A mechanistic theory of planning in prefrontal cortex

Kristopher T. Jensen[@1,3], **Peter Doohan**[2], **Mathias Sablé-Meyer**[1], **Sandra Reinert**[1], **Alon Baram**[3], **Thomas Akam**[1,2], **and Timothy E. J. Behrens** [1,3]

[1] Sainsbury Wellcome Centre, University College London, London, UK
[2] Department of Experimental Psychology, University of Oxford, Oxford, UK
[3] Oxford Centre for Integrative Neuroimaging, University of Oxford, Oxford, UK
[@] Corresponding author (Kris.Torp.Jensen@gmail.com)

Abstract

Planning is critical for adaptive behaviour in a changing world, because it lets us anticipate the future and adjust our actions accordingly. While prefrontal cortex is crucial for this process, it remains unknown how planning is implemented in neural circuits. Prefrontal representations were recently discovered in simpler sequence memory tasks, where different populations of neurons represent different future time points. We demonstrate that combining such representations with the ubiquitous principle of neural attractor dynamics allows circuits to solve much richer problems including planning. This is achieved by embedding the environment structure directly in synaptic connections to implement an attractor network that infers desirable futures. The resulting 'spacetime attractor' excels at planning in challenging tasks known to depend on prefrontal cortex. Recurrent neural networks trained by gradient descent on such tasks learn a solution that precisely recapitulates the spacetime attractor – in representation, in dynamics, and in connectivity. Analyses of networks trained across different environment structures reveal a generalisation mechanism that rapidly reconfigures the world model used for planning, without the need for synaptic plasticity. The spacetime attractor is a testable mechanistic theory of planning. If true, it would provide a path towards detailed mechanistic understanding of how prefrontal cortex structures adaptive behaviour.

Introduction

While different cortical areas support different functions, common computational principles are shared across many areas. For functions as disparate as sensory processing, spatial reasoning, and language comprehension, features of the environment are inferred from partial information. To do so, structural knowledge about the world must be embedded in synaptic connections (Ko et al., 2011; Burak and Fiete, 2009; Iacaruso et al., 2017; Turner-Evans et al., 2020). This constrains neural circuits to represent meaningful interpretations of the environment, and inputs select between these interpretations. It is not known whether similar principles generalise to complex prospective behaviours, such as planning extended action sequences to achieve a distant goal. Recent recordings from prefrontal cortex give an important clue. When mice or monkeys must execute a sequence of actions, neurons represent the entire sequence concurrently (El-Gaby et al., 2023; Xie et al., 2022). Different neuronal populations encode different steps of the future behaviour. If such a representation could be inferred from inputs indicating goals and constraints, the network could plan the future. Excitingly, this solution would use algorithmic principles similar to those known to infer features of the present in other cortical areas.

This paper has four overlapping aims. (i) To develop a detailed circuit model that infers explicit representations of the future from partial inputs. (ii) To discern the principles of synaptic connectivity that enable such a model to solve complex problems including planning. (iii) To understand when and why this algorithm succeeds while simpler circuit models fail. (iv) To explore how it relates to prefrontal representations, connectivity, and function.

Sequence representations in PFC Recent work has uncovered PFC representations underlying sequence working memory (Xie et al., 2022; El-Gaby et al., 2023; Whittington et al., 2023; Botvinick and Plaut, 2006). Separate neural 'subspaces', or groups of neurons, represent the expected or desired state of the world at different steps along the sequence of future behaviour. In other words, some neural subspace 'A' represents the present, while subspace 'B' represents the immediate future, and subspace 'C' the more distant future (Figure 1A). Critically, these subspaces are active simultaneously. Together, they instantaneously represent the entire behavioural sequence. Importantly, El-Gaby et al. (2023) showed that the PFC subspaces representing different steps of the future are not independent. Neuronal correlations reflect the structure of the task being performed, even during rest.

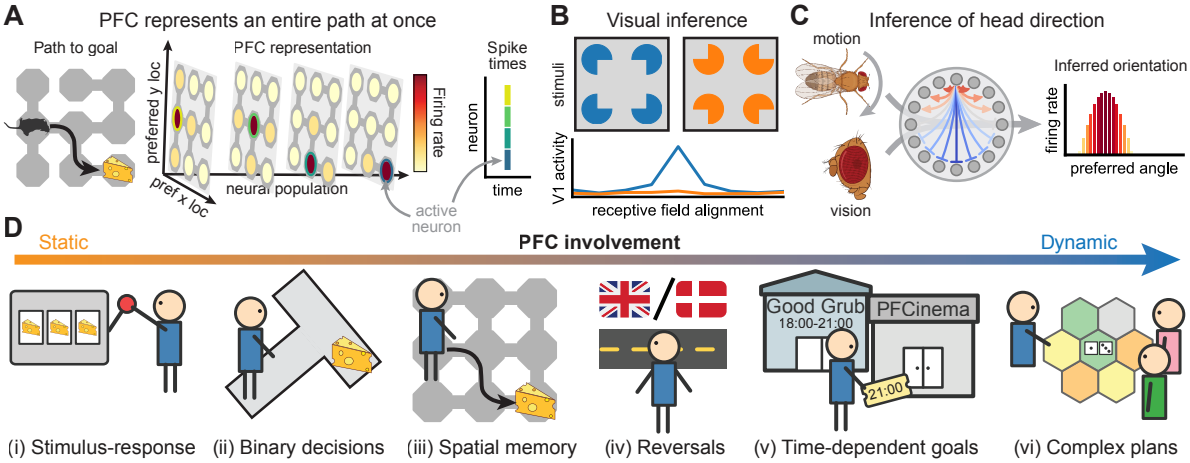

Figure 1: **Background. (A)** Recent work has identified explicit sequence representations in prefrontal cortex during working memory (Xie et al., 2022; El-Gaby et al., 2023). When an animal has to execute a behavioural sequence (left), individual neurons represent conjunctions of location and sequence element (centre). Separate populations (planes) therefore represent the expected location at different times in the future. The entire sequence is represented concurrently by the simultaneous firing of different neurons that encode the expected location at each time in the future (right). **(B)** An example V1 cell fires when its receptive field is aligned with an inferred line (blue), but not for a control stimulus with no inference (orange). Such visual inference is mediated by structural priors embedded in the circuit connectivity, where neurons representing consistent visual features excite each other (Iacaruso et al., 2017; Shin et al., 2023). Figure adapted from Lee and Nguyen (2001). **(C)** Structural knowledge is embedded in the synapses of the head direction system (centre; Turner-Evans et al., 2020), which constrains the network to represent single angles (Kim et al., 2017). Visual and proprioceptive inputs (left) determine which angle should be represented (right). We suggest that a mechanism like (B) and (C) infers the representation in (A). **(D)** Prefrontal cortex is particularly important in dynamic environments. Stimulus-response associations and repeated choices are robust to prefrontal lesions (i; ii), and acortical mice can solve spatial memory tasks (iii; Zheng et al., 2024). However, PFC is needed for reversal learning (iv; Walton et al., 2010) and when different goals are important, or 'rewarding', at different times (v; Shallice and Burgess, 1991). In multiplayer board games (vi), different resources are valuable at different stages, and opponents can dynamically change their strategy.

Planning by inferring the future The correlation structure observed in PFC resembles other attractor circuits known to infer the current state of the world, such as the instantaneous visual input or head direction of an animal (Figure 1B-C; Ko et al., 2011; Chaudhuri et al., 2019). If a similar inference process acted on the future representation in PFC, it would extend its function beyond sequence memory to situations where entire action sequences are inferred from partial cues. Planning could be solved by inferring sequences of desired actions from a set of goals in a network implementation of 'planning as inference' (Botvinick and Toussaint, 2012; Levine, 2018). The resulting algorithm would naturally cope with dynamic environments, because it represents each time in the future separately. This is intriguing because PFC is particularly important for problems that require the correct behaviour to be expressed at an appropriate time (Figure 1D; Shallice and Burgess, 1991; Volle et al., 2011). Planning via attractor dynamics is also consistent with winner-take-all dynamics identified in frontal cortex during non-sequential behaviours (Ruff et al., 2025; Inagaki et al., 2019). Such planning as inference differs from most planning algorithms studied in cognitive science and machine learning, which often rely on sequential search (Callaway et al., 2022; Schrittwieser et al., 2020). Search is easily adapted to new environments, but it is slow at decision time. In familiar environments where the structure is embedded in cortical connections, attractor dynamics provide a complementary mechanism for rapid evaluation of many possible futures in parallel.

A mechanistic theory In this paper, we show that known features of PFC representations and connectivity are sufficient to implement a powerful planning algorithm with minimal additional assumptions. The resulting 'spacetime attractor' (STA) instantiates an explicit world model in the synaptic connections between neurons, which allows it to plan by inferring optimal future trajectories. This

algorithm resembles other cortical circuits known to infer features of the present, thereby unifying our understanding of PFC with the rest of cortex. The spacetime attractor excels at dynamic problems with changing reward and transition structures, which PFC is critical for and existing mechanistic models cannot solve. RNNs trained on dynamic tasks implement a spacetime attractor in their dynamics, suggesting that it is an efficient solution. Our findings provide a precise mechanistic theory of adaptive behaviour that reconciles prior work on PFC representations with other known cortical computations and deficits from lesions.

Results

An attractor network in space and time

We now introduce the spacetime attractor in more detail. We first review ring and grid attractors, which illustrate how simple circuit motifs can guide inference from partial information. We then explain how the spacetime attractor uses similar principles to infer future behaviour. Neurons are assumed to encode single environment features, but similar ideas apply when individual neurons represent combinations of variables (Clark et al., 2025).

Ring attractors Ring attractors are neural networks that can only encode circular variables such as angles (Zhang, 1996; Ben-Yishai et al., 1995). Different neurons have different 'preferred orientations', and the network dynamics have a set of 'fixed points' (stable activity patterns) that encode a particular angle. At these fixed points, neurons with a preferred angle near the encoded angle are active, and the remaining neurons are silent (Figure 2A, bottom left). This is achieved by a network where neurons are mutually excitatory if they have similar preferred orientations, and inhibitory otherwise (Figure 2A, top left). The network then *infers* which angle is most compatible with noisy inputs such as vision and proprioception. This property underlies the ability of ring attractors to integrate angular velocity in the head direction system (Turner-Evans et al., 2017; Skaggs et al., 1994).

