form of that map is the multi-scale random-walk transition kernel (Zhao et al., 2025), whose non-negative spectral embeddings reproduce localized place fields, compose across scales by matrix squaring, and yield a route by climbing a similarity field rather than searching a graph. Whether it survives being fitted to offline trajectories, which supply neither the known lattice nor the exact kernel its demonstrations assume, and what it is then worth as a planner, has been open. Here we show that it survives the fit and that what its ascent buys over shortest-path search on the same support is executability, not optimality: it leads A* on 13 of 16 embodied OGBench datasets (Park et al., 2025), and support geometry alone recovers three quarters of what that preference is worth to search. Two received rules fail there: greedy ascent (Zhao et al., 2025), sufficient on the exact kernel, falls on the fitted map into limit cycles that one sampled alternative (Lin et al., 2026) escapes; and distance-optimal search loses to the map, while fast marching (Sethian, 1996) told what the ascent prefers, as a cost $n(x)^{-1/2}$ on the visit count $n(x)$, beats everything here, 20 points past the published baselines. Offline planning on a recovered support is therefore not a shortest-distance problem: the support's boundary is not the world's, and the same cost improves every planner we tested. A predictive map is a usable offline planner whose value lies less in the routes it produces than in what it reveals about which routes are worth producing.

1 INTRODUCTION

Reaching a goal one was never trained to reach is a hallmark of flexible behavior, and the hippocampal–entorhinal system is the standard biological reference for how it is done (Tolman, 1948; O'Keefe & Dostrovsky, 1971; Behrens et al., 2018). The successor representation (SR) makes that computational (Dayan, 1993; Stachenfeld et al., 2017), and Zhao et al. (2025) make it multi-scale and give it a read-out: place cells as non-negative spectral embeddings of the τ-step random-walk transition kernel, scale composed by matrix squaring, and a route produced by greedy ascent over the eight lattice neighbors. That is the whole of the biological motivation we draw on, and Section 4.1 is where we check that the object we fit still has the property the reference is named for; everything else in this paper is a measurement about planning.

Figure 1 states the method and Figure 2 shows what it amounts to on a real map. It is attractive as a planner because it is entirely representational: no graph is searched, and a route is produced by climbing a similarity field. But it has been demonstrated only where the lattice and its obstacles are given and the kernel is computed exactly, and offline reinforcement learning supplies neither.

The construction survives the fit, and the fit is loose. Measured against the exact kernel over every pair of free cells, the rank correlation runs 0.98–1.00 from $\tau = 256$ upward and only 0.70–0.89 at $\tau \leqslant 32$ (Section F). The quantity the ascent consults survives anyway: on the states where the exact field has a strict maximum, the fitted field prefers the same neighbor 99.2% of the time. The greedy ascent of Zhao et al. (2025) still stalls, and every stall we logged is a limit cycle rather than a wrong step, so the repair belongs in the read-out. We take it from Lin et al. (2026), who argue that planning on a cognitive map is better understood as stochastic sampling than as maximization: drawing the

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.

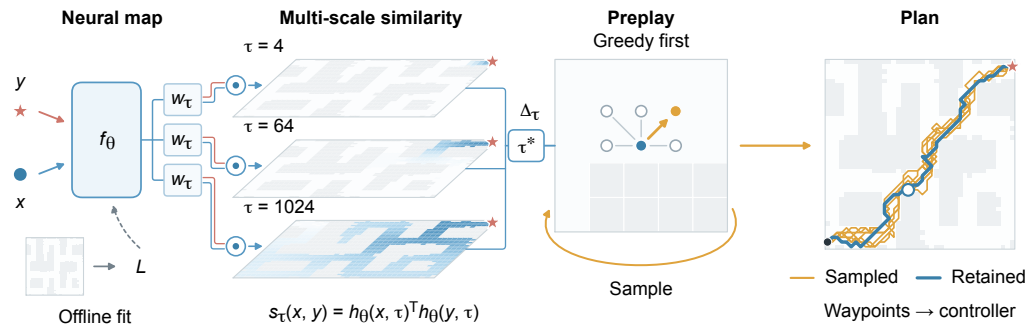


Figure 1: **Preplay: a map fitted offline and read by sampled ascent.** A network f_θ is fitted offline to the multi-scale transition kernel of the cells the data visited, giving a non-negative embedding at each of eleven scales. A route from x to y is read out by climbing the similarity $s_\tau(x, y) = h(x, \tau)^\top h(y, \tau)$ at the scale carrying the strongest signal. The deterministic ascent is run to completion first and its route kept if it arrives; only if it does not are K sampled rollouts drawn, of which the shortest arrival becomes the plan. The waypoints are handed to the low-level controller, and nothing is searched at planning time.

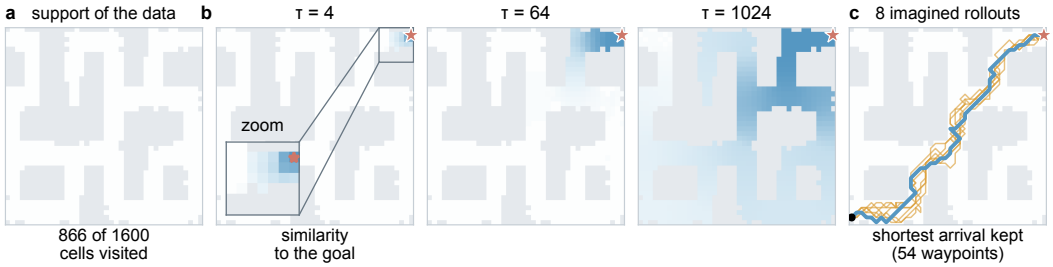


Figure 2: **Preplay on the map it builds.** (a) The support of the data, 866 of 1600 cells, is the only description of the environment the method receives. (b) The fitted embedding against the goal (star) at three of eleven scales; the inset shows the shortest, which is a few cells wide. (c) Eight rollouts from Equation (5) at $T = 0.3$; the shortest arrival becomes the plan.

next step from a softmax over the improvement it offers takes `pointmaze-large-navigate` from 74.1% to 100% and leaves the exact kernel at the 100% the greedy rule already reaches there. We call the resulting planner Preplay, after the hippocampal sequences that run ahead of the animal at a choice point (Dragoi & Tonegawa, 2011; Johnson & Redish, 2007); the name describes rollouts imagined before the body moves and claims nothing about the statistics of those sequences. Carvalho et al. (2025) use the word for a different mechanism, one that trains a representation where ours reads one already fitted; we note the collision and keep the term.

With that in place, the map is a working offline planner: under one configuration on all 24 OGBench state-based maze datasets, it is best on 13 against baselines (Park et al., 2025). Seven of the thirteen are PointMaze, where a plain shortest-path search on the same free set matches or beats us on seven of eight; there the free set carries the whole result and any planner reading it saturates. The sixteen embodied datasets are where a read-out is actually tested, and there we lead on six: five clearly, plus `antmaze-large-stitch` at 67.1 against the 67 ± 5 of HIQL, a tie. All six are `stitch` or `explore` data whose trajectories never traverse the test route end to end.

The free set is itself a graph, and the ascent beats searching it. On the sixteen embodied datasets it leads A* (Hart et al., 1968) on 13 of 16 and a cost-to-go sweep in the manner of fast marching (Sethian, 1996), which we label FMM, on 11 of 16, by 5.1 and 4.1 points; over all 24 both records are 14–7–3. Both optimize distance on the same nodes, so the sharper question is what the ascent *prefers* instead, and the answer is the main result of this paper. Section 4.4 shows that it prefers executability to optimality and that the preference can be written as a cost: told to prefer the cells the behavior traversed, the same sweep goes from losing to the map to beating it on 21 of 24 datasets by 19.3 points. What divides the nine arms is therefore not learned map against classical search but whether a planner has been told what to prefer, and the telling is worth far more to a global search than to a local rule, +30.7 to FMM against +5.1 to the ascent.

If a classical planner reading the same free set is the strongest thing in this paper, why keep the map? Because the analysis needed it, and the analysis is what we defend. The cost that makes FMM the best planner here was read off the map rather than guessed, by a measurement anyone can repeat:

on identical start–goal pairs the ascent runs $1.20\times$ further from the support boundary than A* and through cells visited $1.40\times$ more often, at $1.05\times$ the length, and Equation (6) is that measurement written as a cost. Two properties come with the map itself. It is a *function*, returning an embedding at coordinates in no training pair, where a stored graph has no entry (Section 4.5); and its read-out needs no graph at planning time, though fitting costs more than one search would, so the saving is per query and never overall (Section 3.1).

2 Background

Place cells as multi-scale transition kernels. We follow the formulation of Zhao et al. (2025). Let a symmetric random walk on the state space have one-step transition matrix P_1, and write P_τ for its τ-step powers, obtained efficiently by repeated squaring, $P_{2\tau} = P_\tau^2$. Normalizing gives $q(y \mid x, \tau) = p(y \mid x, \tau)/\sqrt{p(x \mid x, \tau)\, p(y \mid y, \tau)}$, and the place-cell population at horizon τ, whose fields have radius of order $\sqrt{\tau}$, is a non-negative embedding $h(x, \tau) \in \mathbb{R}^n_{\geqslant 0}$ with

$$\langle h(x, \tau),\, h(y, \tau) \rangle \;\approx\; q(y \mid x, \tau). \tag{1}$$

Non-negativity and the inner-product structure induce sparsity, which is what makes the coordinates read as localized firing fields, and τ sets their size. Planning is greedy ascent on Equation (1) over the eight lattice neighbors, with an adaptively selected scale: from x toward goal y, the next position maximizes $\Delta(z, \tau) = q(y \mid z, \tau) - q(y \mid x, \tau)$ over a local neighborhood, at the scale $\tau^\star$ giving the strongest signal. What Zhao et al. (2025) add to the SR (Dayan, 1993; Stachenfeld et al., 2017) is multi-scale composition and a read-out; what their demonstrations assume is a known lattice and an exact kernel.

Planning on logged experience. Building a graph from a replay buffer and searching it is an established route to long-horizon goal reaching (Savinov et al., 2018; Eysenbach et al., 2019; Emmons et al., 2020), and our search arms are that construction on the recovered lattice, with the distance supplied by the lattice instead of learned: A* (Hart et al., 1968), and FMM, a cost-to-go sweep in the manner of fast marching (Sethian, 1996) implemented as uniform-cost search on the 8-connected free set (Section M). Table 2 is therefore Preplay against a given-metric Eysenbach et al. (2019) on the same nodes. Baek et al. (2025) search a graph over temporally reachable states to stitch sub-trajectories, paying per query where a fitted field pays once; we quote the numbers of Park et al. (2025) rather than benchmark against it. Diffusion planners (Janner et al., 2022; Ajay et al., 2023) store a conditional model of trajectories where we store a representation of reachability, and neither share our support nor decompose into planner and controller as Table 2 requires.

Offline goal-conditioned reinforcement learning. The methods we compare against are offline goal-conditioned reinforcement learning (GCRL) baselines that learn a policy or value function from the same datasets (Park et al., 2025), with hierarchical variants predicting sub-goals (Park et al., 2023) and quasimetric variants fitting a distance (Wang et al., 2023). The quasimetric line is closest in kind, learning a geometry and planning on it, and Table 1 shows it beating us on `pointmaze-teleport-navigate`, the one family where an asserted lattice cannot represent reachability and a learned distance can. The difference is in what gets estimated: a value function is a statistic of the behavior, our kernel of its support. The cost of Equation (6) is a count-based quantity of the kind offline RL uses for pessimism and support constraints (Kumar et al., 2020), applied here to shape a route rather than to penalize a value.

Sampling as planning. Lin et al. (2026) argue that planning on a cognitive map is better modeled as stochastic sampling than as maximization, and supply the read-out we adopt; Carvalho et al. (2025) use *preplay* for counterfactual simulation that *trains* a representation, where ours names a read-out applied to a map already fitted.

3 Method

3.1 Fitting the Map to Offline Experience

Only the planning is search-free, not the pipeline. Planning performs no search, with no queue, no frontier and no distance-to-go table, but *fitting* constructs the dense P_τ over $\mathcal{F}$ and squares it to $\tau = 2048$, at order $|\mathcal{F}|^3 \log \tau$, more than one run of A* on the same graph. The method changes

where the cost is paid, not whether a graph is built, and it bounds the setting to discretizations of the order $|\mathcal{F}| \approx 10^3$ used here (Section D).

Equation (1) presumes a known lattice and an exact kernel; offline data gives neither. We discretize the recorded positions on a $G \times G$ grid and take the *free set* $\mathcal{F}$ to be the cells the dataset visits. That set is the only description of the environment the method receives; no maze layout is supplied. On $\mathcal{F}$ we place the symmetric 3×3 random walk P_1 of Zhao et al. (2025), raise it to P_τ by repeated squaring, and normalize to obtain $q(\cdot \mid \cdot, \tau)$ at scales $\tau \in \{2, 4, \ldots, 2048\}$.

The map is a network: a shared trunk ϕ_θ of six residual layers reads Fourier position features $\gamma(x)$ with $L = 12$ bands at base frequency 0.05, and a per-scale linear head produces a non-negative unit-norm embedding

$$h_\theta(x, \tau) \;=\; \frac{\mathrm{ReLU}(W_\tau\, \phi_\theta(\gamma(x)))}{\|\mathrm{ReLU}(W_\tau\, \phi_\theta(\gamma(x)))\|_2} \;\in\; \mathbb{R}^n_{\geqslant 0}, \tag{2}$$

so that $\langle h_\theta(x, \tau), h_\theta(y, \tau)\rangle \in [0, 1]$ with self-similarity 1, matching the normalization of q. We fit it by regressing the full Gram matrix on the reference kernel,

$$\mathcal{L}(\theta) = \frac{1}{S} \sum_\tau \frac{\sum_{x,y \in \mathcal{F}} w_\tau(x, y) \left(\langle h_\theta(x, \tau), h_\theta(y, \tau)\rangle - q(y \mid x, \tau)\right)^2}{\sum_{x,y \in \mathcal{F}} w_\tau(x, y)}, \quad w_\tau(x, y) = q(y \mid x, \tau) + c. \tag{3}$$

Two details address pathologies of this regression. A short-τ kernel is almost diagonal, so an unweighted squared error is dominated by the zeros and the network can fit the long scales perfectly while leaving the ascent no signal to terminate on; the pair weight w_τ with $c = 0.05$ restores the occupied entries. And short-τ kernels vary over a few cells, which a network reading raw coordinates cannot produce, so we supply Fourier features at a low base frequency. The features are load-bearing, worth 18.4 points on `pointmaze-giant-navigate`. The pair weight is not: removing it improves both ablation columns, by 4.5 and 3.2 points, and we keep it only because every number here was produced under it, which makes them slightly pessimistic (Table A7).