Grid attractors Grid attractors instead encode position in two-dimensional space. This is achieved by neurons that are mutually excitatory if they represent nearby locations in space and inhibitory if they represent more distant locations (Figure 2A, top right; Fuhs and Touretzky, 2006; Burak and Fiete, 2009). The fixed points of the network are hexagonal patterns of activity that resemble canonical grid cells (Figure 2A, bottom right; Hafting et al., 2005), and the inputs determine which location is represented at a given moment in time. These properties underlie the ability of grid cells to perform path integration (Burak and Fiete, 2009).

Spacetime attractors Inspired by these known attractor networks in the brain, we propose the existence of a *spacetime attractor* in prefrontal cortex. The STA is a network that infers explicit representations of the future, and its fixed points therefore have to be entire paths through space and time (Figure 2B, bottom). Given such fixed points, any input would automatically be converted into a representation of future behaviour – an inferred plan. Similar to the ring and grid attractors, this is achieved by connecting STA neurons such that *consistent* states excite each other and *inconsistent* states inhibit each other. In the spacetime attractor, each neuron has both a preferred location and a preferred delay δ, which determines how far into the future its tuning curve is defined. If a neuron has $\delta = 0$, it will fire when the agent is currently at the preferred location of that neuron. If a neuron has $\delta = 3$, it will fire when the agent expects to be at the preferred location after three actions. In such a network, consistent states are those that can be part of a single trajectory, while inconsistent states cannot be part of the same trajectory. The connectivity between neurons in subspaces with preferred delays δ and $\delta+1$ should therefore correspond to the structure – more specifically the adjacency matrix – of the environment (Figure 2B, top).

Reward input selects the future By embedding a world model in its connections, the spacetime attractor creates fixed points that are future trajectories through space and time. It can then use reward information from the environment to bias the representation towards trajectories that represent desirable futures. This resembles how visual and proprioceptive inputs to a ring attractor bias its representation towards particular orientations (Kim et al., 2019). Reward information must similarly be an input to the STA to enable fast adaptation to changing rewards without rewiring the synaptic connections. Importantly, the STA can accommodate time-varying reward structures known to engage PFC (Figure 1D; Shallice and Burgess, 1991; Volle et al., 2011; Carlesimo et al., 2014). This is possible because different neural populations that represent different times in the future can receive different inputs. Given a restaurant booking in 2 hours and a cinema ticket in 4 hours, the $\delta = 2$ neurons would receive reward input at the restaurant, and the $\delta = 4$ neurons at the cinema. The network dynam-

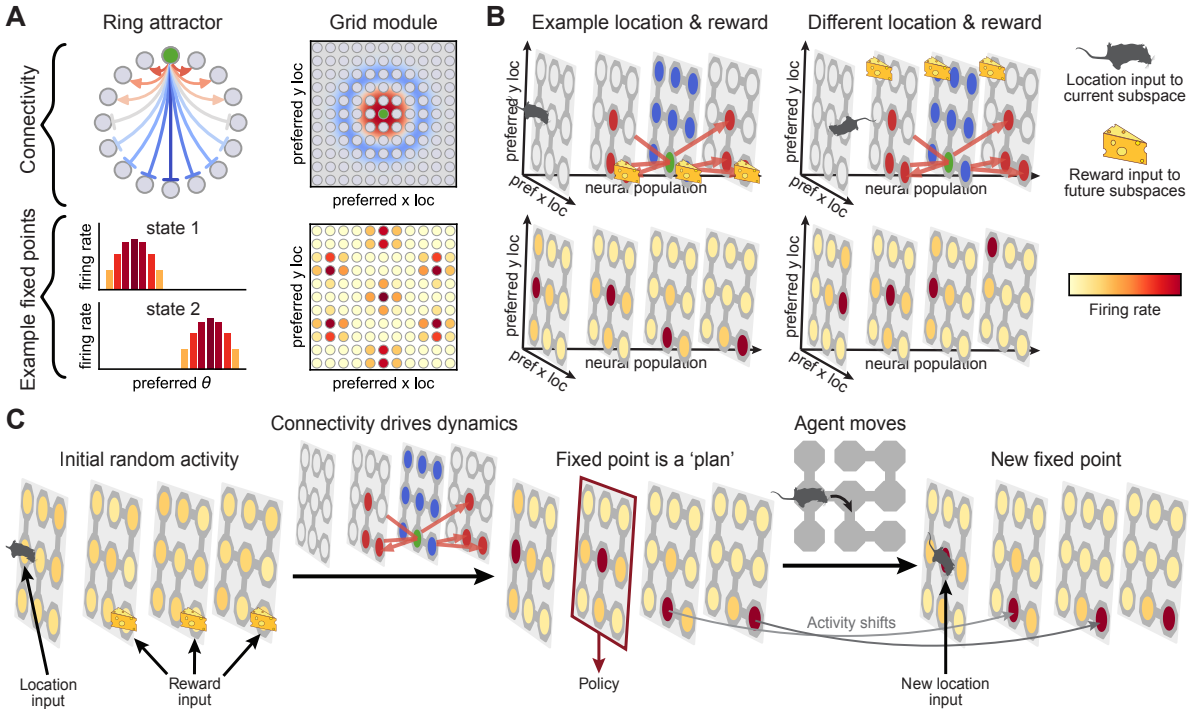

Figure 2: **The spacetime attractor. (A)** Left; in a one-dimensional ring attractor, neurons representing similar directions excite each other (top, red), and different directions inhibit each other (blue). Connections are shown for the green example cell. Stable network states (fixed points) have neurons at a particular encoded angle being most active (bottom). Right; grid cells (bottom) emerge from a two-dimensional attractor network (top), where cells with similar preferred location excite each other (red) and intermediate distances inhibit each other (blue). **(B)** The spacetime attractor is a three-dimensional generalisation, where neurons have both a preferred location (planes) and delay (horizontal axis). This resembles the PFC representation in Figure 1A. Cells that represent adjacent locations in both space and time excite each other (top, red), and different locations at the same time inhibit each other (blue). Given inputs indicating the current location (mouse) and future reward (cheese), the fixed points represent reward-maximising paths through space and time (bottom). While this example has a single static goal, reward inputs can also differ between subspaces if the reward is dynamic. **(C)** Dynamics of the spacetime attractor. Initial activity is diffuse (left), followed by convergence to a stable representation of a plan (centre). At convergence, a policy is read out from the subspace that represents the next location along the desired trajectory. When the agent moves to the next state, neural activity updates to represent the remaining plan-to-go (right; El-Gaby et al., 2023).

ics would then infer a future where the agent is at each location at the appropriate time.

To summarise, the spacetime attractor has four main components: (i) different neural populations represent different times in the future; (ii) the neurons are connected according to the structure of the environment; (iii) the current location is an input to the 'present' subspace; and (iv) the reward structure is an input to all future subspaces. The resulting fixed points (Figure 2B, bottom) represent trajectories that maximise cumulative reward. The network dynamics implement a gradual relaxation process, where reward inputs bias the representation in each subspace towards high-reward locations, while inputs from neighbouring subspaces constrain their representations to be consistent. Together, these two processes converge to a stable representation of a coherent future plan (Figure 2C).

The STA guides behaviour After computing an explicit representation of a plan, it can be used to inform behaviour. In particular, an agent should simply take whichever action leads it to the next location on the inferred trajectory (Figure 2C, middle). The entire representation of the future then needs to move by one 'action', so the state that was previously represented in subspace δ is now in subspace $\delta - 1$. The resulting dynamics resemble a 'conveyor belt' that allows the STA to execute entire trajectories without recomputing the policy after every action. Such conveyor belt dynamics have been observed experimentally during sequence working memory (El-Gaby et al., 2023). Impor-

tantly, the representation remains a fixed point of the spacetime attractor dynamics, because the location and reward inputs are updated due to (i) the movement of the agent, and (ii) the passing of time.

World models for planning Planning requires access to an internal world model that predicts the consequences of different actions. However, the spacetime attractor uses such a world model differently from many other planning algorithms. Most algorithms apply a world model sequentially to simulate actions one by one. The spacetime attractor instead instantiates multiple copies of the world model explicitly in the synapses between different neural subspaces. This allows the network to simulate many possible futures in parallel. Entorhinal grid cells are also thought to embed a world model in their connectivity (McNaughton et al., 2006), but they do not represent the distant future explicitly (Ouchi and Fujisawa, 2024). Instead, they infer a single location at a time. The spacetime attractor therefore suggests that circuit principles in prefrontal cortex resemble other cortical areas that use structural knowledge to infer features of the world. We propose the major difference to be that PFC instantaneously represents the world at many points in time, which extends known circuit principles to complex planning problems. This also requires knowledge of the reward available in different states, which could be estimated separately by other neural circuits. In this work, we simply assume access to ground-truth rewards.

STAs are flexible planners

Prefrontal cortex is particularly important for tasks that require flexible behaviour in changing environments (Figure 1D; Burgess and Wu, 2013). To understand whether the spacetime attractor is a good model of planning in PFC, we therefore need to (i) study its behaviour and performance in such dynamic tasks, and (ii) characterise when and how it differs from existing models. We will see that the spacetime attractor is well-suited to dynamic 'PFC-like' problems, which other mechanistic models struggle to solve.

From static to dynamic tasks We designed a set of four tasks that vary in how much the reward changes in space and time (Figure 3A). The tasks are all embedded in Euclidean space for simplicity of exposition, but the underlying principles generalise to any environment with known structure. Task 1 is simple navigation towards a static goal that remains constant across trials. In task 2, the static goal changes between trials. Task 3 is navigation to a goal that also moves within each trial, where each location is only rewarded when the goal is there. Task 4 generalises the idea of time-dependent goals to a non-binary reward landscape. Reward magnitudes are sampled independently between -1 and +1 for each location at each time point in each trial. The objective is to maximise cumulative reward over the trial, which requires balancing immediate reward with the potential for future reward (Figure 3A, right). This is reminiscent of the example from Figure 1D, where the restaurant and cinema are desirable at different times. To better understand how the STA solves these tasks, we compare it to two models commonly studied in systems neuroscience. The first is temporal difference (TD) learning, which has neural correlates in striatum (Sutton, 1988; Schultz et al., 1997). The second is the successor representation (SR), which has neural correlates in hippocampus (Dayan, 1993; Stachenfeld et al., 2017). These models assume fixed rewards across trials (TD) or within a trial (SR). Unlike the spacetime attractor, they do not generalise well when the reward changes rapidly.

Simpler models can solve static tasks TD learners gradually propagate value from rewarded locations to all other locations, and optimal policies are only learned when rewards are constant across both time and trials (Figure 3B-C, orange). The SR agent uses the environment structure and trial-specific reward function to compute values, and it can solve tasks where the goal changes between trials (Figure 3B-C, green). The STA can also solve these tasks, but it does so by inferring an entire spatial trajectory to the goal instead of computing a value function (Figure 3B-C, blue). Because the reward is constant throughout a trial, each subspace receives the same reward input. However, the inferred representation differs across subspaces because it is also constrained to satisfy the current agent location and the transition structure of the world. As we will see, the inference of an explicit spacetime representation by the STA increases its flexibility compared to the amortised state values computed by TD and SR agents.

Only the STA solves dynamic tasks While the SR agent can adapt to rewards that change across trials, the reward structure still needs to remain constant for the duration of the planning horizon. This is because the SR computes time-averaged occupancy and lacks fine-grained information about when the agent is where. When the reward changes within a trial, the SR is limited to computing average values across time-within-trial, and it fails

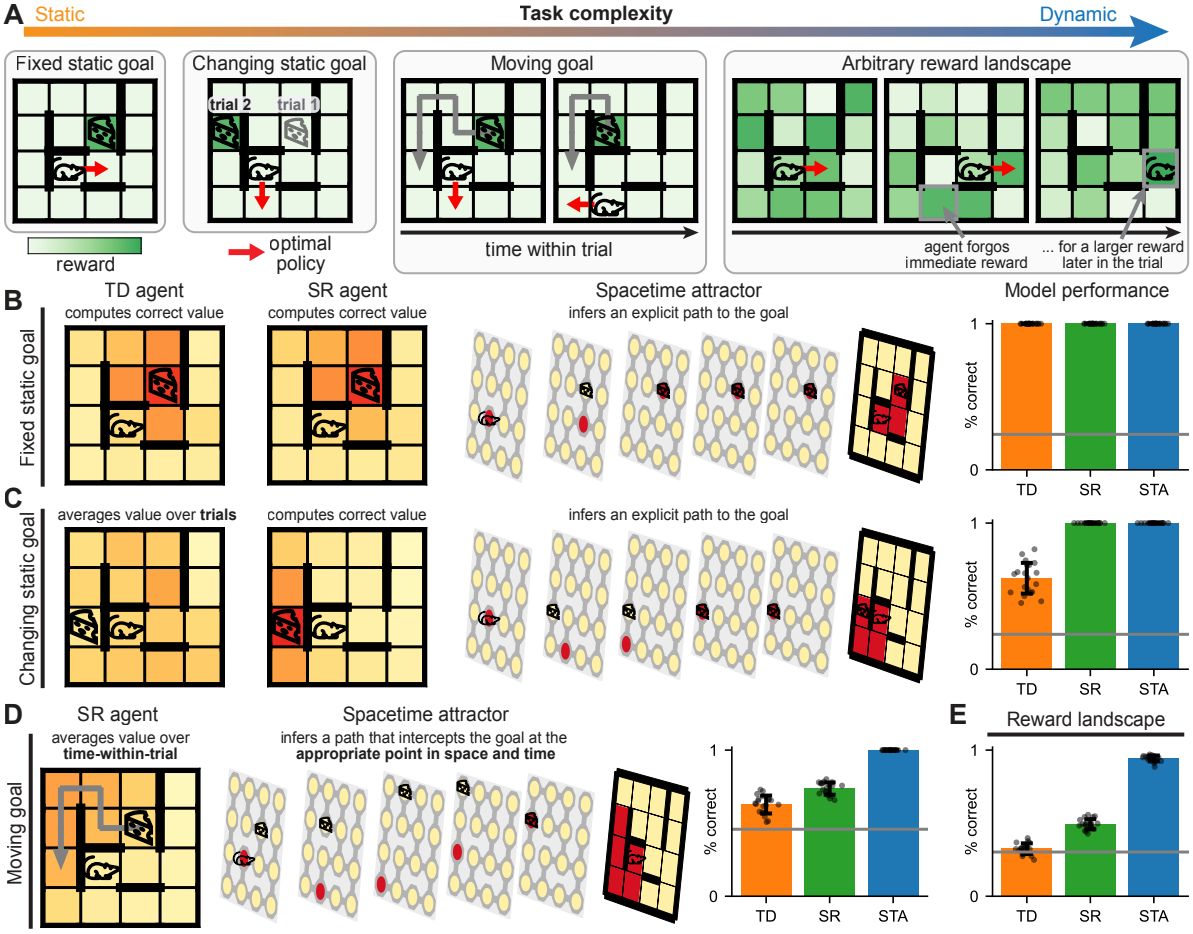

Figure 3: **Model comparisons in static and dynamic tasks. (A)** Example tasks with different degrees of dynamic reward. Colours indicate reward at each location (white to green), and red arrows indicate the optimal policy. In the 'moving goal' task, the agent intercepts a target that moves along a different trajectory in each trial (arrow). In the 'reward landscape' task, the reward is sampled independently in space and time. The agent has to maximise cumulative reward, and it can be optimal to forgo immediate reward for a later payoff. **(B)** Representations and performance in the static goal task with a fixed goal. Left: value functions learned by TD and SR agents. Centre: STA representation at convergence, which encodes a path through space and time. The final plane is a max projection that summarises the path. Right: performance of each model in the task (grey: random baseline). **(C)** As in (B), now for the task where the static goal changes between trials. **(D)** Representations and performance in the moving goal task. Left: the SR computes a value function that averages reward across time-within-trial. Middle: the STA takes into account the moving goal and computes a path that intercepts it. Right: performance of each agent. **(E)** Performance in the reward landscape task. All error bars indicate 1 standard deviation across 20 agents and environments (dots).

to intercept the goal at the correct point in space and time (Figure 3D). In contrast to the TD and SR agents, the STA can solve the moving goal task because the input to each subspace is the reward at that specific moment in time. The representation is therefore biased towards coinciding with the goal in both space and time, and the network dynamics relax to a future trajectory that correctly intercepts the goal (Figure 3D). The advantage of the STA is exacerbated in the more dynamic reward landscape task, where very different locations can be rewarded at different times (Figure 3E). These results suggest that different algorithms could contribute differently to decision making. TD learning could drive rapid decisions in environments with stable rewards; the SR would be more flexible when rewards change on intermediate timescales; and the STA could enable adaptive behaviour in environments with rapid changes. Finally, sequential search would facilitate slower planning in novel environments and could be guided by partial plans from a spacetime attractor.

STAs are efficient planners

We have seen that a handcrafted spacetime attractor can solve dynamic problems that simpler algorithms cannot. However, this requires many neurons to explicitly represent the future. We therefore investigated whether superior solutions exist to these 'PFC-like' problems by training a recurrent neural network (RNN) to solve them efficiently (Mante et al., 2013; Stroud et al., 2023). We will see that regularised RNNs solve dynamic planning problems with an STA-like algorithm, suggesting that it is efficient. This result also demonstrates that the STA is consistent with a prominent theory that PFC resembles a recurrent meta-learner (Wang et al., 2018). In this view, PFC learns connections over long timescales that implement adaptive behaviour on short timescales through recurrent dynamics. This is exactly how planning happens in the STA, which embeds task structure in the weights and uses recurrent dynamics for rapid decision making with different rewards. We extend the meta-learning theory of PFC by showing that an STA is the mechanism implemented in the dynamics of RNNs meta-trained on challenging planning tasks.

The three defining features that allow a spacetime attractor to plan via inference are: (i) it has an explicit representation of the future; (ii) the connectivity embeds a model of the world; and (iii) attractor dynamics infer the future representation. We will see that RNNs trained on the reward landscape task exhibit all of these properties. The RNNs were trained with supervised learning to generate optimal actions from inputs that indicated the current location and trial-specific reward structure (Figure S1; Methods; see Supplementary Note for a discussion of different modelling choices). The reward input was only provided during an initial 'planning phase' prior to the 'execution phase' of the task, and the loss function also penalised the magnitude of neural firing rates and network weights to encourage an efficient solution.

RNNs learn explicit future representations Spacetime attractors compute explicit representations of the future (Figure 4A-B). Critically, these representations generalise over trajectories. In other words, the representation in subspace δ depends only on the expected location in δ actions, and not on the rest of the trajectory or the history of the agent (Figure 4A, right). To confirm that the RNN learned such a representation, we trained linear decoders to predict the future from its hidden state. To ensure generalisation across trajectories, we trained the decoders while holding out each 'current location' and computed the test accuracy only from those held out locations. The RNN had an explicit representation of the future during both planning and execution (Figure 4C-D; Figure S2). In a spacetime attractor, the same subspace always encodes location δ actions into the future. The future representation of the RNN also exhibited such 'conveyor belt' dynamics (Figure 4E). The neural representation of the RNN is thus consistent with a spacetime attractor.

Intriguingly, RNNs trained on the simpler static and moving goal tasks did not reliably learn explicit future representations (Figure S3). Additionally, the RNN trained on the reward landscape task could solve these simpler tasks, while RNNs trained on the simpler tasks failed in the reward landscape task (Figure 4F). Alternative solutions therefore exist in the simpler tasks, and these solutions are favored by a pressure to be energetically efficient (Figure 4G). We identify a tradeoff between generality and efficiency, where the PFC-like solution is general but needs more neurons and synapses, while simpler algorithms can solve simpler tasks more efficiently. Unlike other theories of PFC such as value coding, this suggests a reason why PFC occupies such a large cortical territory.

RNNs learn a world model In a spacetime attractor, subspaces that represent expected future locations are connected according to the environment structure. Testing this prediction in the RNN requires interpretation of the network weights, which is challenging because the functions of individual neurons are unknown. We therefore project the weights into a coordinate system, where the axes are orthogonal directions in neural state space that predict each point in spacetime (Methods). While this does not perfectly recover the true subspaces in the handcrafted STA, it is a good estimate. The projected RNN weights can therefore be interpreted as interactions between representations of different points in spacetime (Figure 5A).