At evaluation the tabular kernel is discarded and the map is the network: the planner sees nothing about the environment except which cells are free.

3.2 Preplay: Planning by Sampled Ascent

Given the map, a route from x to goal y is produced by climbing the similarity field. Writing

$$\Delta(z, \tau) \;=\; \langle h(y, \tau), h(z, \tau)\rangle - \langle h(y, \tau), h(x, \tau)\rangle \tag{4}$$

for the improvement offered by a candidate z at scale τ, the deterministic rule of Zhao et al. (2025) moves to $\arg\max_z \Delta(z, \tau^\star)$ at the scale $\tau^\star = \arg\max_\tau \max_z \Delta(z, \tau)$ carrying the strongest signal. On an exact kernel over a known lattice this is well behaved. On a fitted kernel the ascent stalls, and it stalls conspicuously: on PointMaze the plan and the episode succeed or fail together, so the stall accounts for the failures and not the controller (Section 4.2).

Our planner replaces the maximization with sampling, following Lin et al. (2026): noise in action selection produces a diversity of imagined trajectories and a self-correcting homing-in, in which a step that is wrong on its own is compensated later in the same rollout. From x we draw

$$z \;\sim\; \mathrm{softmax}\left(\frac{\Delta(\cdot, \tau^\star)}{T \cdot (\max_{z'} |\Delta(z', \tau^\star)| + \epsilon)}\right), \tag{5}$$

with $\epsilon = 10^{-12}$, which matters only on a flat plateau, where the draw is uniform. Normalizing by $\max_{z'}|\Delta|$ makes T comparable across scales and across maps whose gradients differ by orders of magnitude. Candidates z are the eight lattice neighbors of x lying in $\mathcal{F}$, the same neighborhood every search arm uses, so all of them plan on one graph. A rollout is a sequence of such draws, terminating at the goal cell or at a step cap of 400; $K = 16$ are imagined and the *shortest* that reaches y becomes the plan. The rule degenerates to a deterministic ascent as $T \to 0$, which is the greedy arm on PointMaze (Section M).

Two details of the rule. The greedy ascent is run to completion first and its route kept if it arrives, so where the deterministic rule suffices the planner is that rule exactly, and whether it stalls is decided on the map before the environment is touched. And among the rollouts that arrive we keep the shortest, since a noisy route costs budget on the largest mazes. Neither is what makes sampling work; escaping the cycle is (Section 4.2).

3.3 Execution

The plan is a sequence of cells, one per ascent step, whose world-coordinate centers are the waypoints, so spacing is one cell. The FMM arms descend their field a cell at a time and already match that spacing; the A* arms are shortcut and then resampled to it, so no arm gets sparser sub-goals than another. The executor advances to the next waypoint once the body is close enough and skips ahead when it stops making progress, with thresholds fixed per suite (Table A2). On PointMaze the waypoint is followed by an unlearned rule that drives at unit speed toward it; on the embodied suites by an implicit Q-learning (IQL) controller trained on the same dataset with hindsight-relabeled sub-goals, whose value function is fitted without actions and whose policy head is an ordinary state-conditioned actor (Section B). Both are identical across every arm we compare, so differences in the tables are attributable to the map and the planner. No part of Preplay performs graph search, and there is no shortest-path fallback: an episode in which the ascent fails to reach the goal is scored as the failure it is.

4 Experiments

We evaluate on OGBench, the offline GCRL suite of Park et al. (2025), following its protocol: five evaluation goals per dataset, fifty episodes each, the OGBench success flag with its own tolerance, and the episode budget registered for the environment, 1000 steps for the point and ant embodiments and 2000 and 4000 for the humanoid ones (Section A). Baseline numbers are as reported by Park et al. (2025); every Preplay cell is a mean over three independently fitted maps at 250 evaluation episodes, the sample size of one OGBench evaluation epoch, and the replicates of the search arms are stated with Table 2. Four things move a cell, the map, the reset block, the controller and the device, the last by 2.7 points on average and up to 6.8 with nothing else changed (Section E), so a few points on one embodied dataset is a tie, while a suite mean, whose standard error under that perturbation alone is 0.9, is less so.

4.1 The Fitted Map

Before planning on it we check that the map is the object the construction is named for. Nothing in the objective asks for localized units beyond non-negativity and the inner-product form, yet they come out localized (Figure A1). Under the definitions of Section G, the fraction of a unit's active cells lying within two radii of its own center is 0.94–0.99 at every scale on every map, and the code is sparse where the construction says it should be, about five of five hundred coordinates active at $\tau = 2$ and between 188 and 249 at $\tau = 2048$. Field size grows with the horizon as a power law with exponent 0.359, 0.363 and 0.385 on the three maps of Figure A1, short of the $\sqrt{\tau}$ an unbounded lattice would give; the shortfall is the arena and not the network, since on the four PointMaze datasets where we hold both kernels the exact tabular kernel gives 0.384–0.414 against 0.375–0.402 for the fitted map.

The fit is loose where a global statistic looks and tight where the ascent looks (Table A5): over every pair of free cells the rank correlation with the exact kernel is 0.98–1.00 from $\tau = 256$ upward and 0.70–0.89 at $\tau \leqslant 32$, the scales the ascent terminates on, but what an $\arg\max$ over eight neighbors consults is a local ordering, and on the states where the exact field has a strict maximum the fitted field prefers the same neighbor on 99.2% of them. The failures below are therefore failures of the read-out.

4.2 Fitting Breaks the Read-Out, Not the Fit

On the fitted map the greedy ascent of Zhao et al. (2025) stalls, and on PointMaze the plan and the episode succeed or fail together, so the stall accounts for the failures and not the controller. Whether that is a property of the rule or of the approximation is settled by the same read-out on the tabular kernel the network was fitted to (Table A6): on three PointMaze datasets greedy ascent on the exact kernel returns 100.0, 100.0 and 99.6, against 74.1 ± 8.5, 95.2 ± 4.8 and 86.0 ± 4.8 on the fitted map, and sampling adds nothing to the exact kernel while closing most of that gap on the fitted one. Greedy ascent is not a weak rule; what breaks it is the fit.

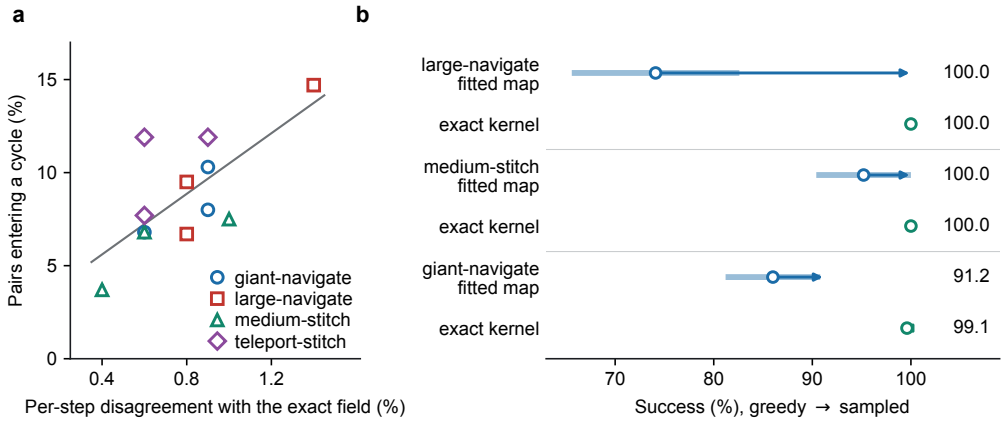


Figure 3: **Limit cycles on the fitted map.** (a) Per-step disagreement with the exact field against the fraction of start–goal pairs the deterministic ascent loses to a limit cycle, on four PointMaze datasets at three fitted maps each. Disagreement never exceeds 1.4% of the states with a strict maximum, yet 3.7–14.7% of routes fail, and the two move together ($r = 0.71$, $n = 12$). (b) The same read-out on the exact kernel and on the map fitted to it: the dot is the deterministic ascent with its spread, the arrow what sampling changes, and the number where it arrives. Spreads are s.d. over three replicates; exact-kernel greedy is a single deterministic run.

What approximation does to the field is neither of the two things one would expect. It does not flatten the field, and it barely moves the preferred neighbor, which agrees with the exact one on 99.2% of the states that have a strict maximum. One per cent of per-step disagreement does something a flatter or a shifted field cannot: it makes the failures absorbing. Every one of the 5061 deterministic rollouts that fails, over 57,549 start–goal pairs, revisits a cell it has already occupied rather than halting at a fixed point, and 99.98% of those cycles have length two, two cells each of which is the $\arg\max$ of the other (Figure 3). That is why sampling repairs it and why one alternative suffices: leaving a cycle needs only difference, not correctness. Section H gives the measurement in full and rules out the pair weight as its cause.

Sampling is accordingly what makes the fitted map usable: it leads the greedy ascent by 13.7 points on the embodied datasets and 13.0 over all 24, and takes the routed fraction from 58% to 98%, on the large and giant `antmaze` datasets from 7–17% to 91–96% (Table 2). The two details of the rule in Section 3.2 are not what makes sampling work (Table A7); escaping the cycle is, and one sampled rollout is already within 3.6 and 0.7 points of thirty-two on the two PointMaze datasets with headroom. Over the embodied suite the tabular kernel plans about as well as the network (7 of 16, -2.0), so distilling it into a function buys accuracy nowhere here; it buys queryability, which Section 4.5 measures.

4.3 Preplay Against the OGBench Baselines

Table 1 gives the full comparison. Preplay attains the best success rate on 13 of the 24 datasets under one configuration applied to all of them; the aggregate over all 24 is $+1.7$ points against the per-dataset strongest baseline, a tie, and not the interesting cut. Seven of the thirteen are PointMaze, where a plain shortest-path search on the same free set matches or beats us on seven of eight, so any planner reading that free set saturates. Split by what the data contains, the eleven stitch and explore datasets outside the teleport family, whose trajectories never traverse the test route end to end, give 9 of 11 and 15.1 points above the strongest reference baseline, including three on which no baseline exceeds 10 (`antmaze-giant-stitch`, 28.5 against 2; `antmaze-large-explore`, 42.8 against 10; `humanoidmaze-giant-stitch`, 6.9 against 3), while the thirteen navigate and teleport datasets, where whole demonstrations exist or the metric is broken, run 9.8 points below. Section 4.4 improves on those three by a further factor of three, with a planner this paper derives rather than proposes.

The column reports the sampled planner throughout, never the better of it and the deterministic ascent per dataset; and since its two constants were swept on two datasets that appear here, Table A8 also gives the untuned $T = 1$, $K = 1$ column, best on 11 of the 24 instead of 13 and 3.2 points lower on average. Nothing below depends on which is used.

Table 1: **Success rate on the 24 OGBench state-based maze datasets.** Baselines are as reported by Park et al. (2025) over eight seeds; Preplay is a mean over three fitted maps at 250 episodes each. The best entry in each row is bold on a cream cell and the runner-up underlined on a green one, so the ranking reads without color. The comparison is against the six reference baselines; Table 2 adds planners of our own, the strongest of which leads these baselines on 12 of the 16 embodied datasets.

Suite	Dataset	GCBC	GCIVL	GCIQL	QRL	CRL	HIQL	**Preplay**
PointMaze	medium-navigate	9±6	63±6	53±8	82±5	29±7	79±5	**100.0**±0.0
	large-navigate	29±6	45±5	34±3	86±9	39±7	58±5	**100.0**±0.0
	giant-navigate	1±2	0±0	0±0	68±7	27±10	46±9	**91.2**±5.3
	teleport-navigate	25±3	**45**±3	24±7	4±4	24±6	18±4	30.9±0.2
	medium-stitch	23±18	70±14	21±9	80±12	0±1	74±6	**100.0**±0.0
	large-stitch	7±5	12±6	31±2	84±15	0±0	13±6	**99.7**±0.4
	giant-stitch	0±0	0±0	0±0	50±8	0±0	0±0	**99.9**±0.2
	teleport-stitch	31±9	44±2	25±3	9±5	4±3	34±4	**52.9**±3.2
AntMaze	medium-navigate	29±4	72±8	71±4	88±3	95±1	**96**±1	85.2±1.4
	large-navigate	24±2	16±5	34±4	75±6	83±4	**91**±2	71.2±3.6
	giant-navigate	0±0	0±0	0±0	14±3	16±3	**65**±5	23.3±5.5
	teleport-navigate	26±3	39±3	35±5	35±5	**53**±2	42±3	21.1±0.5
	medium-stitch	45±11	44±6	29±6	59±7	53±6	**94**±1	77.5±1.2
	large-stitch	3±3	18±2	7±2	18±2	11±2	67±5	**67.1**±3.3
	giant-stitch	0±0	0±0	0±0	0±0	0±0	2±2	**28.5**±1.1
	teleport-stitch	31±6	**39**±3	17±2	24±5	31±4	36±2	19.3±1.9
	medium-explore	2±1	19±3	13±2	1±1	3±2	37±10	**85.7**±1.0
	large-explore	0±0	10±3	0±0	0±0	0±0	4±5	**42.8**±0.6
HumanoidMaze	medium-navigate	8±2	24±2	27±2	21±8	60±4	**89**±2	47.5±1.6
	large-navigate	1±0	2±1	2±1	5±1	24±4	**49**±4	40.0±3.8
	giant-navigate	0±0	0±0	0±0	1±0	3±2	**12**±4	9.5±3.3
	medium-stitch	29±5	12±2	12±3	18±2	36±2	**88**±2	65.9±0.5
	large-stitch	6±3	1±1	0±0	3±1	4±1	28±3	**35.6**±2.6
	giant-stitch	0±0	0±0	0±0	0±0	0±0	3±2	**6.9**±3.9

One asymmetry should be stated before the table is read: Preplay is three components, a discretization, a kernel fitted on the resulting free set, and a separately trained controller, against single end-to-end policies, so the comparison is not like for like. Two things bound how much that can flatter us. The search arms of Table 2 get the same free set, so within that table only the read-out differs; and the controller contributes nothing on its own, scoring 0.0 on eleven of the sixteen embodied datasets and below 0.5 on three more when handed the goal with no plan, 42.5 points below Preplay on average, the exceptions being the teleport mazes where start and goal are sometimes adjacent. With a body, planning is the whole result, so Table 1 is not an artifact of training the low level separately; it shows that the pipeline reaches goals end-to-end policies do not, without saying which part is responsible. The converse holds on the humanoid, where the ascent routes on every episode and the episodes are still lost at the budget, consuming 76%, 92% and 100% of the steps on medium, large and giant (Section K).