Remarkably, the recurrent weights between adjacent subspaces closely resembled the adjacency matrix of the environment (Figure 5B; Figure S4; mean $\pm$ std correlation of 0.91 ± 0.07 vs. 0.72 ± 0.06 for control environments). The RNN thus learned a world model explicitly in its recurrent weights. Consistent with a spacetime attractor, the $\delta = 0$ subspace received input indicating the current agent location, and subspace δ received input indicating the reward in δ actions (Figure 5C). The RNN also learned weaker connections between more dis-

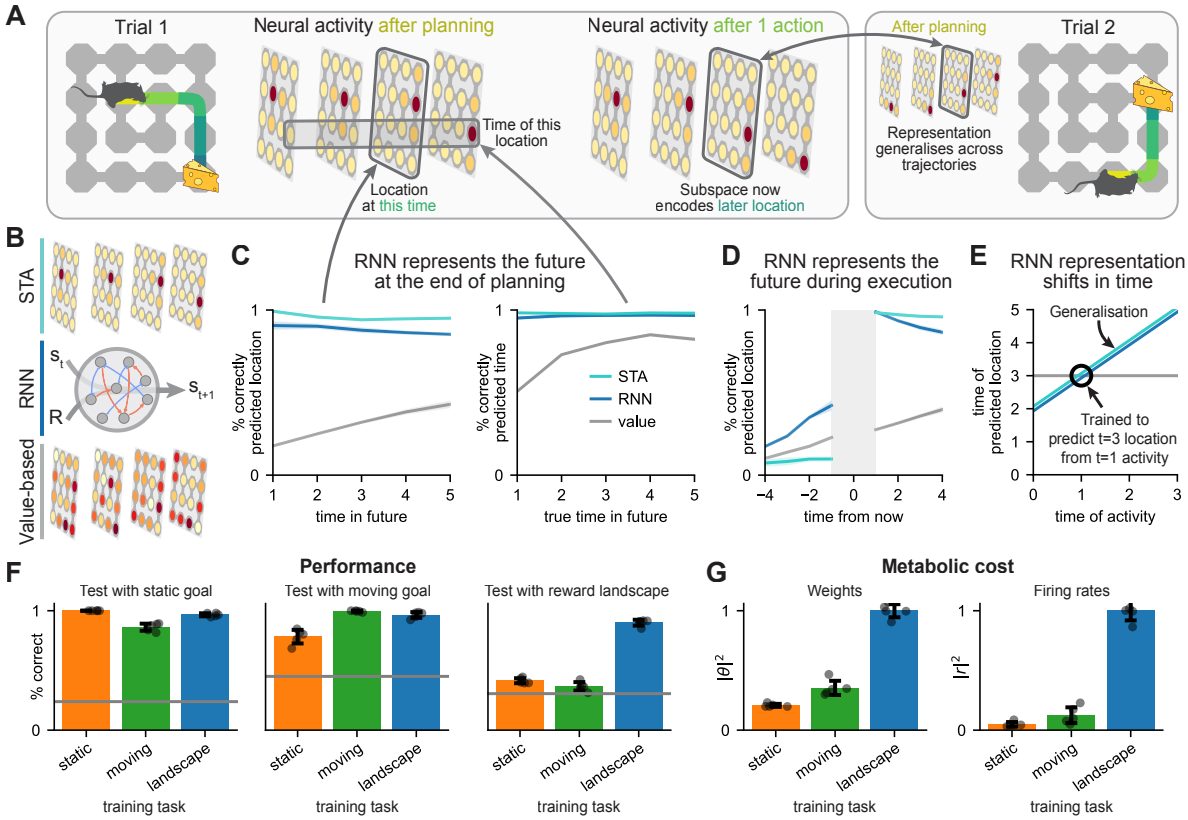

Figure 4: **RNNs learn spacetime representations in the reward landscape task. (A)** An STA infers the entire future during planning and represents the 'future-to-go' during execution (left). The representation in each subspace generalises across all trajectories passing through that location in spacetime (right). **(B)** We compare a spacetime attractor; an RNN trained on the reward landscape task; and an agent that computes an exact value function in space and time. The value-based agent computes an optimal policy from 'neural activity' containing (i) the value function, (ii) the agent location, and (iii) the time-within-trial (Methods). **(C)** Decoding accuracy at the end of planning for: left; agent location at each time in the future. Right; the time at which the agent will be at a given location, plotted as a function of the actual time the location was visited. Decoders were trained in crossvalidation across the current agent location (Methods). **(D)** Decoding accuracy during execution of location at each time in the past or future. **(E)** We trained a single decoder to predict location at time 3 from neural activity at time 1 (black circle). The same decoder predicted location at time $t + 2$ from neural activity at any other t, demonstrating 'conveyor belt' dynamics. **(F)** Performance in the static goal (left), moving goal (centre), and reward landscape (right) tasks for RNNs trained on either task (x-labels; colours). **(G)** Normalised parameter magnitudes of the three RNNs (left) and average firing rates in the static goal task (right). All error bars indicate 1 standard deviation across 5 RNNs (dots).

tant subspaces, and these connections reflected the environment structure (Figure 5D-E; Supplementary Note). In summary, the RNN learned all of the structural components that allow a spacetime attractor to infer future actions by integrating expected reward across time using a world model.

RNNs learn attractor dynamics In a spacetime attractor, the structured connections give rise to attractor dynamics with fixed points corresponding to explicit representations of desirable futures. To demonstrate similar attractor dynamics in the trained RNN, we artificially perturbed its neural activity at the end of planning. This resembles how attractor dynamics have been demonstrated in biological circuits (Inagaki et al., 2019; Kim et al., 2017; Vinograd et al., 2024). An attractor network should be robust to small perturbations, and the representation should be more sensitive to perturbations towards other attractor states than perturbations along random directions in neural state space. We analysed the RNN in a setting with two 'good' paths that were close in value (Figure 5F). Like the handcrafted STA, the RNN initially converged to a stable representation of the better path, which persisted in the presence of weak perturbations. Stronger perturbation of a single location on the alternate

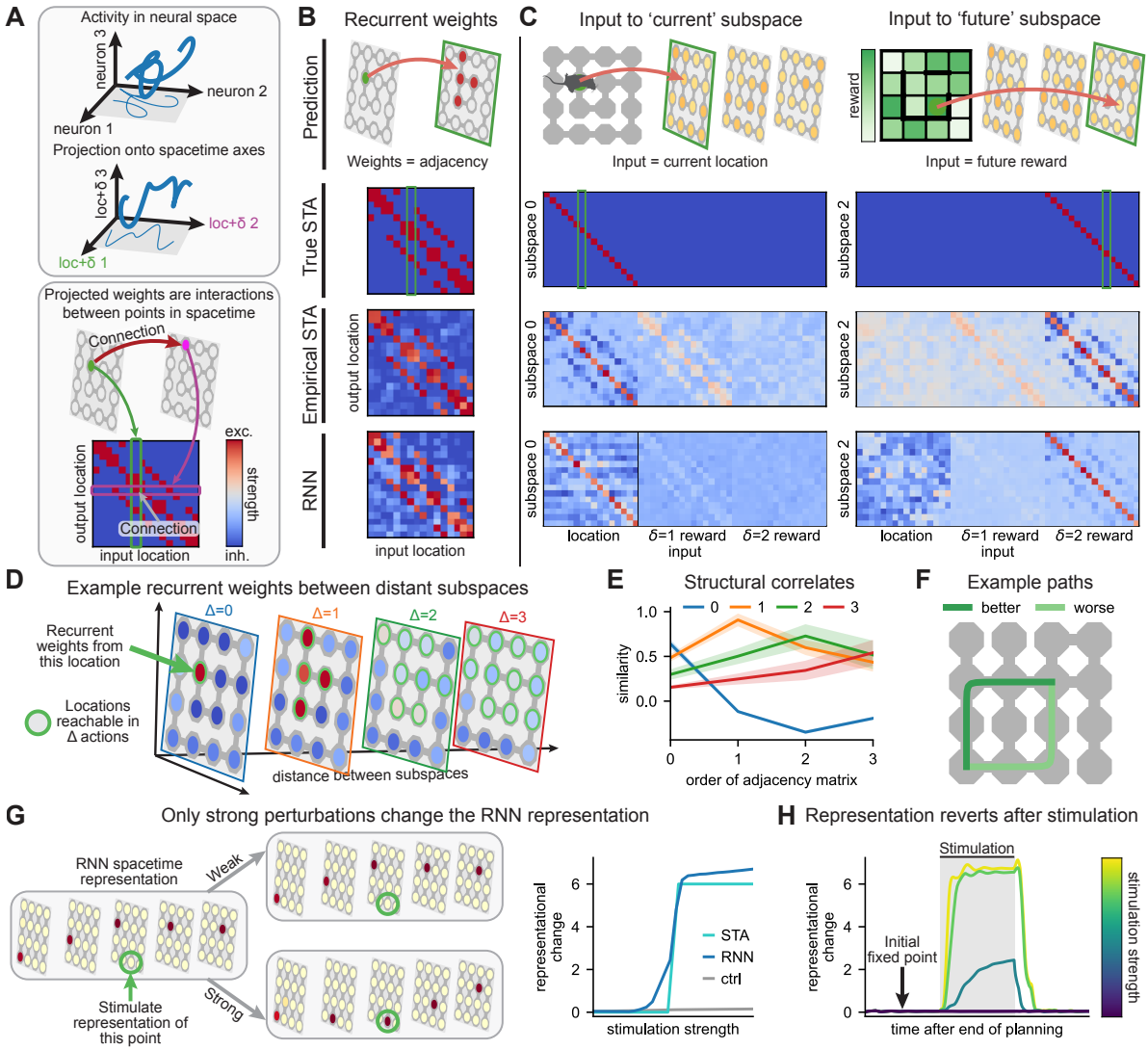

Figure 5: **RNNs learn a world model. (A)** The weights of RNNs trained on the reward landscape task are projected into an orthonormal coordinate system with axes that predict different points in spacetime (top). The projected weights are interactions between points in spacetime (bottom). **(B)** The average recurrent weights between subspaces separated by a single action resemble the environment adjacency matrix (bottom, 'RNN'). 'Empirical STA' is the same analysis performed on approximate subspaces estimated from neural activity in the handcrafted STA. 'True STA' indicates weights between the ground truth future-coding subspaces in the handcrafted STA. Green box in 'True STA' indicates weights between (i) the single location denoted by a green circle in 'Prediction', and (ii) all locations in the subspace indicated by a green square. **(C)** Input weights to the 'current' ($\delta = 0$; left) and a 'future' ($\delta = 2$; right) subspace. Consistent with an STA, the current subspace of the RNN receives location input, and future subspaces receive reward information. **(D)** Average recurrent weights between an example location (green arrow) and other locations at increasing time differences (Δ; planes). Locations in subspaces Δ apart are more strongly connected if they can be reached in Δ actions. Light green circles indicate these Δ^{th} order adjacency matrices ($\Delta = 0$ corresponds to the identity). **(E)** Correlation between (i) the average connectivity between subspaces separated by Δ actions (lines; legend), and (ii) different order adjacency matrices (x-axis). Shading indicates standard deviation across 5 RNNs. **(F)** Example environment with a high-value path and a lower-value path. **(G)** Weak stimulation does not affect the spacetime representation of the RNN, but strong stimulation switches it to the lower-value path. **(H)** Magnitude of representational change over time across stimulation strengths (Methods).

path caused a discrete switch to a representation of this entire path, including in subspaces not directly perturbed (Figure 5G). Perturbations of similar magnitude in random control directions had little effect (Figure 5G, grey). Unlike the handcrafted STA, the RNN representation relaxed back to the

original fixed point after removing the perturbation (Figure 5H; Figure S5), which suggests that the RNN learned an attractor landscape that is less susceptible to local minima.

Together, our analyses show that RNNs trained on a dynamic planning task learn to closely approximate a spacetime attractor. This was also true across variations in model architecture (Methods; Figure S6; Figure S7). Finally, RNNs with too few hidden units to learn a spacetime attractor failed to solve the task (Figure S8), suggesting that other solutions are not readily learned by gradient descent. The spacetime attractor is thus an efficient solution to dynamic planning problems, and it is consistent with a prominent theory that PFC can be understood as a recurrent meta-learner.

STAs can generalise across environments

Hardwiring the transition structure of the environment into the synapses of a spacetime attractor seems prohibitive, because it might prevent the network from generalising across different structures (Figure 6A). Here, we show that is not the case. RNNs trained in environments with changing structure still learn spacetime attractors. The network dynamics are adapted to each environment through input-mediated gating of a generic scaffold. This is a simple mechanism for rapid adaptation that preserves the ability to plan under changing rewards.

Representations adapt to the environment To study planning in environments with changing structure, we trained RNNs on a variant of the reward landscape task, where the transition structure was different on every trial (Methods). The structure was provided to the RNN as a binary input that indicated whether pairs of otherwise adjacent states were separated by a wall. Importantly, the network had to adapt to each new environment through its dynamics while keeping the weights fixed across environments (Wang et al., 2018; Jensen et al., 2024). The generalising RNNs performed nearly as well as networks trained in a single environment (Figure 6B), raising the question of how generalisation is achieved. The neural representation resembled a spacetime attractor in any single environment (Figure S9B). However, the effective connectivity between subspaces changed between environments. Remarkably, the connectivity reflected the transition structure of whichever environment the agent was currently in (Figure 6C-D). This indicates a general solution that allows recurrent networks to use attractor dynamics for planning in any similar environment, without requiring synaptic plasticity.

A model of generalisation We can understand how the network adapts to the environment at both the 'population level' and at a mechanistic level. At a population level, the directions in neural state space that represent each point in spacetime were more similar in different trials from the same environment than across trials from different environments (Figure 6E, left). This change allows the network to align its representation with the appropriate components of the connectivity matrix (Figure 6E, right). How is this achieved mechanistically? We hypothesised that the network explicitly represents future *transitions* rather than future locations. In other words, there are directions in neural state space that represent '*in δ actions from now, move from state s_i to state s_j*', which we denote τ_δ^{ij}. This is different from the vanilla spacetime attractor, which has a representation of future locations that is invariant to the subsequent state.

While it requires more neurons, explicitly representing future transitions allows the network to use input-mediated inhibition to adaptively gate off transitions that are not available in any particular environment. The learned structure of the remaining transitions is then used for planning (Figure 6F). Importantly, the scaffold only needs to include transitions that are possible in at least one environment. This model makes three predictions that we verified: (i) the network explicitly represents future transitions (Figure S9C), (ii) the representations of future transitions are connected by a world model (Figure 6G), and (iii) input indicating a wall between s_i and s_j specifically inhibits representations of future transitions between those states (Figure 6H). These analyses show that the spacetime attractor is an efficient planning algorithm in structurally changing environments, and they suggest a general mechanism for adaptive behaviour.

Discussion

We have taken inspiration from known cortical attractor dynamics and recent findings on prefrontal sequence representations (Figure 1) to develop a new theory of planning. This spacetime attractor model uses different neural subspaces to represent the expected location of an agent at different times in the future. The subspaces are connected according to the structure of the environment, which allows the network to infer optimal future trajectories (Figure 2). The spacetime attractor can solve problems with dynamic rewards, which PFC is important for and simpler algorithms struggle with (Figure 3). It solves these problems using inter-

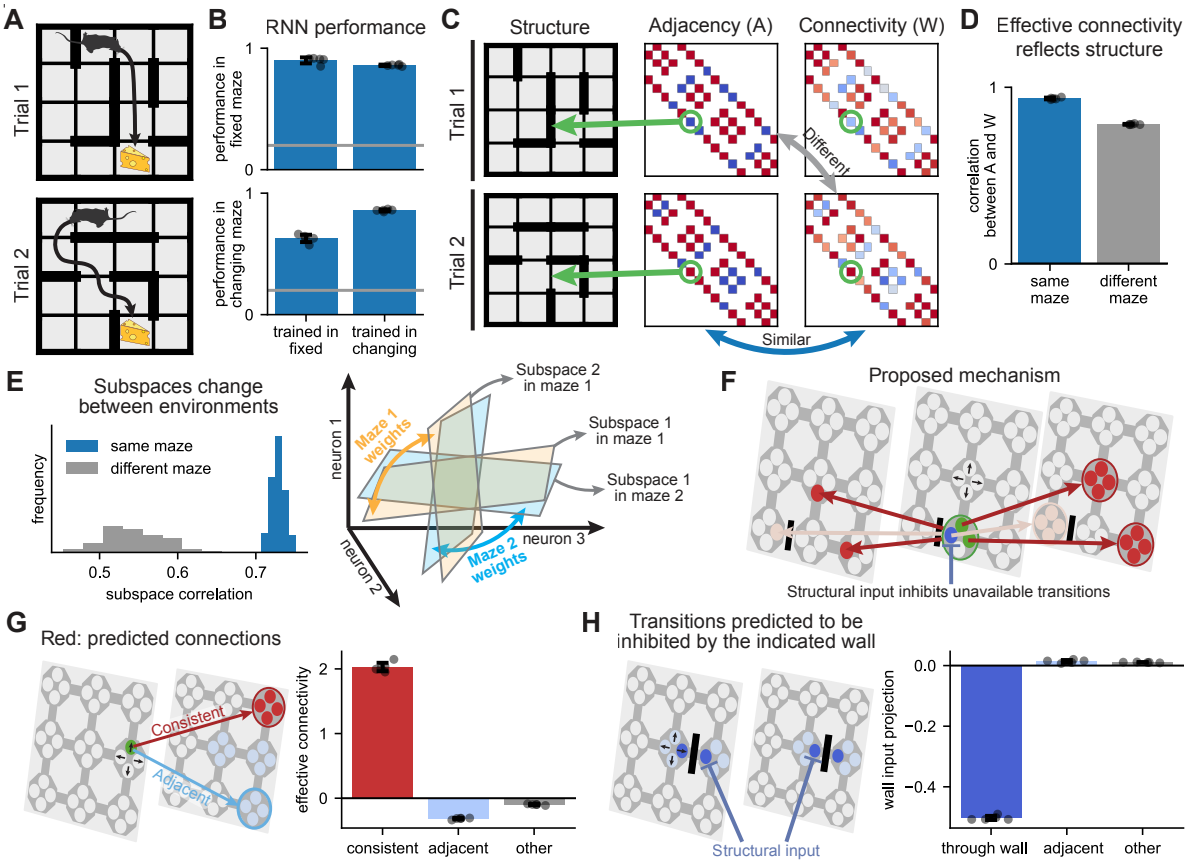

Figure 6: **Spacetime attractors can adapt to changing structure.** **(A)** Changing structure requires adaptation. **(B)** Performance of RNNs trained on the reward landscape task in a single maze or with a different structure on every trial, when evaluated in either a single maze (top) or across changing mazes (bottom). **(C)** Two example mazes (left) and their corresponding adjacency matrices (centre). The effective connectivity between adjacent subspaces in a single RNN (right) reflects the structure of each maze. **(D)** Average correlation between subspace connectivity in a maze and the structure of either the same (blue) or a random (grey) maze. **(E)** The spacetime representation is more similar across distinct trials from the same maze than trials from different mazes (left). By using slightly different subspaces in different environments, the network can match their connectivity to the environment structure (right; orange vs. blue). **(F)** Putative mechanism for structural generalisation. Instead of representing each future *location*, neurons encode expected future *transitions* (black arrows in example state). Each transition τ_δ^{ij} (green) connects to transitions that can follow ($\tau_{\delta+1}^{jk}$) or precede ($\tau_{\delta-1}^{li}$) it (red arrows). Structural input to PFC inhibits transitions that are not available in a given environment (blue), preventing planning between states that would otherwise be connected (light red arrows). **(G)** Effective connectivity between directions in neural state space that encode 'consistent' consecutive future transitions (τ_δ^{ij} and $\tau_{\delta+1}^{jk}$; green to red), 'adjacent' transitions (green to blue), or any other transitions (green to grey). **(H)** Projection of the input from wall w_{ij} onto representations of future transitions *through* the wall (τ^{ij}; dark blue), transitions to other adjacent states (τ^{ik}; light blue), or any other transitions (grey). All error bars indicate 1 standard deviation across 5 RNNs (dots).

nal representations that closely resemble prefrontal representations during sequence working memory, thereby unifying working memory and planning in PFC. RNNs trained to solve dynamic planning tasks learn to implement the spacetime attractor algorithm in their internal dynamics, suggesting that it is an efficient solution (Figure 4; Figure 5). Finally, spacetime attractors can generalise across environments with different transition structures through rapid gating of a general scaffold to reflect each particular environment (Figure 6).