Support saturates long before behavior does. The kernel is a statistic of *where the data went* and a value function of *what the data did*: on PointMaze, whose unlearned controller makes a data reduction affect only the planner, Preplay holds 100.0, 98.7 and 86.5 at 1% of the training episodes where HIQL returns 2.1, 0.4 and 2.4 (Figure A4). The credit belongs to the recovered support and not to the fitted map: search on the same free set returns our value to the decimal at 1% on `giant-navigate`, failing on the same episodes (Section L).

4.4 Executability, Not Optimality

Table 2 varies nothing but where the waypoints come from, and the fitted embedding earns a few points over searching the same free set: over the embodied sixteen the ascent leads A* on 13 of 16 by 5.1 and FMM on 11 of 16 by 4.1, and over all 24 both records are 14–7–3, worth 2.4 and 5.0, while PointMaze reverses the margin, since a massless point executes any route and there search leads by 3.0. Both searches optimize distance on the same nodes, so the sharper question is which routes the ascent *prefers*.

Table 2: **Read-outs on one recovered support.** All arms share the free set, the controller and the executor, and differ only in where the waypoints come from. **none**: the controller is handed the goal and no plan. **greedy** and **Preplay**: deterministic and sampled ascent on the fitted map. **+cov**: Preplay with the bonus of Equation (6). **FMM** and **FMM-cov**: a cost-to-go field swept from the goal and then descended, without and with the same charge. Success (%) over 250 episodes and three replicates, three fitted maps for the arms that read one and three disjoint blocks of 250 episodes for the rest; spreads are population s.d. The Preplay column repeats Table 1. The five PointMaze datasets on which every arm scores about 100 are in Table A9 with the tabular-kernel and A* arms. Row best is bold on a cream cell, runner-up underlined on a green one.

Suite	Dataset	none	greedy	Preplay	+cov	FMM	FMM-cov
PointMaze	giant-navigate	0.0±0.0	86.0±4.8	91.2±5.3	99.2±1.1	60.0±0.0	**100.0±0.0**
	teleport-navigate	6.3±1.1	22.5±4.5	30.9±0.2	30.0±2.8	30.3±1.9	**31.6±0.7**
	teleport-stitch	6.8±1.0	51.2±6.6	52.9±3.2	32.0±1.3	29.7±1.5	**80.7±0.9**
AntMaze	medium-navigate	0.1±0.2	85.7±1.6	85.2±1.4	86.8±1.2	82.9±1.5	**96.1±1.3**
	large-navigate	0.0±0.0	16.9±9.7	71.2±3.6	77.3±3.6	64.9±2.9	**89.2±0.6**
	giant-navigate	0.0±0.0	6.7±5.7	23.3±5.5	30.9±9.5	29.5±1.8	**78.4±4.8**
	teleport-navigate	**25.1±0.5**	24.5±2.0	21.1±0.5	23.3±0.4	23.3±0.8	**25.1±2.2**
	medium-stitch	0.0±0.0	76.4±4.7	77.5±1.2	79.7±0.2	76.1±0.5	**96.8±0.9**
	large-stitch	0.0±0.0	16.0±2.1	67.1±3.3	79.1±0.9	58.8±0.9	**91.5±1.0**
	giant-stitch	0.0±0.0	9.5±4.3	28.5±1.1	39.7±2.0	23.1±2.7	**86.0±2.1**
	teleport-stitch	21.1±1.0	17.1±1.8	19.3±1.9	21.1±1.3	20.1±1.7	**58.3±1.0**
	medium-explore	0.0±0.0	69.1±7.8	85.7±1.0	83.1±1.6	73.3±0.8	**98.5±0.7**
	large-explore	0.0±0.0	7.5±3.9	42.8±0.6	70.5±0.4	25.7±1.9	**89.2±1.4**
HumanoidMaze	medium-navigate	0.4±0.0	44.5±3.8	47.5±1.6	47.1±0.4	39.7±1.6	**68.1±1.6**
	large-navigate	0.0±0.0	36.7±5.8	40.0±3.8	38.5±2.7	40.4±3.3	**60.7±1.0**
	giant-navigate	0.0±0.0	8.7±2.7	9.5±3.3	15.5±5.2	0.0±0.0	**30.4±1.5**
	medium-stitch	0.1±0.2	51.6±4.2	65.9±0.5	68.0±0.6	61.1±3.9	**86.9±2.0**
	large-stitch	0.0±0.0	29.3±1.5	35.6±2.6	39.6±1.8	42.1±3.0	**64.8±1.8**
	giant-stitch	0.0±0.0	8.0±1.8	6.9±3.9	8.8±3.4	0.1±0.2	**32.1±0.8**

Where a body has to follow the plan, the kernel diffuses over where the data went, so climbing it favors corridors the behavior used, among which a shortest path is indifferent. Charging a planner

$$c(x) \propto n(x)^{-\alpha}, \qquad \alpha = \tfrac{1}{2}, \tag{6}$$

in the visit count $n(x)$ of the cell entered states that preference in one line. Over all 24 datasets the charge is worth +8.4 to A* and +24.3 to FMM, +30.7 on the embodied sixteen; measured against the map, cov-A* leads it by 6.0 and FMM-cov by 19.3, where plain FMM *loses* by 5.0. Added to our own ascent it is worth +5.1 on the embodied sixteen and +2.9 over all 24, the difference being the teleport family, where it *costs* the ascent 4.4 points and takes `pointmaze-teleport-stitch` from 52.9 to 32.0: both mouths of a teleporter are heavily visited, so the bonus pulls a purely local rule to one of them and strands it, the lattice kernel having no edge across. A search that reaches a mouth still has the graph to fall back on, so the same charge is worth +8.2 to A* and +23.1 to FMM there: the signal helps a global planner and hurts a local one.

Most of that is geometry, not knowledge. A high-count cell is usually one in the middle of a corridor, so Equation (6) may be a proxy for *support clearance*, the distance to the nearest cell the data never entered; that is the boundary of the recovered support and not of the true obstacles, and it helps a wide body avoid walls while saying nothing about what the map knows. Two controls test this (Figure A5 and Table A10). Replacing the count by support clearance recovers +23.2 of the +30.7; shuffling the counts across occupied cells retains +7.7, which we attribute to the field being non-uniform at all; and the counts keep their edge where the proxy breaks, on `antmaze-teleport-stitch` (58.3 against 21.7) and the four `humanoidmaze` datasets outside the `large` pair. The routes themselves say the same without any bonus (Section N). That edge is not something the map possesses, since a uniform 3×3 walk on $\mathcal{F}$ cannot tell a cell visited ten times from one visited ten thousand. The reading is therefore in two layers: most of what the charge is worth to search is reproduced by the geometry of the recovered support, which the kernel encodes, and on top of it a visit-density signal the kernel discards, which improves every read-out we tested and helps most where support geometry is least informative.

The strongest planner here is therefore not the learned map. FMM-cov leads Preplay on 21 of 24 datasets by 19.3 points and the OGBench reference baselines on 12 of 16 embodied ones by 20.6, concentrated where trajectories must be composed (86.0 against 2 on `antmaze-giant-stitch`,

89.2 against 10 on `antmaze-large-explore`). We report $\alpha = \frac{1}{2}$ because it was the first value we tried; a sweep on four datasets prefers $\alpha = 1$ by 3.6 points on average but resolves no single gap, so the charge is robust to its exponent over a factor of four (Table A11). We give the planner as a product of this analysis and not a rival: it exists only because we asked what the map preferred, and it needs the coordinates and counts a griddable state space supplies and the fitted map does not. Two qualifications bound the margin. Graph-assisted stitching (Baek et al., 2025) reports 88.3 and 94.2 on those two datasets, ahead of FMM-cov, so we make no claim to the state of the art; the comparison that carries our argument is inside Table 2. And 13.3 points separate cov-A* from FMM-cov under the same charge, a gap that we leave open (Section M).

None of this is about our planner. Offline planning on a recovered support is not a shortest-distance problem: plain FMM, optimizing distance on that graph, *loses* to a map read by ascent, while the same FMM told to prefer traversable cells beats everything here. A planner in this setting should optimize executability, and the data supply two handles on it, support geometry, which the diffusion kernel encodes, and visit density, which it discards and which is worth adding explicitly.

4.5 A Function, Not a Table

Section 4.4 leaves the map no accuracy advantage over the table it distills or the planner distilled from it; its one remaining advantage is that it is a function (Figure A6). Cells hidden from the kernel and from fitting, then restored at evaluation, are answered where a stored graph has no entry at all, and better the longer the horizon: at $\tau = 256$ the rank correlation on held-out cells is 0.89–0.99 against 0.90–1.00 on cells the network trained on, at $\tau = 2$ it is 0.56–0.71 against 0.76–0.85 (Section O). The map interpolates the smooth long-horizon kernels and is markedly worse at the few-cell short ones the ascent terminates on, and its planning lead over search on the same punctured graph, where neither arm enters a hidden cell, is small. A map fitted entirely on ant trajectories drives the humanoid to 36.7 against 40.0 with its own, so the kernel is a statistic of the place and not a portable recipe (Table A12); on pixels the routes measure as routes with no coordinates anywhere, while closed-loop success is near zero (Sections Q and R).

4.6 What Does Not Work

Four changes that follow naturally from the construction were implemented and all four failed, and one of them is load-bearing. Estimating the one-step kernel from the transitions the data performed, rather than from a lattice walk, drops the assumption that adjacency implies reachability and builds the better graph, fully connected on `pointmaze-teleport` where the lattice walk isolates eight cells, and then routes on 0% of episodes. That is not a failure of fitting but of what can be fitted: the estimated P_1 is asymmetric and the symmetric parts of its normalized powers are not positive semi-definite (PSD), carrying 40–42% of their spectral mass on negative eigenvalues at $\tau = 2048$ against 0.00% for the lattice, and $\langle h, h \rangle$ with $h \geqslant 0$ can represent nothing but a PSD kernel. This directed target lies outside that class, which is why the adjacency assumption cannot simply be dropped, why the teleport family stays unsolved, and why the remedy would have to be a symmetrized or directed embedding. The other three changes cost 42, 46 and nearly everything (Section S).

5 Conclusion

A predictive map that reproduces hippocampal place fields can be fitted to offline trajectories and planned on by ascending it, with no search at planning time, and two findings carry beyond this construction. Fitting breaks the read-out and not the fit: the fitted field prefers the same neighbor as the exact one on 99.2% of the states that have one, yet every one of the 5061 greedy failures we logged is a limit cycle, almost all of length two, which one stochastic alternative escapes. And what the ascent contributes over search on the same nodes is executability, not optimality: written as a cost, it makes search the strongest planner here, and support geometry alone, which the kernel encodes, recovers three quarters of that gain. Three things bound the claim: the humanoid is controller-limited, on pixels we can plan on the kernel but not drive a body along its routes, and the kernel estimated for a teleporter is directed, which no symmetric embedding holds (Section 4.6).

AI USE STATEMENT

Large language models (LLMs) were used during this project as coding and writing assistants, including to help draft and refactor experiment and analysis scripts, revise prose, and check the consistency of the manuscript with its tables and figures. The empirical results reported in this paper are based on experiment runs conducted and reviewed by the authors, with tables and figures generated primarily from recorded experimental outputs. Citations and LLM-assisted content were reviewed by the authors. The research questions, experimental design, evaluation choices, and interpretation of the results were determined by the authors, who take responsibility for the final content of the paper.

ETHICS STATEMENT

This work uses only the navigation datasets of OGBench (Park et al., 2025), a public benchmark of simulated offline data released under the MIT license, through its registered environments and without modification. It involves no human subjects, no animal experiments and no personal data, and the place fields it reports belong to a fitted network, not to recorded neurons. The method plans routes for simulated bodies in mazes. Deployed on a physical system, it would inherit the preferences of its data: the visit-count charge of Equation (6), which improves every planner we tested, works by steering the planner along the corridors the behavior data used, so data that contains unsafe behavior yields a planner that prefers it, and nothing in the method certifies a route as safe. We depart from the reference evaluation protocol only where Section A says so, quote the baseline numbers as published, and are aware of no conflict of interest arising from this work.

REPRODUCIBILITY STATEMENT

All source code will be released publicly upon acceptance: the fitting of the map, Preplay and every other planner the paper compares, the embodied evaluation harness, the training of the low-level controller, the scripts that turn the logged runs into the numbers in the paper, and, for each figure and table, the data it is drawn from and the script that draws it. The logged records of the reported runs and the submission scripts, which carry the arguments of every run, are released with it, and those scripts rebuild the fitted maps and the low-level controllers at the cost given in Section D. The paper itself states what a reimplementation requires. The environments, datasets and success criterion are those of the public OGBench release, used unmodified, and Section A audits our protocol against its source; the reference kernel, the embedding and the fitting objective are given in closed form in Equations (1) to (3), and the planner in Equation (5); Table A2 lists every setting and Section C how each was chosen; and our HIQL runs in Figure A4 use the reference implementation of Park et al. (2025) with the per-dataset settings of its Table 10. Section T states what reproduces exactly: a rerun reproduces a cell bit for bit on the device that produced it, while moving the low-level controller between GPU and CPU moves a cell by up to 6.8 points.