Experimental predictions The spacetime attractor is inspired by data and makes precise predictions that can be tested in future experiments:

- After planning a behavioural sequence, different subspaces of PFC activity should represent distinct steps of the plan.

- Optogenetic activation of neurons in a future subspace should bias representations and behaviour towards the stimulated state at a delay corresponding to the stimulated subspace.
- The effective connectivity between subspaces should reflect the structure of the environment. This could be tested in noise correlations across neurons or by explicit holographic stimulation.
- Different patterns of neurons should represent the future in environments with different transition structures.
- The neurons active in each environment should be connected according to its structure. Inputs that mediate this gating could come through sensory cortex in observable environments or hippocampus when the structure is learned.

The first two predictions would also be true of independent subspaces used to store sequence memories (Xie et al., 2022; El-Gaby et al., 2023), while the last three are unique predictions of a structured network that plans behavioural sequences.

Decision making across the brain Our comparisons of different models across different tasks suggest complementary contributions of different brain regions to decision making. Striatum is thought to implement temporal difference learning (Schultz et al., 1997), which facilitates rapid responses in stable environments. Hippocampus embeds the structure of the world, but it is thought to either average representations over future timesteps in a successor representation (Stachenfeld et al., 2017) or represent one state at a time in a sequence (George et al., 2021; Whittington et al., 2020; Jensen et al., 2024). This provides an efficient solution to problems with consistent spatial structure and rewards that change on intermediate timescales. A spacetime attractor in frontal cortex facilitates adaptive behaviour in dynamic environments with a familiar structural scaffold that has been embedded in PFC. Finally, explicit search would facilitate slower planning in novel environments and could be guided by partial plans from a spacetime attractor. While spacetime attractors are particularly useful in dynamic environments, they can also solve simpler problems. It is therefore possible that animals have developed a spacetime attractor because they *sometimes* need it for planning, and then reuse it in simpler tasks. However, it is also plausible that many laboratory behaviours instead engage simpler algorithms in the basal ganglia or hippocampus. If that is the case, richer spacetime problems may be needed to query the representations used for planning in prefrontal cortex.

Interactions with other planning algorithms Planning via inference in a spacetime attractor differs from most models of planning in cognitive science. In particular, the spacetime attractor posits that planning can happen via *recognition*, where the sequence of steps needed to reach some desired state is directly inferred. In contrast, many studies of human planning focus on explicit search, where different paths are simulated and evaluated sequentially. We suggest that these two processes coexist, with spacetime attractor dynamics facilitating rapid planning as inference in familiar environments, where the structure has been embedded in prefrontal connections. The STA could also help focus explicit search towards putative high-value paths and evaluate the utility of different paths. This could happen through interactions with hippocampal activity sequences that have been proposed to facilitate planning (either replays or theta sequences; Foster, 2017; Widloski and Foster, 2022; Jensen et al., 2024). In particular, Jensen et al. (2024) suggest that PFC biases which sequences are replayed in hippocampus. PFC would then update its representation to make high-reward sequences more likely in an iterative policy improvement process. Hippocampal replay has also been proposed to implement a 'DYNA' algorithm, where past experience is used to learn a model-free policy during rest or sleep (Mattar and Daw, 2018). This is complementary to both the spacetime attractor and explicit search. DYNA facilitates rapid decision making when the optimal policy remains stable across time, while decision-time planning allows adaptation to changing environments.

Learning a spacetime attractor In the handcrafted spacetime attractor, we embedded copies of the environment adjacency matrix in the connections between every pair of consecutive subspaces. The RNN analyses show that such structure can be learned from repeated experience. However, learning copies of the same parameters independently for each pair of subspaces is inefficient. It would be more efficient to store a cache of experienced trajectories that can be used to learn all of the parameters. Hippocampal replay has been proposed to build cognitive maps via such interactions with prefrontal cortex (Bakermans et al., 2023; Ou et al., 2025). In particular, Ou et al. (2025) suggest that hippocampal replay consolidates structural information from hippocampus into cortex. Replay from hippocampus to PFC during sleep or rest could therefore provide a mechanism for learning all of the STA parameters from the same data.

The state space of planning We have assumed that planning happens at the level of individual locations in the environment. However, humans often plan in more abstract spaces, which improves efficiency by reducing the required planning depth. We also plan hierarchically by first computing an abstract plan that can be refined in increasing levels of detail (Eckstein and Collins, 2020). Multiple spacetime attractors operating at different levels of abstraction can be coupled to implement such a hierarchical planner. One 'abstract' STA would compute a high-level plan. A second STA would treat the next abstract state as a goal and compute a detailed plan to get there. Stacking spacetime attractors in this way allows inference of plans that are exponentially long in the depth of the hierarchy, and therefore the number of neurons, in contrast to the linear scaling of a non-hierarchical STA. However, learning appropriate abstractions and embedding them in an STA remains an unsolved challenge.

Bidirectional environment interactions We have focused on environments that evolve independently of the agent. The rewards and structure of the environment can therefore be estimated up front and provided as inputs to a spacetime attractor in PFC. However, many problems involve structure that depends on the behaviour of an agent. One example is 'key-door' problems, where moving through a locked door becomes possible only after picking up the key. We posit that such problems can be solved by structural gating of a learned scaffold, similar to the STA that generalises across environment structures (Figure 6). The structural gating would no longer be an external input, but instead a function of the spacetime representation itself. For example, activating a cell that represents 'being at the key' could disinhibit 'transitioning through the door' at a later time.

Another important example of bidirectional interactions is social behaviours, where agents can change their plans in response to each other. A system of two spacetime attractors could simulate the joint dynamics of two such agents. The first STA 'A' infers future behaviour for agent A. The reward input would be the output of a different STA 'B' that predicts the behaviour of another agent. The reward for STA B would itself be an output of STA A, which couples the predicted behaviour of the two agents. The combined system relaxes to a fixed point where the behaviour of A is optimal given the predicted behaviour of B and vice versa – a putative neural implementation of 'theory of mind'.

Outlook We have developed a new theoretical framework for planning in prefrontal cortex. It builds on recently discovered prefrontal working memory representations and known attractor dynamics in other neural circuits. The spacetime attractor extends these principles to prospective behaviours in complex and changing environments – a setting that has previously eluded mechanistic circuit models. Existing data does not allow us to test these ideas explicitly. Instead, we have provided a series of concrete predictions for future experiments. We hope this will inspire new work in both experimental and computational neuroscience.

Author contributions

KTJ and TEJB developed the conceptual framework with input from TA, PD, MSM, SR, and AB. KTJ ran the simulations and performed the analyses. KTJ made the figures with help from MSM, SR, and AB. KTJ and TEJB wrote the manuscript with input from all authors.

Code availability

Code is available at github.com/KrisJensen/pysta. This includes code for running example models and reproducing all results from the paper.

Acknowledgments

We are grateful to Diksha Gupta, Mohamady El-Gaby, Will Dorrell, and Tom Mrsic-Flogel for helpful feedback on the manuscript. This work was supported by a Wellcome Principal Research Fellowship (219525/Z/19/Z; TEJB, KTJ, AB); the Gatsby Initiative for Brain Development and Psychiatry (GAT3955; TEJB); the Jean Francois and Marie-Laure de Clermont Tonerre Foundation (TEJB); a Wellcome Trust Career Development Award (225926/Z/22/Z; TA); the Oxford Clarendon Fund (PD); the Human Frontier Science Program (LT0040/2024-L; SR); EMBO (ALTF 651-2023; SR); and a FYSSEN postdoctoral study grant (MSM). KTJ, TEJB, MSM, and SR were also supported by the Sainsbury Wellcome Centre's core provided by Wellcome (219627/Z/19/Z) and the Gatsby Charitable Foundation (GAT3755).

References

Bakermans, J. J., Warren, J., Whittington, J. C., and Behrens, T. E. (2023). Constructing future behaviour in the hippocampal formation through composition and replay. *bioRxiv*, pages 2023–04.

Ben-Yishai, R., Bar-Or, R. L., and Sompolinsky,

H. (1995). Theory of orientation tuning in visual cortex. *Proceedings of the National Academy of Sciences*, 92(9):3844–3848.

Blanco-Pozo, M., Akam, T., and Walton, M. E. (2024). Dopamine-independent effect of rewards on choices through hidden-state inference. *Nature Neuroscience*, 27(2):286–297.

Botvinick, M. and Toussaint, M. (2012). Planning as inference. *Trends in cognitive sciences*, 16(10):485–488.

Botvinick, M. M. and Plaut, D. C. (2006). Short-term memory for serial order: a recurrent neural network model. *Psychological review*, 113(2):201.

Burak, Y. and Fiete, I. R. (2009). Accurate path integration in continuous attractor network models of grid cells. *PLoS computational biology*, 5(2):e1000291.

Burgess, P. W. and Wu, H. (2013). Rostral prefrontal cortex (Brodmann area 10). *Principles of frontal lobe function*, pages 524–544.

Callaway, F., van Opheusden, B., Gul, S., Das, P., Krueger, P. M., Griffiths, T. L., and Lieder, F. (2022). Rational use of cognitive resources in human planning. *Nature Human Behaviour*, 6(8):1112–1125.

Carlesimo, G. A., di Paola, M., Fadda, L., Caltagirone, C., and Costa, A. (2014). Prospective memory impairment and executive dysfunction in prefrontal lobe damaged patients: is there a causal relationship? *Behavioural neurology*, 2014(1):168496.

Chaudhuri, R., Gerçek, B., Pandey, B., Peyrache, A., and Fiete, I. (2019). The intrinsic attractor manifold and population dynamics of a canonical cognitive circuit across waking and sleep. *Nature neuroscience*, 22(9):1512–1520.