REFERENCES

Anurag Ajay, Yilun Du, Abhi Gupta, Joshua B. Tenenbaum, Tommi S. Jaakkola, and Pulkit Agrawal. Is conditional generative modeling all you need for decision making? In *Proceedings of International Conference on Learning Representations (ICLR)*, 2023. 3

Seungho Baek, Taegeon Park, Jongchan Park, Seungjun Oh, and Yusung Kim. Graph-assisted stitching for offline hierarchical reinforcement learning. In *Proceedings of International Conference on Machine Learning (ICML)*, 2025. 3, 9

Timothy E. J. Behrens, Timothy H. Muller, James C. R. Whittington, Shirley Mark, Alon B. Baram, Kimberly L. Stachenfeld, and Zeb Kurth-Nelson. What is a cognitive map? Organizing knowledge for flexible behavior. *Neuron*, 100(2):490–509, 2018. 1

Wilka Carvalho, Sam Hall-McMaster, Honglak Lee, and Samuel J. Gershman. Preemptive solving of future problems: Multitask preplay in humans and machines. arXiv:2507.05561, 2025. 2, 3

Peter Dayan. Improving generalization for temporal difference learning: The successor representation. *Neural Computation*, 5(4):613–624, 1993. 1, 3

George Dragoi and Susumu Tonegawa. Preplay of future place cell sequences by hippocampal cellular assemblies. *Nature*, 469(7330):397–401, 2011. 2

Scott Emmons, Ajay Jain, Michael Laskin, Thanard Kurutach, Pieter Abbeel, and Deepak Pathak. Sparse graphical memory for robust planning. In *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*, 2020. 3

Benjamin Eysenbach, Ruslan Salakhutdinov, and Sergey Levine. Search on the replay buffer: Bridging planning and reinforcement learning. In *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*, 2019. 3

Peter E. Hart, Nils J. Nilsson, and Bertram Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2):100–107, 1968. 2, 3

Michael Janner, Yilun Du, Joshua B. Tenenbaum, and Sergey Levine. Planning with diffusion for flexible behavior synthesis. In *Proceedings of International Conference on Machine Learning (ICML)*, 2022. 3

Adam Johnson and A. David Redish. Neural ensembles in CA3 transiently encode paths forward of the animal at a decision point. *Journal of Neuroscience*, 27(45):12176–12189, 2007. 2

Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative Q-learning for offline reinforcement learning. In *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*, 2020. 3

Hui Lin, Yukun Yang, Rong Zhao, Giovanni Pezzulo, and Wolfgang Maass. Neural sampling from cognitive maps enables goal-directed imagination and planning. *Nature Machine Intelligence*, 8 (7):1045–1065, 2026. 1, 3, 4

John O'Keefe and Jonathan Dostrovsky. The hippocampus as a spatial map: Preliminary evidence from unit activity in the freely-moving rat. *Brain Research*, 34(1):171–175, 1971. 1

Seohong Park, Dibya Ghosh, Benjamin Eysenbach, and Sergey Levine. HIQL: Offline goal-conditioned RL with latent states as actions. In *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*, 2023. 3

Seohong Park, Kevin Frans, Benjamin Eysenbach, and Sergey Levine. OGBench: Benchmarking offline goal-conditioned RL. In *Proceedings of International Conference on Learning Representations (ICLR)*, 2025. 1, 2, 3, 5, 7, 10, A1, A14, A16

Nikolay Savinov, Alexey Dosovitskiy, and Vladlen Koltun. Semi-parametric topological memory for navigation. In *Proceedings of International Conference on Learning Representations (ICLR)*, 2018. 3

James A. Sethian. A fast marching level set method for monotonically advancing fronts. *Proceedings of the National Academy of Sciences (PNAS)*, 93(4):1591–1595, 1996. 1, 2, 3, A9

Kimberly L. Stachenfeld, Matthew M. Botvinick, and Samuel J. Gershman. The hippocampus as a predictive map. *Nature Neuroscience*, 20(11):1643–1653, 2017. 1, 3

Edward C. Tolman. Cognitive maps in rats and men. *Psychological Review*, 55(4):189–208, 1948. 1

Tongzhou Wang, Antonio Torralba, Phillip Isola, and Amy Zhang. Optimal goal-reaching reinforcement learning via quasimetric learning. In *Proceedings of International Conference on Machine Learning (ICML)*, 2023. 3

Dehong Xu, Ruiqi Gao, Wen-Hao Zhang, Xue-Xin Wei, and Ying Nian Wu. On conformal isometry of grid cells: Learning distance-preserving position embedding. In *Proceedings of International Conference on Learning Representations (ICLR)*, 2025. A14

Minglu Zhao, Dehong Xu, Deqian Kong, Wen-Hao Zhang, and Ying Nian Wu. Place cells as multi-scale position embeddings: Random walk transition kernels for path planning. In *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*, 2025. 1, 3, 4, 5, A3, A6, A7, A9, A15

A Protocol Audit Against OGBench

Every number we report is compared against numbers produced by someone else, so we went through the OGBench source and paper axis by axis and record here what matched, what did not, and what we did about it. Two of the mismatches flattered us and one handicapped us, which is why we audited in both directions.

Points of agreement. The success flag is the one the environment computes: we read `info['success']` and add no distance test of our own, so we inherit whatever tolerance and timing the environment applies (1.0 for the point embodiment, 0.5 for the ant and the humanoid). Because the registered environments set `terminate_at_goal=True`, ending an episode at the first successful step is equivalent to reading the flag from the final transition, which is what the reference harness does. Actions at evaluation are deterministic on both sides: the reference implementations evaluate at `eval_temperature=0` with no action noise, and our low-level controller emits a $\tanh$ mean with no sampling. Both use the five test-time goals the benchmark defines.

The episode budget is not uniform. OGBench registers 1000 steps for the point and ant embodiments, 2000 for the medium and large humanoid mazes and 4000 for the giant one, and we read the budget from the environment specification rather than fixing it. A cap of 1000 everywhere would evaluate the humanoid at between a half and a quarter of the budget the published baselines were given; under that cap the ascent routes on every episode and the episodes end at the cap with the body short of the goal.

Evaluation sample size. The reference protocol evaluates each test-time goal with 50 rollouts and averages over three evaluation epochs, so each published cell is an average over 750 rollouts. We evaluate 50 rollouts per goal over a single trained map, giving 250 per map and 750 per cell across our three maps. We do not average over training checkpoints: that device smooths the run-to-run variation of a reinforcement learning objective, and the map is fitted by regression, whose end point does not fluctuate in the same way. We seed the environment reset, which the reference harness does not; this makes our cells exactly reproducible at the cost of estimating the average over a fixed set of initial states rather than over the initial-state distribution.

Baseline hyperparameters are not the implementation defaults. The OGBench reference implementations ship the *navigate* settings as their defaults, and `main.py` does not vary them by dataset type, but Table 10 of Park et al. (2025) does: the policy goal-sampling ratio is $(0, \frac{1}{2}, 0, \frac{1}{2})$ on stitch and $(0, 1, 0, 0)$ on navigate, and the discount is 0.995 on the giant mazes against 0.99 elsewhere. Running the reference implementation as shipped therefore reproduces the published number on navigate but not on stitch. It gives 19.6 at full data against the published 74, while with the per-dataset settings of OGBench Table 10 the same code reaches 75.6–81.6 across seeds. It is the latter we report.

The controller budget. The mismatch running the other way is larger. Every OGBench baseline is trained for 10^6 gradient steps, whereas a controller trained for 80 epochs at batch 8192 sees roughly 8800 on a million-transition dataset, a factor of 114 fewer. Since the navigate datasets are precisely the ones where the ascent already routes on essentially every episode, the entire residual there is execution, and it was being produced by a controller given a hundredth of the training its competitors received. Section B sweeps the budget; the reported results use 1200 epochs.

B Low-Level Controller Budget

Table A1 gives the sweep, against the 10^6 gradient steps every OGBench baseline receives. The two embodiments answer differently, and the difference decided how we configured the final runs. The ant is budget-limited: 1200 epochs is worth 14 points over 80, and 4000 overshoots it. The humanoid is not, at either budget or width. The 80, 1200 and 1200-at-double-width settings give 46.0, 46.0 and 52.0, a spread inside the sampling error of the sweep itself and no larger than the spread over maps. We nevertheless train every controller in the reported results to the same 1200 epochs on every suite. Setting the budget once and reporting that one embodiment ignores it is a result; setting it per suite after seeing which suite it helped would be a configuration chosen on the outcome.

Table A1: **Success against the training budget of the low-level controller.** Measured on the two datasets whose ascent routes on every episode, so that any change is the controller. Every OGBench baseline receives 10^6 steps. Each cell is one run of 50 episodes on one map, a fifth of the budget the reported results use, so a binomial standard error of about 7 points bounds what this resolves.

Controller budget	gradient steps	AntMaze medium	HumanoidMaze medium
80 epochs	8800	72.0	46.0
400 epochs	44,000	76.0	42.0
1200 epochs	132,000	86.0	46.0
4000 epochs	440,000	82.0	–
1200 epochs, width 1024	132,000	–	52.0
OGBench baselines	1,000,000		–

Table A2: **Every setting used to produce the reported results.** The configuration is identical on all 24 datasets except for the block at the bottom, which varies by suite or by embodiment and none of which is chosen on outcomes.

	Setting	Value
Reference kernel	grid	occupied cells of a $G \times G$ discretization
	one-step walk	symmetric 3×3, obstacles absorbing
	scales τ	$2, 4, \ldots, 2048$ (eleven, by repeated squaring)
	normalization	$q = p/\sqrt{p_{xx}p_{yy}}$
Map	embedding dimension	500
	trunk	six residual layers, width 512
	input features	Fourier, $L = 12$, base frequency 0.05
	pair weight c	0.05
	optimizer	AdamW, 10^{-3}, weight decay 10^{-4}
	schedule	linear decay to $0.1\times$ over training
	iterations	32,000 full-batch
Planner	temperature T	0.3
	imagined rollouts K	16
	ascent step cap	400
	greedy route tried first	yes
	coverage exponent α	0.5
	coverage weight in **+cov**	0.5, in units of $\max_z \lvert\Delta\rvert$
Controller	objective	action-free IQL, expectile 0.7
	policy extraction	AWR, $\beta = 10$, weights clipped at 100
	discount	0.99
	network	six layers, width 512
	batch	8192
	epochs	1200 on every suite
Varies by suite	grid size G	40; 60 on the larger ant mazes
	relabeling horizon	$[4, 20]$ ant, $[15, 40]$ humanoid
	waypoint advance radius	0.5 world units PointMaze, 1.5 embodied
	waypoint skip after	60 stale steps PointMaze, 80 embodied

C HYPERPARAMETERS

Table A2 lists them, and two deserve a word. The grid is 60 on `antmaze-large` and `antmaze-giant` and 40 everywhere else, including every `humanoidmaze`. That is not a rule about arena size: `humanoidmaze-giant` is a larger arena than `antmaze-large` and uses the smaller grid. It is a per-suite choice inherited from the order in which the experiments were run, never tuned on a return, and Section P reports the control that measures what it costs. The relabeling horizon differs between the ant and the humanoid because their step frequencies differ by roughly a factor of three, and one horizon for both would give the humanoid sub-goals it cannot reach within the window. The two executor thresholds differ only between the massless point and the embodied suites.

How these were chosen. The map hyperparameters, meaning dimension, depth, input features, pair weight and schedule, were selected on a quantity that never touches the environment: the rank correlation between the fitted inner products and the reference kernel on held-out state pairs. This

is why the architecture rows are identical on all 24 datasets. No return, success rate or episode was consulted in setting any of them.

The two planner constants are a different matter and we state their provenance exactly. T and K were swept over $T \in \{0, 0.1, 0.3, 0.6, 1.0\}$ and $K \in \{1, 2, 4, 8, 16, 32\}$, at three independently trained maps per point, on `pointmaze-giant-navigate` and `pointmaze-teleport-stitch`, the two datasets of Table A7, chosen because they are the ones that retain headroom, and both of which also appear in Table 1. That is a held-in selection and we do not claim otherwise. The damage is limited by the shape of the sweep, not by its protocol: success varies by 3.6 and 0.7 points across the whole of $K \in \{1, \dots, 32\}$, and by 3.5 and 3.4 points across $T \in \{0.1, \dots, 1.0\}$, against the 9.2 and 24.5 points that separate $T = 0.3$ from the greedy of Zhao et al. (2025), or 5.2 and 1.7 from the $T \to 0$ limit of the rule. The result does not rest on the value of either constant, and a reader who prefers $T = 1.0$ and $K = 1$ chosen blind loses 3.2 points (Section J).

We ran no per-dataset sweep of any kind, and the cost of not doing so is visible in Table 1: several datasets would plainly be improved by a temperature of their own, and none has one. Of the two quantities that do vary by suite, the relabeling horizon is set by the step frequency of the body, which can be read off before evaluating anything, and the grid is a per-suite choice never tuned on a return.

D Compute

All experiments ran on a single H100 per job. Fitting one map takes about ten minutes at 32,000 full-batch iterations on a grid with roughly a thousand occupied cells; the 24 datasets at three maps each are therefore around twelve GPU-hours. Training one low-level controller at 1200 epochs takes about ninety minutes, and there are sixteen of them. Evaluation dominates: 250 episodes on `humanoidmaze-giant` at a 4000-step budget is a million simulator steps per map, and the full table at three maps per cell is the largest single cost in the paper. Planning itself is negligible. An ascent step is a few hundred inner products against the embeddings of the neighboring cells, and a whole route costs less than a millisecond, which is the practical reason a planner that does not search can afford to be re-run whenever the body has drifted.

The baseline reproductions used for Figure A4 are the expensive part per run: the official implementation at 10^6 gradient steps takes two to five hours per seed on this hardware, and the data-efficiency curve is three datasets by five fractions by three seeds.

E How Large a Difference Has to Be to Count

Four quantities move a cell of Table 1 while the method stays fixed: which map was fitted, which block of reset seeds was drawn, which device the controller ran on, and which controller was trained. We measure each.

Maps and reset seeds. The spreads in Table 1 are over maps at one fixed set of 250 initial states. Holding the map fixed and changing the reset seeds instead moves the score by 0.8, 2.5 and 3.6 points on the three datasets we checked, comparable to the spread over maps and larger on two of them.