Chen, J., Zhang, C., Hu, P., Min, B., and Wang, L. (2024). Flexible control of sequence working memory in the macaque frontal cortex. *Neuron*.

Clark, D. G., Abbott, L., and Sompolinsky, H. (2025). Symmetries and continuous attractors in disordered neural circuits. *bioRxiv*, pages 2025–01.

Dayan, P. (1993). Improving generalization for temporal difference learning: The successor representation. *Neural computation*, 5(4):613–624.

Dorrell, W., Hsu, K., Hollingsworth, L., Lee, J. H., Wu, J., Finn, C., Latham, P. E., Behrens, T. E., and Whittington, J. C. (2024). Don't cut corners: Exact conditions for modularity in biologically inspired representations. *arXiv preprint arXiv:2410.06232*.

Eckstein, M. K. and Collins, A. G. (2020). Computational evidence for hierarchically structured reinforcement learning in humans. *Proceedings of the National Academy of Sciences*, 117(47):29381–29389.

El-Gaby, M., Harris, A. L., Whittington, J. C., Dorrell, W., Bhomick, A., Walton, M. W., Akam, T., and Behrens, T. E. (2023). A cellular basis for mapping behavioural structure. *bioRxiv*, pages 2023–11.

Foster, D. J. (2017). Replay comes of age. *Annu. Rev. Neurosci*, 40(581-602):9.

Fuhs, M. C. and Touretzky, D. S. (2006). A spin glass model of path integration in rat medial entorhinal cortex. *Journal of Neuroscience*, 26(16):4266–4276.

George, D., Rikhye, R. V., Gothoskar, N., Guntupalli, J. S., Dedieu, A., and Lázaro-Gredilla, M. (2021). Clone-structured graph representations enable flexible learning and vicarious evaluation of cognitive maps. *Nature communications*, 12(1):2392.

Hafting, T., Fyhn, M., Molden, S., Moser, M.-B., and Moser, E. I. (2005). Microstructure of a spatial map in the entorhinal cortex. *Nature*, 436(7052):801–806.

Iacaruso, M. F., Gasler, I. T., and Hofer, S. B. (2017). Synaptic organization of visual space in primary visual cortex. *Nature*, 547(7664):449–452.

Inagaki, H. K., Fontolan, L., Romani, S., and Svoboda, K. (2019). Discrete attractor dynamics underlies persistent activity in the frontal cortex. *Nature*, 566(7743):212–217.

Jensen, K. T. (2023). An introduction to reinforcement learning for neuroscience. *arXiv preprint arXiv:2311.07315*.

Jensen, K. T., Hennequin, G., and Mattar, M. G. (2024). A recurrent network model of planning explains hippocampal replay and human behavior. *Nature neuroscience*, 27(7):1340–1348.

Kim, S. S., Hermundstad, A. M., Romani, S., Abbott, L., and Jayaraman, V. (2019). Generation of stable heading representations in diverse visual scenes. *Nature*, 576(7785):126–131.

Kim, S. S., Rouault, H., Druckmann, S., and Jayaraman, V. (2017). Ring attractor dynamics in the drosophila central brain. *Science*, 356(6340):849–853.

Kingma, D. P. and Ba, J. (2015). Adam: A method for stochastic optimization. In Bengio, Y. and LeCun, Y., editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Ko, H., Hofer, S. B., Pichler, B., Buchanan, K. A., Sjöström, P. J., and Mrsic-Flogel, T. D. (2011). Functional specificity of local synaptic connections in neocortical networks. *Nature*, 473(7345):87–91.

Lee, T. S. and Nguyen, M. (2001). Dynamics of subjective contour formation in the early visual cortex. *Proceedings of the National Academy of Sciences*, 98(4):1907–1911.

Levine, S. (2018). Reinforcement learning and control as probabilistic inference: Tutorial and review. *arXiv preprint arXiv:1805.00909*.

Mante, V., Sussillo, D., Shenoy, K. V., and Newsome, W. T. (2013). Context-dependent computation by recurrent dynamics in prefrontal cortex. *nature*, 503(7474):78–84.

Mattar, M. G. and Daw, N. D. (2018). Prioritized memory access explains planning and hippocampal replay. *Nature neuroscience*, 21(11):1609–1617.

McNaughton, B. L., Battaglia, F. P., Jensen, O., Moser, E. I., and Moser, M.-B. (2006). Path integration and the neural basis of the'cognitive map'. *Nature Reviews Neuroscience*, 7(8):663–678.

Ou, J., Qu, Y., Xu, Y., Xiao, Z., Behrens, T., and Liu, Y. (2025). Replay builds an efficient cognitive map offline to avoid computation online. *bioRxiv*, pages 2025–01.

Ouchi, A. and Fujisawa, S. (2024). Predictive grid coding in the medial entorhinal cortex. *Science*, 385(6710):776–784.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., et al. (2011). Scikit-learn: Machine learning in python. *the Journal of machine Learning research*, 12:2825–2830.

Ruff, D. A., Markman, S. K., Kim, J. Z., and Cohen, M. R. (2025). Linking neural population formatting to function. *bioRxiv*.

Schrittwieser, J., Antonoglou, I., Hubert, T., Simonyan, K., Sifre, L., Schmitt, S., Guez, A., Lockhart, E., Hassabis, D., Graepel, T., et al. (2020). Mastering Atari, Go, chess and shogi by planning with a learned model. *Nature*, 588(7839):604–609.

Schultz, W., Dayan, P., and Montague, P. R. (1997). A neural substrate of prediction and reward. *Science*, 275(5306):1593–1599.

Shallice, T. and Burgess, P. W. (1991). Deficits in strategy application following frontal lobe damage in man. *Brain*, 114(2):727–741.

Shin, H., Ogando, M. B., Abdeladim, L., Durand, S., Belski, H., Cabasco, H., Loefler, H., Bawany, A., Hardcastle, B., Wilkes, J., et al. (2023). Recurrent pattern completion drives the neocortical representation of sensory inference. *bioRxiv*.

Skaggs, W., Knierim, J., Kudrimoti, H., and McNaughton, B. (1994). A model of the neural basis of the rat's sense of direction. *Advances in neural information processing systems*, 7.

Stachenfeld, K. L., Botvinick, M. M., and Gershman, S. J. (2017). The hippocampus as a predictive map. *Nature neuroscience*, 20(11):1643–1653.

Stroud, J. P., Watanabe, K., Suzuki, T., Stokes, M. G., and Lengyel, M. (2023). Optimal information loading into working memory explains dynamic coding in the prefrontal cortex. *Proceedings of the National Academy of Sciences*, 120(48):e2307991120.

Sutton, R. S. (1988). Learning to predict by the methods of temporal differences. *Machine learning*, 3:9–44.

Tian, Z., Chen, J., Zhang, C., Min, B., Xu, B., and Wang, L. (2024). Mental programming of spatial sequences in working memory in the macaque frontal cortex. *Science*, 385(6716):eadp6091.

Turner-Evans, D., Wegener, S., Rouault, H., Franconville, R., Wolff, T., Seelig, J. D., Druckmann, S., and Jayaraman, V. (2017). Angular velocity integration in a fly heading circuit. *Elife*, 6:e23496.

Turner-Evans, D. B., Jensen, K. T., Ali, S., Paterson, T., Sheridan, A., Ray, R. P., Wolff, T., Lauritzen, J. S., Rubin, G. M., Bock, D. D., et al. (2020). The neuroanatomical ultrastructure and function of a biological ring attractor. *Neuron*, 108(1):145–163.

Vinograd, A., Nair, A., Kim, J. H., Linderman, S. W., and Anderson, D. J. (2024). Causal evidence of a line attractor encoding an affective state. *Nature*, 634(8035):910–918.

Volle, E., Gonen-Yaacovi, G., de Lacy Costello, A., Gilbert, S. J., and Burgess, P. W. (2011). The role of rostral prefrontal cortex in prospective memory: a voxel-based lesion study. *Neuropsychologia*, 49(8):2185–2198.

Walton, M. E., Behrens, T. E., Buckley, M. J., Rudebeck, P. H., and Rushworth, M. F. (2010). Separable learning systems in the macaque brain and the role of orbitofrontal cortex in contingent learning. *Neuron*, 65(6):927–939.

Wang, J. X., Kurth-Nelson, Z., Kumaran, D., Tirumala, D., Soyer, H., Leibo, J. Z., Hassabis, D., and Botvinick, M. (2018). Prefrontal cortex as a meta-reinforcement learning system. *Nature neuroscience*, 21(6):860–868.

Whittington, J. C., Dorrell, W., Behrens, T. E., Ganguli, S., and El-Gaby, M. (2023). On prefrontal working memory and hippocampal episodic memory: Unifying memories stored in weights and activation slots. *bioRxiv*, pages 2023–11.

Whittington, J. C., Muller, T. H., Mark, S., Chen, G., Barry, C., Burgess, N., and Behrens, T. E. (2020). The Tolman-Eichenbaum machine: Unifying space and relational memory through generalization in the hippocampal formation. *Cell*, 183(5):1249–1263.

Widloski, J. and Foster, D. J. (2022). Flexible rerouting of hippocampal replay sequences around changing barriers in the absence of global place field remapping. *Neuron*, 110(9):1547–1558.

Xie, Y., Hu, P., Li, J., Chen, J., Song, W., Wang, X.-J., Yang, T., Dehaene, S., Tang, S., Min, B., et al. (2022). Geometry of sequence working memory in macaque prefrontal cortex. *Science*, 375(6581):632–639.

Zhang, K. (1996). Representation of spatial orientation by the intrinsic dynamics of the head-direction cell ensemble: a theory. *Journal of Neuroscience*, 16(6):2112–2126.

Zheng, J., Guimaraes, R., Hu, J. Y., Perona, P., and Meister, M. (2024). Mice in the manhattan maze: Rapid learning, flexible routing and generalization, with and without cortex. *Cognitive Computational Neuroscience*.

Zintgraf, L., Shiarlis, K., Igl, M., Schulze, S., Gal, Y., Hofmann, K., and Whiteson, S. (2019). VariBAD: A very good method for Bayes-adaptive deep RL via meta-learning. *arXiv preprint arXiv:1910.08348*.