The hardware alone. Every spread above is over something we chose; here we measure one we did not. We reran the FMM arm on all sixteen embodied datasets with the low-level policy on CPU instead of GPU, holding the reset seeds, the recovered free set, the planner, the controller weights, the executor, the waypoint spacing, the episode budget and the success flag identical. The planner is `numpy` in both arms and never reads the map, so nothing separates the two runs but the rounding of the policy forward pass. The control that licenses reading the gap as a device effect is the A-vs-A arm: a second CPU run of two of the datasets returns 81.6 and 58.0, bit-identical to the first, so the pipeline is deterministic given a device and a seed. Table A3 gives the sixteen pairs. The mean absolute difference is 2.7 points and the largest is 6.8; the sign is up on ten datasets, down on five and unchanged on one, so this is dispersion and not a bias of one device against the other, and the two datasets that barely move (`humanoid-giant-navigate`, `humanoid-giant-stitch`) are the two with almost no headroom. What produces a 6.8-point swing out of the last bits of a matrix multiply is the executor. A waypoint-following controller in contact-rich dynamics is a chaotic map: two trajectories that differ in the twelfth decimal of one action diverge within a few hundred steps,

Table A3: **The same evaluation on two devices.** FMM on the recovered support, success (%) over 250 episodes, with the low-level controller on GPU (the runs the tables report) and on CPU. Everything else is held fixed, including the reset seeds, so the difference is floating-point rounding amplified by the executor, and it reaches 6.8 points.

Dataset	GPU	CPU	difference
`ant-giant-navigate`	29.6	32.8	+3.2
`ant-giant-stitch`	19.2	22.4	+3.2
`ant-large-explore`	24.4	21.2	-3.2
`ant-large-navigate`	68.8	69.6	+0.8
`ant-large-stitch`	57.6	63.6	+6.0
`ant-medium-explore`	74.4	78.0	+3.6
`ant-medium-navigate`	80.8	81.6	+0.8
`ant-medium-stitch`	75.6	77.6	+2.0
`ant-teleport-navigate`	23.2	23.6	+0.4
`ant-teleport-stitch`	18.4	22.0	+3.6
`humanoid-giant-navigate`	0.0	0.0	+0.0
`humanoid-giant-stitch`	0.4	0.0	-0.4
`humanoid-large-navigate`	44.8	38.0	-6.8
`humanoid-large-stitch`	38.0	39.6	+1.6
`humanoid-medium-navigate`	40.0	37.6	-2.4
`humanoid-medium-stitch`	63.6	58.0	-5.6

Table A4: **Three controllers per humanoid dataset on one map.** Success (%) over 250 episodes with seeds 1234, 7 and 21, every training argument but the seed held equal, and the map fixed at the first of the three we fitted. The spread over controllers is smaller than the spread over maps on the same datasets, given alongside.

Dataset	1234	7	21	sd	sd over maps
`humanoid-medium-navigate`	48.8	44.4	50.0	2.41	1.6
`humanoid-medium-stitch`	66.4	68.8	65.6	1.36	0.5
`humanoid-large-navigate`	34.8	38.0	36.4	1.31	3.8
`humanoid-large-stitch`	35.6	34.8	33.6	0.82	2.6
`humanoid-giant-navigate`	5.6	8.0	8.8	1.36	3.3
`humanoid-giant-stitch`	7.2	8.4	4.0	1.86	3.9

and an episode that was going to end just inside the success radius ends just outside it. This is a property of the benchmark and not of anything we propose. PointMaze is not covered by the measurement, since its controller is a proportional rule in `numpy` and its search arms never touch a network; its spread over maps is reported throughout and is the relevant one there.

Retraining the controller. Every spread reported elsewhere holds the controller fixed, which on the humanoid is where a reader should push: the ascent routes on every episode there and the episodes are still lost at the budget, so the residual is execution and the controller is the component it belongs to. We trained two further controllers on each of the six humanoid datasets, seeds 7 and 21 against the 1234 every reported number uses, with every other argument identical ($k \in [15, 40]$, width 512, six layers, 1200 epochs, batch 8192), and evaluated all three against one fixed map so that the controller is the only thing that varies. Table A4 gives the eighteen cells. The spread over controllers is 0.8–2.4 points with a mean of 1.5, against 0.5–3.9 and a mean of 2.6 over maps on the same six datasets, and below what moving the same controller between devices does there (0.0–6.8, mean 2.8). Retraining the controller therefore moves the number by less than refitting the map does, and the concern that the humanoid results rest on one lucky controller does not survive the measurement. The seed-1234 column reproduces the corresponding reported run bit for bit on all six datasets, so the rest of the row is the controller and not a configuration difference.

What a difference of a few points means. None of the four sources is negligible, and their sum is why we read a few points on one embodied dataset as a tie. Averaged over a suite they are not: the standard deviation of the sixteen device differences is 3.5, so the standard error of a sixteen-dataset mean under that perturbation is 0.9, and the margins the paper draws conclusions from are an order of magnitude above it.

Table A5: **Rank correlation between the fitted and the exact kernel per scale.** Computed over all pairs of free cells on the three maps behind Table 1. Unlike every other table in this paper these are single maps, not three: the quantity is a property of one fitted network against the kernel it was fitted to, a structural check like the place-field statistics of Section 4.1, so it carries no episode sampling and no spread. The fit is exact at long horizons and worst in the middle, which is why the diagnosis in Section 4.2 uses the local preferred neighbor rather than this statistic.

τ	large-navigate	medium-stitch	giant-navigate
2	0.788	0.757	0.826
4	0.881	0.886	0.887
8	0.893	0.890	0.876
16	0.797	0.826	0.778
32	0.773	0.826	0.702
64	0.864	0.932	0.797
128	0.960	0.988	0.908
256	0.995	1.000	0.981
512	1.000	1.000	0.999
1024	1.000	1.000	1.000
2048	1.000	1.000	1.000

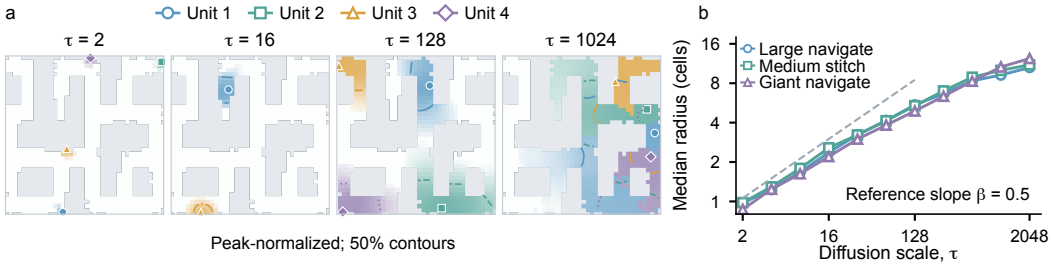


Figure A1: **Place fields of the fitted map.** (a) Four units at four of eleven scales, peak-normalized with the 50% contour drawn. (b) Median field radius against scale on three maps, with a reference slope of $\frac{1}{2}$. The exponent is 0.36–0.39 across all eleven scales: the curves are steepest below $\tau = 32$ and flatten from $\tau = 512$, where the fields reach the walls.

F How Well the Maps Are Fitted

Section 4.1 claims that the read-out fails on a map fitted well enough for the failure not to be a fitting error. For each of the three maps behind Table 1 we take the Spearman correlation between the fitted inner product $\langle h(x,\tau), h(y,\tau)\rangle$ and the exact normalized kernel $q(y \mid x,\tau)$ over every pair of free cells, at every scale (Table A5). The fit is essentially exact at the long horizons, 0.98 and above from $\tau = 256$ and indistinguishable from 1 by $\tau = 1024$, and distinctly worse in the middle, bottoming at 0.702 around $\tau = 32$. That shape is the opposite of convenient for us: the scales the ascent terminates on are the short ones, and they are not the ones fitted best. It is why the argument in Section 4.2 rests on the preferred-neighbor agreement rather than on this table. A global rank correlation says how well the whole kernel is reproduced, while what an $\arg\max$ over eight neighbors consults is a local ordering, and the two come apart here.

G Place Fields of the Fitted Map

Section 4.1 states the measurement; this appendix gives the definitions behind it. A unit is *active* at a cell where it carries at least a tenth of the largest response at that cell. Its center is the response-weighted mean of its active cells, its radius the median distance from that center, and its *contiguity* the fraction of its active cells lying within two radii of it. The measure is 1 for a single blob and falls as a unit breaks into pieces. The exponent is a least-squares fit of $\log(\text{radius})$ on $\log\tau$ over all eleven scales; restricted to the five at $\tau \leqslant 32$, before the fields reach the walls, it is 0.441, 0.443 and 0.440 on the three maps. One caveat on the word *sparse*. The free sets here hold 866 to 1192 cells, so an embedding of dimension 500 is within a factor of two of the rank a tabular factorization would need; the measurement says that the coordinates in use are spatially contiguous and that their number grows with the horizon, not that the representation is small. We do not test compression; the network buys being defined off the cells it was fitted on (Section O), not being smaller than them.

Table A6: **The exact tabular kernel against the fitted map under both read-outs.** Success (%) over 250 episodes. Exact-kernel greedy is a single deterministic run on a single lattice and is the only cell here without a spread; the other three columns are three replicates each and appear again in Table A9.

	Exact kernel		Fitted map	
Dataset	greedy	sampled	greedy	sampled
`point-large-navigate`	100.0	$100.0_{\pm 0.0}$	$74.1_{\pm 8.5}$	$100.0_{\pm 0.0}$
`point-medium-stitch`	100.0	$100.0_{\pm 0.0}$	$95.2_{\pm 4.8}$	$100.0_{\pm 0.0}$
`point-giant-navigate`	99.6	$99.1_{\pm 0.4}$	$86.0_{\pm 4.8}$	$91.2_{\pm 5.3}$

H The Effect of Approximation on the Field

The read-out fails on the fitted map and not on the kernel it was fitted to, and this appendix is the measurement that says why. Two explanations we expected are wrong, and the third is the one Section 4.2 uses.

Approximation does not flatten the field: the fraction of states at which no neighbor offers a detectable improvement is 0.52–0.63 for the fitted map and the same for the exact kernel, which already has them. Nor does it move the preferred neighbor appreciably: only 71% of states have a strict maximum on the exact field at all, and restricted to those the fitted field agrees on 99.2%. The two figures measure different things, since the ascent takes an $\arg\max$ over Δ whether or not any Δ is positive, so a state can have a strict maximum and still offer no improvement, and it is exactly such states that close a cycle when each of two of them is the $\arg\max$ of the other. One per cent of per-step disagreement cannot produce the failure rates we see, 4.8–48.8% of goals missed by the greedy ascent on the four PointMaze datasets that retain headroom and 6.0–10.5% of the start–goal pairs failed in this diagnostic, averaged over the three maps of each dataset against the 3.7–14.7% per map plotted in Figure 3, unless the failures are absorbing, and they are. Over the 57,549 pairs the diagnostic enumerates on four datasets and three maps each, 5061 deterministic rollouts fail, and every one of them revisits a cell it has already occupied rather than halting at a fixed point, then runs to the step cap: a cycle is what exhausts the budget, not slow progress. The cycles are as small as a cycle can be. 99.98% of them have length two, two cells each of which is the $\arg\max$ of the other, which is exactly the configuration a plateau permits, and the longest we found anywhere has length three. That is why sampling repairs it, and why one stochastic alternative suffices: leaving a cycle needs only difference, not correctness.

Nor are the cycles an artifact of the pair weight we ourselves report as harmful (Section I). Refitting the two ablation datasets at $c = 0$ leaves the failure mode unchanged in kind and smaller in size, 4% and 3% of pairs against 8% and 10%, still with no fixed points, and on `giant-navigate` the $c = 0$ map agrees with the exact field on *fewer* states (96.0% against 99.2%) while cycling less often, a second reason not to read single-step agreement as the operative quantity.

The same read-outs on the exact kernel. Table A6 puts the tabular kernel the network is fitted to beside the network, under both read-outs, on the three PointMaze datasets where the exact kernel is available and the ascent has headroom. Greedy ascent on the exact kernel does not fail. Sampling adds nothing there and recovers most of what fitting costs on the network, which is the comparison Section 4.2 rests on: the same rule, the same episodes, the same controller, and only the object being read changes.

I What Each Part Is Worth

Ablating where the method already scores 100% measures nothing, so we ablate on the two datasets that retain headroom; Table A7 gives every row, and two of them went against what we expected when we built the method.

Two components carry the method. Removing the Fourier features costs 18.4 points on `giant-navigate`, since a network reading raw coordinates cannot produce the few-cell variation of the short-τ kernels the ascent terminates on; removing the rule that tries the greedy route first costs 12.5 there and 22.0 on `teleport-stitch`. The $T = 0$ row is the greedy of Zhao et al. (2025), which $T = 0.3$ beats by 9.2 and 24.5 points; against the $T \to 0$ limit of the rule itself, the greedy column of Table 2, the margin is 5.2 and 1.7, and everything from 0.3 to 1.0 is within two points of everything else.

Table A7: **What each part of the method is worth.** Measured on the two PointMaze datasets that retain headroom under the final configuration; success rate (%), three independently trained maps at 250 evaluation episodes each. The input features and the greedy-route rule carry the method; the pair weight in the training objective does not, and removing it would improve both columns.

	PointMaze	
Configuration	giant-navigate	teleport-stitch
Preplay	**91.2**$_{\pm 5.3}$	**52.9**$_{\pm 3.2}$
without the pair weight of Equation (3)	95.7$_{\pm 6.0}$	56.1$_{\pm 0.8}$
without Fourier input features	72.8$_{\pm 3.6}$	44.7$_{\pm 2.9}$
without the greedy route tried first	78.7$_{\pm 2.7}$	30.9$_{\pm 0.2}$
temperature $T = 0$ (greedy of Zhao et al.)	82.0$_{\pm 2.0}$	28.4$_{\pm 1.8}$
$T = 0.1$	87.7$_{\pm 6.4}$	49.5$_{\pm 3.5}$
$T = 0.3$ (used)	91.2$_{\pm 5.3}$	52.9$_{\pm 3.2}$
$T = 0.6$	91.1$_{\pm 4.7}$	52.5$_{\pm 3.0}$
$T = 1.0$	90.3$_{\pm 4.1}$	51.2$_{\pm 4.0}$
one sampled rollout, $K = 1$	88.3$_{\pm 6.7}$	52.4$_{\pm 2.9}$
$K = 16$ (used)	91.2$_{\pm 5.3}$	52.9$_{\pm 3.2}$
$K = 32$	91.9$_{\pm 4.9}$	53.1$_{\pm 3.3}$

Two rows correct our own design. The pair weight of Equation (3) does not earn its place on these two datasets, since removing it improves them by 4.5 and 3.2 points. Acting on that means refitting every map in the paper and re-running every arm that reads one, so what we can say is that the term is not load-bearing and that the reported numbers do not depend on it: Section H shows the limit cycles, which are the mechanism the paper turns on, unchanged in kind at $c = 0$. And the number of imagined rollouts, which the description of the method makes central, is nearly irrelevant *here*: $K = 1$ is within 3.6 and 0.7 points of $K = 32$ on these two datasets. Both are PointMaze and both have short routes, and we did not sweep K on the embodied suite, so this does not license dropping it: Section J shows the blind $T = 1$, $K = 1$ configuration losing 4.5 points on average over the large and giant mazes against 2.1 over the medium ones, and its single worst case (-13.0) is a medium one, so we can say the constants are not free without being able to say which of the two is doing the work. The method needs the greedy route, one stochastic alternative when it stalls, and a temperature that is not too small.

J The Untuned Planner Configuration

T and K were swept on `pointmaze-giant-navigate` and `pointmaze-teleport-stitch`, both of which appear in Table 1. That is held-in selection, so we give the configuration a reader would pick without measuring anything: $T = 1$, the unit softmax temperature, and $K = 1$, a single imagined rollout. It costs 3.2 points on average and two of the thirteen bold cells. Both columns are three independently fitted maps at 250 episodes, and the spread of the untuned column runs 0.0–9.6, so most of the per-dataset differences in Table A8 are inside it, and the average carries the comparison.

K Where the Failures of Each Dataset Come From

Every episode logs whether the ascent routed as well as whether it succeeded, so failures split exactly into episodes without a route and routes that were not executed (Figure A2). On the humanoid mazes the ascent reaches the goal on *every* evaluation episode, so what fails there is the execution of the route; the episodes end at the budget with the body short of the target, consuming 76%, 92% and 100% of the steps on medium, large and giant. On the ant mazes it depends on size: at medium the ascent also routes on every episode and the residual 18 points are execution, while on giant the ascent still routes on 91–93% and the rest is planner failure. The teleport mazes fail a third way. The lattice walk keeps the two ends of a teleporter maximally far apart, so the map lacks the shortcut the task is built around. Figure A3 shows one episode of each of the three kinds, drawn from the logged world coordinates of the planned route and of the body that tried to follow it.

Table A8: **The reported planner constants against an untuned choice.** Success (%) on all 24 datasets at $T = 0.3$, $K = 16$ and at $T = 1$, $K = 1$; three independently fitted maps at 250 episodes each. The untuned setting costs 3.2 points on average.

Dataset	T=0.3, K=16	T=1, K=1	Δ
ant-giant-navigate	23.3±5.5	11.5±4.6	-11.8
ant-giant-stitch	28.5±1.1	17.1±3.0	-11.4
ant-large-explore	42.8±0.6	51.1±2.3	+8.3
ant-large-navigate	71.2±3.6	63.9±6.5	-7.3
ant-large-stitch	67.1±3.3	63.5±1.5	-3.6
ant-medium-explore	85.7±1.0	86.5±1.3	+0.8
ant-medium-navigate	85.2±1.4	84.1±1.0	-1.1
ant-medium-stitch	77.5±1.2	78.7±1.6	+1.2
ant-teleport-navigate	21.1±0.5	20.4±0.7	-0.7
ant-teleport-stitch	19.3±1.9	20.4±0.3	+1.1
humanoid-giant-navigate	9.5±3.3	7.6±4.4	-1.9
humanoid-giant-stitch	6.9±3.9	5.9±3.4	-1.0
humanoid-large-navigate	40.0±3.8	35.5±6.4	-4.5
humanoid-large-stitch	35.6±2.6	25.6±1.8	-10.0
humanoid-medium-navigate	47.5±1.6	44.7±2.7	-2.8
humanoid-medium-stitch	65.9±0.5	52.9±2.7	-13.0
point-giant-navigate	91.2±5.3	86.8±7.2	-4.4
point-giant-stitch	99.9±0.2	88.4±9.6	-11.5
point-large-navigate	100.0±0.0	100.0±0.0	+0.0
point-large-stitch	99.7±0.4	99.9±0.2	+0.2
point-medium-navigate	100.0±0.0	100.0±0.0	+0.0
point-medium-stitch	100.0±0.0	100.0±0.0	+0.0
point-teleport-navigate	30.9±0.2	28.5±0.7	-2.4
point-teleport-stitch	52.9±3.2	51.5±4.2	-1.4

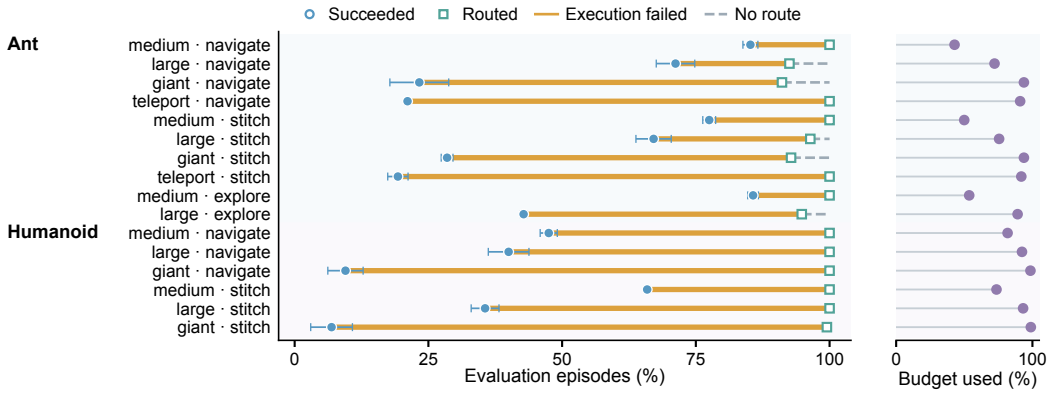


Figure A2: **Where each embodied dataset loses its episodes.** The fraction of episodes on which the ascent routed (square) and on which the body reached the goal (circle, s.d. over three maps); the solid bar between them is what the controller loses and the dashed tail what the planner loses. Every humanoid dataset routes on 99.5–100%. Right: fraction of the budget consumed.

L Data Efficiency in Full

A map and a policy are estimated from different things: the kernel of Equation (1) is a statistic of *where the data went*, a value function of *what the data did*, and support saturates long before behavior does. We test this on PointMaze, whose controller is unlearned, so removing data removes what the *planner* is given and nothing else; whole episodes are dropped at $1\%, 3\%, 10\%, 30\%, 100\%$.

The separation in Figure A4 is not a matter of degree. At 1% of the episodes Preplay holds 100.0 ± 0.0, 98.7 ± 1.9 and 86.5 ± 18.8 on `large-navigate`, `medium-stitch` and `giant-navigate`, where HIQL returns 2.1, 0.4 and 2.4; the ascent still routes on essentially every episode. The baseline uses the per-dataset hyperparameters of OGBench Table 10 and reproduces the published full-data values on `medium-stitch` and `giant-navigate` (78.3 and 45.9 against 74 and 46) but falls 10.3 points short on `large-navigate` (47.7 against 58), which cannot account for a collapse to 0.4–2.4 at 1%.

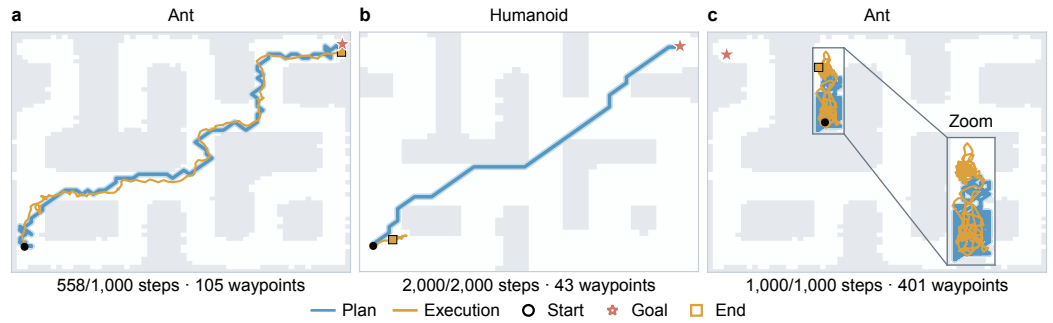


Figure A3: **Three ways an episode ends.** Drawn from the logged world coordinates. (a) Plan and body coincide and the goal is reached. (b) The route is complete and short and the humanoid runs out of budget. (c) The ascent never routed: its 401 waypoints double back within a small region. The panels are re-rendered rather than taken from the evaluation log, since the harness advances one planner RNG across a run while the renderer seeds each episode on its own.

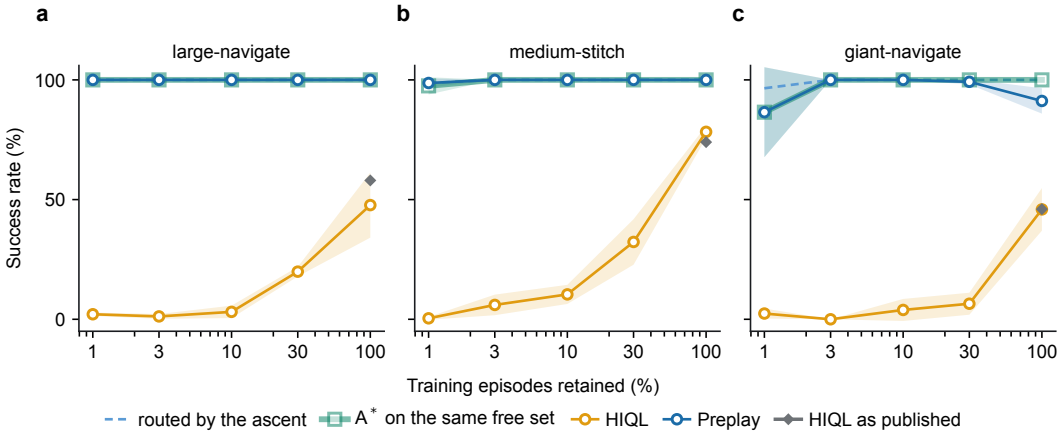


Figure A4: **Data efficiency on PointMaze.** Success against the fraction of training episodes retained; the controller is unlearned, so the reduction affects only the planner. Preplay and A* on the same free set agree to within 1.2 points at every fraction but full-data `giant-navigate` (91.2 against 100.0), while HIQL collapses; the dashed line is the fraction on which the ascent routed at all. Shading is the s.d. over three maps, and over three seeds for HIQL.

The limits of the contrast. The contrast with HIQL is real; the credit does not belong to the fitted embedding. Over the same fractions, search on the same free set traces the same curve: 100.0 throughout on `large-navigate`, 97.5 ± 3.6 against our 98.7 at 1% on `medium-stitch`, and 86.5 ± 18.8 on `giant-navigate`, our value to the decimal, because the two fail on the same episodes. The quantity that saturates early is the *support*, and any planner reading it inherits the property.

M Every Planner Arm on Every Dataset

Table 2 shows six of the nine read-outs on the sixteen embodied datasets. The three omitted there are the tabular kernel the network is fitted to, plain A*, and the same A* under the coverage charge; all three are discussed in Section 4.4 and appear in Table A9 with the rest, over all 24 datasets. Each cell is three replicates: three independently fitted maps for the arms that read one (**greedy**, **Preplay**, **+cov**), and three disjoint blocks of 50 reset seeds (0–49, 50–99, 100–149), 250 episodes each, for the arms that do not, which are deterministic once the free set and the seeds are fixed. **greedy** is the $T \to 0$ limit of the rule on PointMaze and the read-out of Zhao et al. (2025) on the embodied suite, where Preplay tries the former first, so the two need not agree episode by episode. A* here is A* with line-of-sight shortcutting resampled to one cell. **FMM** and **FMM-cov** are a Dijkstra sweep from the goal over the 8-connected free set, diagonal steps charged $\sqrt{2}$, followed by steepest descent of the cost-to-go field it leaves; that descent already advances one cell at a time, so unlike the A* arms they are neither shortcut nor resampled. The name follows the fast-marching method (Sethian, 1996), which the sweep discretizes, but eight directions on a lattice is not an upwind eikonal solve

Table A9: **All nine read-outs on all 24 datasets.** Every arm shares one free set, one controller and one executor. Success (%); three replicates per cell. **exact**$_s$ is the sampled read-out run on the exact tabular kernel rather than the fitted map. This table contains Table 2 entire and adds the three arms omitted there together with the five PointMaze datasets on which every arm scores about 100. Rows follow Table 1.

Suite	Dataset	none	greedy	Preplay	+cov	exact$_s$	A*	cov-A*	FMM	FMM-cov
PointMaze	medium-navigate	38.9±0.2	98.8±1.0	100.0±0.0	100.0±0.0	100.0±0.0	100.0±0.0	100.0±0.0	100.0±0.0	100.0±0.0
	large-navigate	28.9±1.4	74.1±8.5	100.0±0.0	100.0±0.0	100.0±0.0	100.0±0.0	100.0±0.0	100.0±0.0	100.0±0.0
	giant-navigate	0.0±0.0	86.0±4.8	91.2±5.3	99.2±1.1	99.1±0.4	100.0±0.0	97.7±1.0	60.0±0.0	100.0±0.0
	teleport-navigate	6.3±1.1	22.5±4.5	30.9±0.2	30.0±2.8	30.5±1.0	49.1±0.2	30.9±0.4	30.3±1.9	31.6±0.7
	medium-stitch	39.3±0.2	95.2±4.8	100.0±0.0	100.0±0.0	100.0±0.0	100.0±0.0	100.0±0.0	100.0±0.0	100.0±0.0
	large-stitch	29.9±0.2	75.9±6.7	99.7±0.4	100.0±0.0	100.0±0.0	100.0±0.0	100.0±0.0	100.0±0.0	100.0±0.0
	giant-stitch	0.0±0.0	78.1±10.0	99.9±0.2	100.0±0.0	100.0±0.0	100.0±0.0	100.0±0.0	100.0±0.0	100.0±0.0
	teleport-stitch	6.8±1.0	51.2±6.6	52.9±3.2	32.0±1.3	57.9±1.2	49.6±2.6	78.5±2.5	29.7±1.5	80.7±0.9
AntMaze	medium-navigate	0.1±0.2	85.7±1.6	85.2±1.4	86.8±1.2	82.8±1.7	81.3±1.6	86.1±1.2	82.9±1.5	96.1±1.3
	large-navigate	0.0±0.0	16.9±9.7	71.2±3.6	77.3±3.6	70.3±1.8	66.8±3.1	81.3±1.0	64.9±2.9	89.2±0.6
	giant-navigate	0.0±0.0	6.7±5.7	23.3±5.5	30.9±9.5	26.1±1.8	19.1±0.5	36.5±2.8	29.5±1.8	78.4±4.8
	teleport-navigate	25.1±0.5	24.5±2.0	21.1±0.5	23.3±0.4	22.5±0.8	21.3±0.7	22.9±2.0	23.3±0.8	25.1±2.2
	medium-stitch	0.0±0.0	76.4±4.7	77.5±1.2	79.7±0.2	78.9±1.6	75.9±2.6	83.5±1.4	76.1±0.5	96.8±0.9
	large-stitch	0.0±0.0	16.0±2.1	67.1±3.3	79.1±0.9	56.0±3.7	55.3±4.0	81.2±3.1	58.8±0.9	91.5±1.0
	giant-stitch	0.0±0.0	9.5±4.3	28.5±1.1	39.7±2.0	23.6±2.3	14.5±1.5	34.7±1.7	23.1±2.7	86.0±2.1
	teleport-stitch	21.1±1.0	17.1±1.8	19.3±1.9	21.1±1.3	19.3±0.9	19.1±0.7	39.5±1.2	20.1±1.7	58.3±1.0
	medium-explore	0.0±0.0	69.1±7.8	85.7±1.0	83.1±1.6	84.7±2.0	78.3±2.7	74.7±1.0	73.3±0.8	98.5±0.7
	large-explore	0.0±0.0	7.5±3.9	42.8±0.6	70.5±0.4	19.9±3.0	25.5±1.3	51.2±0.9	25.7±1.9	89.2±1.4
HumanoidMaze	medium-navigate	0.4±0.0	44.5±3.8	47.5±1.6	47.1±0.4	47.7±1.0	40.8±2.6	54.1±1.2	39.7±1.6	68.1±1.6
	large-navigate	0.0±0.0	36.7±5.8	40.0±3.8	38.5±2.7	39.7±3.3	43.6±0.6	55.9±2.5	40.4±3.3	60.7±1.0
	giant-navigate	0.0±0.0	8.7±2.7	9.5±3.3	15.5±5.2	12.8±2.1	0.9±1.3	12.0±0.3	0.0±0.0	30.4±1.5
	medium-stitch	0.1±0.2	51.6±4.2	65.9±0.5	68.0±0.6	63.1±1.7	61.9±0.2	69.3±2.2	61.1±3.9	86.9±2.0
	large-stitch	0.0±0.0	29.3±1.5	35.6±2.6	39.6±1.8	38.0±1.7	40.3±3.3	48.0±1.7	42.1±3.0	64.8±1.8
	giant-stitch	0.0±0.0	8.0±1.8	6.9±3.9	8.8±3.4	9.9±1.4	0.4±0.6	7.1±1.9	0.1±0.2	32.1±0.8

and carries the metrication error one avoids. The charge is clipped at $c(x) \leqslant 10^3$ so that a cell the data barely entered cannot make an edge arbitrarily expensive. **+cov** gives the same preference to a rule that maximizes rather than minimizes, so there it enters as a bonus proportional to $n(x)^{+\alpha}$ added to Δ, not as the cost of Equation (6), and is scaled by $\max_z |\Delta|$ at weight 0.5 so that its strength is comparable across maps whose gradients differ by orders of magnitude. **exact**$_s$ is the *sampled* read-out at $T = 0.3$, $K = 16$, the one Preplay uses, run on the exact tabular kernel rather than the fitted map, so that column changes the object being read and not the rule reading it. Spreads are population standard deviations. PointMaze uses the unlearned proportional controller and the embodied suites the trained IQL one, which is why the no-planner floor is not zero there.

Waypoint spacing, not path shape. The A* column of Table A9 is A* with line-of-sight shortcutting whose route is then resampled to the spacing the ascent produces. Running the same planner without that last step, on an identical graph with an identical search and an identical shortcut route but waypoints left tens of cells apart, gives 19.9 against 40.3 on the embodied sixteen, so delivering the same path in coarser installments costs 20.4 points. It is positive on fifteen of the sixteen and largest on the slowest body, +60.6 on `humanoidmaze-medium-stitch`, which is what one expects if the executor is a proportional rule that needs telling where to go often rather than accurately. With the spacing matched, FMM and A* are level (41.3 against 40.3), so nothing here turns on which classical planner is used, only on how its output is handed to the body. It is also the control against reading the map-versus-search gap as a matter of route shape: the A* arm is shortcut and resampled while FMM descends its field one cell at a time, so the two carry different route shapes and score the same.

The gap between cov-A* and FMM-cov. Over all 24 datasets 13.3 points separate cov-A* from FMM-cov on the same graph under the same charge, which we cannot yet explain. The two differ in route post-processing as well as in search: cov-A* is shortcut to remove the grid staircase, and a shortcut can cut across the low-count cells the charge routed around, while FMM-cov keeps the raw cell-by-cell descent of its field, stepping to the neighbor of least cost-to-go. Uncharged they are level on the embodied sixteen at 40.3 against 41.3, so the gap appears only with a non-uniform cost, which does not rule out an interaction between that cost and the post-processing. We leave it open.

N Separating Coverage from Clearance

Section 4.4 finds that support geometry recovers three quarters of the coverage bonus; Figure A5 and Table A10 hold the runs behind that. **clear** replaces the visit count by the distance to the nearest cell

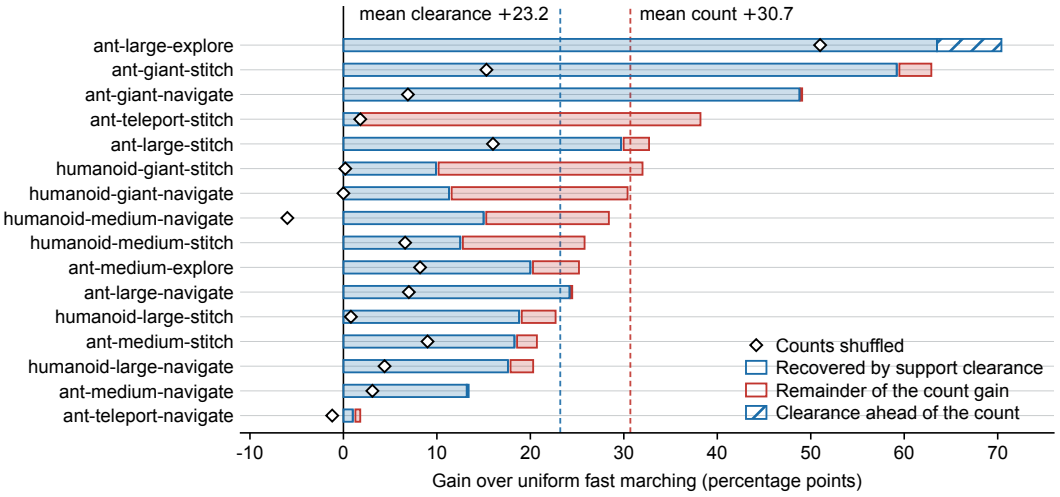


Figure A5: **Geometry against knowledge in the visit-count charge.** Per embodied dataset, the gain of FMM-cov over uniform FMM, split into the part a purely geometric substitute recovers (support clearance, the distance to the nearest never-visited cell, with no counts anywhere) and the arithmetic remainder, which is not a causal share. Diamonds give the same charge with the counts shuffled across occupied cells. Clearance carries most rows; the remainder concentrates on `ant-teleport-stitch` and on the four humanoid datasets outside the `large` pair, which is where that proxy breaks. On `ant-large-explore` clearance is ahead of the count, drawn hatched. Three reset-seed blocks per arm; Table A10 gives the underlying success rates.

Table A10: **FMM under four speed fields on one free set.** Success (%) over three disjoint blocks of 250 episodes: uniform (FMM), the visit-count charge (FMM-cov), support clearance with no counts anywhere (FMM-clear), and the same counts shuffled across occupied cells (FMM-shuf). These are the runs behind Figure A5.

Dataset	FMM	FMM-cov	FMM-clear	FMM-shuf
`ant-giant-navigate`	29.5$\pm$1.8	78.4$\pm$4.8	78.3$\pm$4.0	36.4$\pm$1.1
`ant-giant-stitch`	23.1$\pm$2.7	86.0$\pm$2.1	82.3$\pm$1.9	38.4$\pm$1.0
`ant-large-explore`	25.7$\pm$1.9	89.2$\pm$1.4	96.1$\pm$0.7	76.7$\pm$3.9
`ant-large-navigate`	64.9$\pm$2.9	89.2$\pm$0.6	89.1$\pm$0.8	71.9$\pm$3.5
`ant-large-stitch`	58.8$\pm$0.9	91.5$\pm$1.0	88.5$\pm$1.2	74.8$\pm$2.1
`ant-medium-explore`	73.3$\pm$0.8	98.5$\pm$0.7	93.3$\pm$0.5	81.5$\pm$2.0
`ant-medium-navigate`	82.9$\pm$1.5	96.1$\pm$1.3	96.3$\pm$1.5	86.0$\pm$2.1
`ant-medium-stitch`	76.1$\pm$0.5	96.8$\pm$0.9	94.4$\pm$0.3	85.1$\pm$1.2
`ant-teleport-navigate`	23.3$\pm$0.8	25.1$\pm$2.2	24.3$\pm$0.5	22.1$\pm$1.6
`ant-teleport-stitch`	20.1$\pm$1.7	58.3$\pm$1.0	21.7$\pm$0.5	21.9$\pm$0.8
`humanoid-giant-navigate`	0.0$\pm$0.0	30.4$\pm$1.5	11.3$\pm$1.9	0.0$\pm$0.0
`humanoid-giant-stitch`	0.1$\pm$0.2	32.1$\pm$0.8	10.0$\pm$0.9	0.3$\pm$0.4
`humanoid-large-navigate`	40.4$\pm$3.3	60.7$\pm$1.0	58.0$\pm$2.6	44.8$\pm$1.4
`humanoid-large-stitch`	42.1$\pm$3.0	64.8$\pm$1.8	60.9$\pm$1.4	42.9$\pm$2.3
`humanoid-medium-navigate`	39.7$\pm$1.6	68.1$\pm$1.6	54.7$\pm$1.3	33.7$\pm$0.2
`humanoid-medium-stitch`	61.1$\pm$3.9	86.9$\pm$2.0	73.6$\pm$1.2	67.7$\pm$1.2

the data never entered, so no count enters the speed field at all; **shuf** reassigns the same multiset of counts to occupied cells at random, preserving their distribution and destroying their arrangement. Each arm is three disjoint blocks of 250 episodes, fifty reset seeds at each of five goals. The two controls are not a partition, since $23.2 + 7.7$ exceeds 30.7: they overlap, and neither measures a share specific to the counts.

The routes say the same without any bonus involved. On identical start–goal pairs across four PointMaze datasets the route of the ascent runs $1.20\times$ further from the support boundary than that of A* and passes through cells visited $1.40\times$ more often, at $1.05\times$ the length: it is wider and better traveled, not merely longer. That the count ratio exceeds the clearance ratio when the kernel contains no counts is what one expects if the kernel encodes support geometry and geometry and visit density are correlated in the data.

The exponent of the coverage cost. Table A11 gives the sweep behind the claim in Section 4.4 that α is not tuned. Each cell is the mean over three disjoint blocks of 250 episodes, on the four

Table A11: **FMM under the coverage cost at three exponents.** Success (%), mean and population s.d. over three disjoint blocks of 250 episodes, under $c(x) \propto n(x)^{-\alpha}$. The reported configuration is $\alpha = \frac{1}{2}$.

Dataset	$\alpha = \frac{1}{4}$	$\alpha = \frac{1}{2}$	$\alpha = 1$	α=1 over $\frac{1}{2}$
`ant-giant-stitch`	$74.0_{\pm 2.6}$	$86.0_{\pm 2.1}$	$91.6_{\pm 1.4}$	+5.6
`ant-large-explore`	$82.3_{\pm 0.4}$	$89.2_{\pm 1.4}$	$93.7_{\pm 0.8}$	+4.5
`ant-medium-navigate`	$92.4_{\pm 2.0}$	$96.1_{\pm 1.3}$	$98.0_{\pm 0.9}$	+1.9
`humanoid-large-stitch`	$63.7_{\pm 1.1}$	$64.8_{\pm 1.8}$	$67.3_{\pm 1.9}$	+2.5

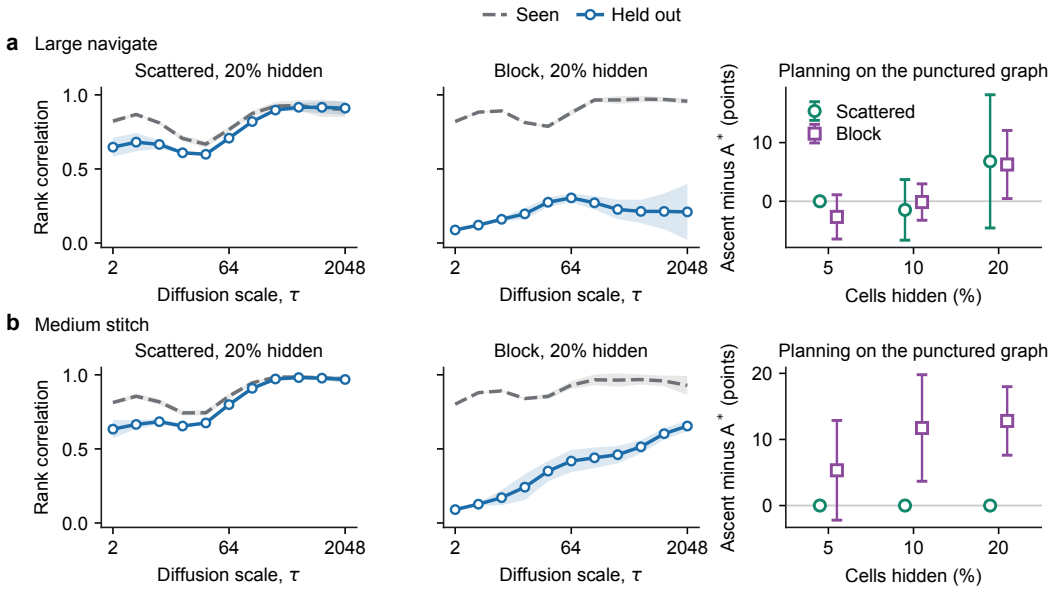


Figure A6: **Querying the map on cells it never saw.** Rank correlation against the exact kernel on hidden and on seen cells at every scale, with 20% of the free set hidden scattered or as one block, and the gain of the ascent over A* on the same punctured graph. Scattered, the gap closes at long horizons; contiguous it does not, and that is where the ascent leads search. Rows are (a) `large-navigate` and (b) `medium-stitch`.

datasets the sweep was run on. $\alpha = 1$ is better on all four, but by 1.9 to 5.6 points rather than by a constant, and the smallest of those gaps is inside the device dispersion of Section E. The sweep establishes the sign, on four datasets, and that the charge is robust to its exponent over a factor of four.

O Querying the Map off Its Support

The map has no accuracy advantage over the table it distills or the planner distilled from it, so what it has instead is that it is a function. Hiding free cells from the kernel and from fitting, then restoring them at evaluation, asks the network for embeddings at positions in no training pair, and scattered over the maze at 5 to 20% hidden it answers, where a stored graph has no entry at all, but not as well as on cells it saw: averaged over the eleven scales the rank correlation is 0.75–0.85 held out against 0.82–0.92 seen. The gap has a shape. At $\tau = 256$ it has nearly closed (0.89–0.99 against 0.90–1.00) and at $\tau = 2$ it is wide (0.56–0.71 against 0.76–0.85): what the network interpolates is the smooth long-horizon kernel, and what it does not is the few-cell variation the ascent terminates on. Planning is a separate measurement, since both arms plan on the punctured graph and neither enters a hidden cell. With the hidden cells scattered, search on the punctured graph loses almost nothing and the map leads by 0.0, -0.7 and $+3.4$ points at 5, 10 and 20% hidden. With them contiguous, where the kernel predictions are worst, the map leads by 1.3, 5.8 and 9.5 points over the same fractions. The average hides a disagreement between the two mazes, since on `medium-stitch` the lead is 5.3, 11.7 and 12.8 while on `large-navigate` it is -2.7, -0.1 and 6.3. Held-out correlation itself collapses in this regime, to 0.15–0.65 against 0.87–0.93 seen. The field interpolates and does not extrapolate.

Table A12: **A map fitted on one body planning for another.** Success rate (%) on `large-navigate`, three independently trained maps at 250 evaluation episodes each; the two suites share an arena, and only the map is exchanged. The humanoid loses little when given the ant map; the ant loses more when given the humanoid map, and the text separates the causes.

Body	Controller	Map	Success
Humanoid	humanoid	humanoid, 40×40	$40.0_{\pm 3.8}$
Humanoid	humanoid	**ant**, 60×60	$36.7_{\pm 7.3}$
Ant	ant	ant, 60×60	$71.2_{\pm 3.6}$
Ant	ant	**humanoid**, 40×40	$31.6_{\pm 3.5}$

P The Map Does Not Depend on the Body

If the kernel is a statistic of where the data went rather than of what produced it, a map fitted on one embodiment should be usable by another in the same place. OGBench permits the test directly: `maze_unit` is 4.0 for every body, so `antmaze-large` and `humanoidmaze-large` are the same arena in world coordinates. We exchange the maps and leave everything else alone, the controller always belonging to the body being evaluated.

Table A12 gives the four cells. The humanoid loses little: 40.0 with its own map against 36.7 with a map built entirely from ant trajectories, a gap smaller than the spread over maps. The ant loses more, 71.2 against 31.6, and two controls separate the causes. Rebuilding the map of the ant *itself* at the 40×40 resolution of the humanoid gives 64.4 ± 1.7, so resolution accounts for 6.8 of the 39.6 points. The rest is the support: the humanoid data covers all 887 cells the ant visits and 305 further cells it reaches and the ant cannot, and routes through those are found (100% routed) and not executable. Intersecting the free set of the transferred map with the support of the ant returns 66.8 ± 0.7, level with a map fitted on ant trajectories, so the kernel is a statistic of the place, not of the body; what fails to transfer is the reach of the other body. Even on the shared cells the bodies weight them differently, at visit-density Spearman 0.415 and total-variation distance 0.362, which is the quantity Section 4.4 turns into a planner. Transfer requires carrying the grid frame with the embedding: a cell index means a place only relative to the extent and resolution it was built with, and handing over the embedding alone returns exactly 0.0 in both directions. The map is therefore the transferable half of the system and the controller is not, the same division Figure A2 draws within a single body.

Q Pixels: The Map Transfers, the Controller Does Not

The reason to want a planner that does not search is that the same procedure should run wherever the kernel can be estimated, including on observations that admit no usable metric over states. We built that pipeline. The map half transfers; the controller half does not, and the split is the same one that limits the humanoid in Figure A2.

The construction is the state-based one with two substitutions. Frames are encoded by a temporally contrastive encoder, the encoded frames are clustered into prototypes that stand in for grid cells, and the one-step kernel is estimated from the transitions that consecutive frames perform between prototypes. No coordinates enter it at any point; the dataset stores true positions, and we use them only to check afterwards what was built.

The check rules out a degenerate solution. An ascent can route on 100% of the benchmark start–goal pairs and be worthless if the routes consist of 6.7 waypoints with a mean true displacement of 8.5 units in an arena forty units across, since such an ascent crosses the maze in a few hops through edges no body could traverse. Two sources produce those edges: episode boundaries the subsampling had not masked, one end-to-start transition per episode, and a small population of prototypes that pool visually similar corridors from different parts of the maze. Masking the boundaries removes half of them. Keeping only edges whose transition was observed at least eight times, a criterion that uses transition counts and not positions, removes the rest; Section R calibrates the encoder's metric and reduces, but does not remove, the aliasing behind the second source.

On the resulting map the ascent again routes on 100% of pairs, and this time the routes survive inspection: 37.4 waypoints with a mean true step of 1.96 units and a maximum of 5.01, against 6.17 before pruning. The map is a real map. Two costs come with it: the count threshold isolates about

Table A13: **Three encoder objectives for the pixel pipeline.** Correlation is between latent distance and true displacement over pairs up to twenty steps apart; the neighbor distance is the true separation of the eight latent-nearest prototypes, against 15.9 for random pairs; the edge statistics are of the transition graph after the count threshold. Each row is one encoder trained once, so these cells carry no spread, as do the single-run columns of Tables A1, A3, A5 and A6.

Encoder objective	corr. with displacement	neighbor dist.	edges $>$ 4u	longest edge
contrastive only	0.765	2.03	2.3%	6.10
isometry only	**0.858**	**1.08**	1.7%	11.30
isometry + contrastive	0.780	1.79	**0.6%**	**5.80**

13% of the prototypes, which are then unreachable, and the waypoint spacing is roughly three times what the state-based system uses.

Closed-loop success is nonetheless 0% of 50 episodes. The controller fails first: the ascent routes on every episode, and removing the waypoints entirely and conditioning the controller on the goal image alone changes the distance traveled from 1.93 to 2.09 units, which is to say not at all. It is the controller, and the controller class is the reason. Cloning actions from the data reaches $R^2 = 0.878$ on held-out actions while moving the ant two units in a thousand steps, because the latent identifies the place but not the phase of the gait, so the conditional mean action over a gait cycle, which is near zero, fits well and does nothing. Giving the controller its own encoder and a stack of three frames, so that limb motion is visible, tripled the distance traveled and left success at zero. Porting the state-based recipe faithfully, action-free IQL with advantage-weighted regression, also left it at zero. That is where the benchmark itself says this ends: on this dataset Park et al. (2025) report 4 for goal-conditioned behavioral cloning, 5 for GCIVL and 4 for GCIQL, against 53 for HIQL and 84 for CRL. The controller families we ported are the ones that do not work here, and making them work is a question about visual control rather than about maps.

We therefore claim what we measured: on pixels, with no coordinates anywhere in the construction, the kernel can be estimated and planned on, and the routes it produces are routes. Whether a body can be driven along them is a separate problem that this paper does not solve.

R Calibrating the Latent Metric by Isometry

The aliasing that forces the count threshold in Section Q has a specific cause: a contrastive objective asks that adjacent frames be close and says nothing about frames that are far apart in the world but similar on the screen. Corridors in this maze look alike, so nothing prevents two of them from collapsing together.

A condition that does forbid it is the conformal isometry proposed for grid cells (Xu et al., 2025), $|v_{t+1} - v_t| = s|a_t|$: if latent distance is proportional to displacement, two places many steps apart cannot be close in the latent, whatever they look like. For this body a_t is joint torque and a poor measure of displacement, and the pixel setting denies us positions, so we impose the condition against elapsed steps, $|v_{t+k} - v_t| = sk$ for $k \leqslant 8$, with s learned, an empirical proxy that constrains nearby pairs only. The encoder output is left unnormalized, since on the unit sphere distances saturate at 2 and could not grow with k.

Table A13 gives the comparison. The two terms do different jobs, and the combination works. Isometry alone gives the best-calibrated metric, with latent neighbors nearly twice as tight as the contrastive ones, but without a term that settles which frames are the same place it leaves an eleven-unit edge in the pruned graph. Contrastive alone settles identity and leaves the scale free. Together they give the cleanest graph, and carried through the full pipeline they cut the longest step in a planned route from 6.17 to 5.01 units and the isolated prototypes from 123 to 113.

These are improvements of a few points, not a repair. Without any pruning the longest edge is still about forty units under all three objectives, so the isometry reduces the aliasing without eliminating it, and the count threshold remains necessary. We report it because the direction is right and measurable, not because it closes the gap.

S THE FOUR CHANGES THAT DID NOT WORK

Four changes that follow naturally from the construction were implemented and measured, and all four failed. The first is to hand the controller a sub-goal a fixed distance ahead along the route rather than the next waypoint, so that the body can move ballistically instead of decelerating at every waypoint. It costs 42 points on `antmaze-large-navigate` at a horizon of eight units and is monotone in the horizon. The reason is geometric: a route through a maze is a staircase, and a point several units ahead along it is often reached only by walking through a wall.

The second is to let the scale set the length of the step and not only its direction. The place field at scale τ has radius $\sim \sqrt{\tau}$, so a stride of that order is the displacement the representation resolves there, and the ascent would become coarse-to-fine on its own. It costs 46 points on `pointmaze-giant-navigate` and lengthens plans from 65 to 244 waypoints. A strided move can step over the goal cell without landing on it, after which the ascent circles the goal; repairing the termination test so that a stride stops on passing through the goal does not recover the loss.

The third is to estimate the one-step kernel from the transitions the dataset performed rather than from a lattice walk, which removes the assumption that spatial adjacency implies reachability and would let the map represent a teleporter. The estimated kernel is the healthier graph, fully connected on `pointmaze-teleport` where the lattice walk leaves eight cells isolated. The ascent then routes on 0% of episodes, because the fitted rank correlation collapses from 1.00 to 0.45 at the longest scale, and the ascent depends on the long scales for its long-range signal. It is not that a network reading coordinates cannot represent a kernel with a discontinuity in space: giving it the leading eigenvectors of the one-step kernel as input instead of the coordinates reproduces the lattice result exactly and does not rescue the estimated one. The reason, given in Section 4.6, is that the estimated target is directed and its symmetric part is not PSD, so no nonnegative Gram kernel can represent it.

The fourth follows from the theory rather than from the implementation, which is what makes it worth reporting. The construction descends from the heat equation with $dx = \sqrt{dt}$, so $\sqrt{\tau}$ *is* the spatial scale and adjacent scales sit a factor $\sqrt{2}$ apart (Zhao et al., 2025). We hold the lattice fixed and vary only τ, which honors the time half of that relation and not the space half: at $\tau = 2048$ the theory asks for a field of radius 45 cells in a grid 40 wide, and Section 4.1 shows the truncation in the fitted exponent. Giving each scale its own lattice, coarsened so that every field spans a constant number of cells, should therefore be the more faithful construction. It is much worse: `pointmaze-large-navigate` falls from 100.0 to 28.0 and `pointmaze-giant-navigate` from 91.2 to 0.0.

The reason is visible in the kernels. Measuring the spread of q_τ over off-diagonal pairs, the fine grids at small τ give 0.0000, the walk having not diffused far enough to produce a gradient, and the coarse grids at large τ fall back to 0.0001 and below, the walk having mixed completely on a lattice of 84 free cells. Only a band in the middle carries any signal, and it is narrower on the maze that fails harder: six of the eleven scales on `large-navigate` against four on `giant-navigate`. More fundamentally, holding the field at a constant number of cells while shrinking the cell makes the eleven scales eleven copies of one local structure at eleven resolutions; relative to the arena there is no longer a coarse scale that sees the whole maze, which is exactly what the adaptive scale selection needs. The fixed lattice is not a defect to be corrected. It is what produces a range of spatial scales relative to a fixed arena, and the truncation of the largest fields is the price of having them.

T REPRODUCIBILITY

All datasets are the public OGBench benchmarks, used through the released package and its registered environments, and the baseline numbers are as published. The map is fitted by regression, the planner is seeded and the evaluation fixes the environment reset seed, so a rerun of the harness reproduces a cell bit for bit on the device that produced it; we checked this in both directions, two evaluations rerun on CPU and, in separate processes on one GPU, both an arm that reads the map and one that does not. It does not reproduce *across* devices: moving the low-level controller between GPU and CPU, with everything else held fixed, moves a cell by 2.7 points on average and by as much as 6.8 (Section E), because the executor amplifies the last bits of a matrix multiply over a

few thousand steps of contact dynamics. A reimplementation should expect to match the reported means to within that, not to the decimal.

Each display item in the paper has a directory under `sources/` holding the numbers it is drawn from as JSON, their provenance, and the script that regenerates it; running that script rebuilds the figure or prints the table body without touching a cluster. The numbers themselves are regenerated from the logged output of every individual run by the scripts in `diag/`, so no value in the paper is transcribed by hand except the published baselines, which are transcribed from Table 2 of Park et al. (2025) and checked twice.

The configuration is identical on all 24 datasets and is listed in full in Table A2; the two quantities that vary by suite, the grid size and the relabeling horizon, are a per-suite choice and a property of the body, neither set on any result (Section C). All source code and the evaluation logs will be released upon acceptance.