Methods

Tasks

We used three different classes of tasks to train and compare models in this work – the 'static goal' task, the 'moving goal' task, and the 'reward landscape' task. All tasks had agents navigate mazes on a 4x4 grid, with walls preventing transitions between some pairs of otherwise adjacent states. The wall configurations defining the mazes were sampled as described by Jensen et al. (2024). For most analyses, the wall configuration remained fixed across all trials, and only the reward function changed. For the analyses in Figure 6, the wall configuration changed across trials, and the agent had to adapt to the new transition structure through its recurrent network dynamics. In all tasks, the agent location was sampled randomly at the beginning of each trial. At the beginning of each trial, the agent location and reward function were 'frozen' for 5-7 (randomly sampled on each trial) iterations of the environment. This constituted an initial 'planning' phase that was followed by an 'execution' phase, where (i) the agent took actions that changed its location, and (ii) the reward function updated as described in more detail below. For all analyses, the full reward function was provided to the agent during the initial planning period. For the handcrafted STA and the RNN in Figure S6, the input provided reward information that was relative in time during execution – it indicated what the reward would be at a given location *in δ steps* rather than *at time t*. The reward input was zero for all δ that corresponded to time points beyond the end of the trial. For all other RNN analyses, the reward input was set to zero during execution.

Moving goal task

In this task, a full goal 'trajectory' was sampled on each trial. The goal trajectory was sampled as a random walk that could only turn around if it reached a dead end. The start location of the agent was restricted to not coincide with the start location of the goal. The trial terminated when the agent was at the same location as the goal at a given moment in time, or after a maximum of six actions. The reward input to the agent, $\boldsymbol{R} \in \mathbb{R}^{T \times N}$, was a matrix indicating the location of the goal at every point in the future. The reward input consisted of '+0.6' for the goal location and '-0.6' for all other locations at each moment in time. In other words, $R_{\delta i}$ was '+0.6' if the goal would be at location s_i δ steps into the future, and -0.6 otherwise.

Static goal task

This task was identical to the moving goal task, except that the goal remained stationary for the duration of each trial. The static goal task was implement in two different variants – one where the goal remained fixed across all trials, and one where the goal was resampled at random in every new trial. The handcrafted models were evaluated in both of these tasks, while the RNNs were only trained with a goal that changed between trials.

Reward landscape task

In this task, every element $R(t, s)$ of the reward function $\mathcal{R}$ was sampled uniformly and independently between -1 and +1. The complete future reward structure was provided as an input to the agent as described above. Every trial finished when the agent had taken six actions.

Quantification of performance

For most performance comparisons, we computed the probability of choosing the optimal first action in each trial. We define the optimal first action as the first action of the trajectory that maximises cumulative reward over the entire trial. We use this metric rather than the actual cumulative reward for two reasons. (i) We are interested in the process of planning, whereby an agent balances immediate and long-term reward. This is most challenging for early actions, while the greedy policy is optimal for the last action. (ii) The probability of choosing an optimal action is readily interpretable as a number between 0 and 1. All results were qualitatively similar if we instead used the probability of choosing an optimal action averaged over the entire trial as a performance metric, or if we used the average empirical reward.

Handcrafted models

Here we provide an overview of the handcrafted STA, TD, and SR agents used in the paper.

Spacetime attractor model

The spacetime attractor is a recurrent neural network with an exponential nonlinearity and within-subspace normalisation. We define $z_{\delta i}$ as the 'potential' and $r_{\delta i}$ as the 'firing rate' of a neuron that represents 'being at location s_i in δ actions'. The spacetime attractor then has the following network dynamics:

$$\tau \dot{z}_{\delta i} = -z_{\delta i} + \hat{R}_{\delta i} + \max\left(\epsilon, \log \sum_j A_{ij} r_{\delta-1,j}\right) + \max\left(\epsilon, \log \sum_k A_{ki} r_{\delta+1,k}\right) + \eta \tag{1}$$

$$z_{\delta i} = \max\left(\epsilon, \; z_{\delta i} - \log \sum_j r_{\delta j}\right) \tag{2}$$

$$r_{\delta i} = e^{z_{\delta i}}. \tag{3}$$

Here, $\tau = 50$ iterations is the time constant of the dynamics, $\epsilon = -100$ is a small constant that thresholds the maximum inhibition between and within subspaces, and $\eta \sim \mathcal{N}(0, \sigma = 0.1)$ is white noise added to the network dynamics. A_{ij} are weights corresponding to the adjacency matrix, which indicates whether s_i can be reached from s_j in one action. $\hat{R}_{\delta i}$ is an input that reflects the normalised reward available at location s_i in δ actions. In particular, $e^{\hat{R}_{\delta i}} \propto e^{\beta R_{\delta i}}$, where $R_{\delta i}$ is the trial-specific reward described above, and we set $\beta = 9$. To provide an input indicating the current agent location, we set $R_{0i} = 20$ if s_i is the current location and 0 otherwise. Since the activity is explicitly normalised within each subspace, the resulting $r_{\delta i}$ can be interpreted as a distribution over desired locations in δ actions. When simulating the behaviour of the STA, we took the policy of the agent to be greedy in the representation in subspace $\delta = 1$ of locations accessible from the current state, $a = \text{argmax}_{i \in \mathcal{N}(s)} [r_{1i}]$. We also added a small amount of noise $\eta_{ij} \sim \mathcal{U}(-0.025, -0.015)$ to each element of the weight matrix instead of using the exact adjacency matrix. Noise was added to ensure robustness, and it was restricted to be negative by adding a bias term that prevents representations from 'teleporting' between locations (Supplementary Note). An additional 'feedforward component' in the form of the identity matrix was added to the connectivity between subspace δ and $\delta - 1$ during the first 100 network iterations (2 time constants) after every action. This is inspired by the 'update neurons' of the fruit fly head direction circuit (Turner-Evans et al., 2017) and stabilises the 'conveyor belt' dynamics, but it is not necessary for any of the main results in the paper (Supplementary Note). We ran the STA dynamics for 400 network iterations before each action to ensure convergence to a stable representation.

Baselines

Here we provide details of the temporal difference and successor representation baselines that we compare the spacetime attractor to. For further details, we refer to Jensen (2023). Common to both of these frameworks is that they explicitly estimate the 'value function' under some policy π:

$$V^{\pi}(s) = \mathbb{E}_{\tau \sim p_{\pi}(\tau)} \left[\sum_{t' \geq t} \gamma^{t'-t} r_{t'} | s_t = s \right]. \tag{4}$$

Here, $\mathbb{E}_{\tau \sim p_{\pi}(\tau)}[\cdot]$ indicates an expectation taken over trajectories τ resulting from the agent following π. The 'TD' and 'SR' heatmaps in Figure 3 show the computed value functions. In both cases, we take the policy of the agent to be greedy in the value function evaluated at all locations accessible from the current state.

Temporal difference learning We implement vanilla temporal difference learning, which computes a value function by iteratively applying the update

$$\Delta V(s_t) = \alpha(-V(s_t) + r_t + \gamma V(s_{t+1})). \tag{5}$$

We set the temporal discount factor to $\gamma = 1$, since we are interested in maximising the non-discounted cumulative reward. We allowed the TD agent to interact with the environment for 4,000 trials with a learning rate of $\alpha = 0.05$ before analysing its performance.

Successor representation In the successor representation formalism, the value function is decomposed as

$$V^\pi(s) = \mathbb{E}_\pi \left[\sum_{t=0} \gamma^t r_t | s_0 = s \right] \tag{6}$$

$$= \sum_{t=0} \gamma^t \sum_{s'} p_\pi(s_t = s' | s_0 = s) r(s') \tag{7}$$

$$= \boldsymbol{r}^T \boldsymbol{m}_s^\pi. \tag{8}$$

Here, $\boldsymbol{r}$ is a vector of the average reward associated with each state, and $\boldsymbol{m}_s^\pi$ is a vector of the expected discounted future occupancy of state s' if the agent starts in state s and follows policy π:

$$M_{ss'}^\pi = \sum_{t=0} \gamma^t p_\pi(s_t = s' | s_0 = s). \tag{9}$$

The full matrix $\boldsymbol{M}^\pi$, constructed from stacking the $\boldsymbol{m}_s^\pi$ corresponding to all states s, is denoted the 'successor matrix', and it allows us to write down a vector of expected rewards from all states as

$$\boldsymbol{v}^\pi = \boldsymbol{M}^\pi \boldsymbol{r}. \tag{10}$$

$\boldsymbol{M}^\pi$ can be learned using a TD-like algorithm as above. However, since the STA algorithm also has explicit access to the transition structure of the environment, we simply computed the exact successor matrix as the geometric series $\boldsymbol{M}^\pi = \boldsymbol{I} + \gamma \boldsymbol{T}^\pi + \gamma^2 (\boldsymbol{T}^\pi)^2 + \ldots = (\boldsymbol{I} - \gamma \boldsymbol{T}^\pi)^{-1}$ with $\gamma = 0.95$. When computing the successor matrix, we take π to be the diffusion policy, and $\boldsymbol{T}^\pi$ is therefore the diffusion matrix. This is similar to how the adjacency matrix serves as a global structural prior in the STA.

Spacetime value agent In Figure 4, we compare representations to an agent that computes a value function in a state space consisting of space *and* time. This agent uses dynamic programming to compute the value of being at location s at time t under an optimal policy for every combination of s and t:

$$V(T, s) = R(T, s) \tag{11}$$

$$V(t, s) = R(t, s) + \max_{s' \in \mathcal{N}(s)} V(t+1, s'). \tag{12}$$

At time t, the optimal policy is then greedy in the value of accessible locations at time $t+1$. For the decoding analyses in Figure 4, we took the 'neural representation' to be the concatenation of (i) the flattened value function $\boldsymbol{v}_f \in \mathbb{R}^{TN}$, (ii) a one-hot representation of the current location, and (iii) a one-hot representation of the current time-within-trial. These three quantities are sufficient to compute an optimal policy.

Recurrent neural networks

In this section, we provide details of how recurrent neural networks were trained and analysed. All networks were fully connected with $N_{\text{rec}} = 800$ hidden units (except Figure S8) and ReLU nonlinearities. The network dynamics were given by

$$\tau \dot{\boldsymbol{z}} = -\boldsymbol{z} + \boldsymbol{W}_{\text{in}} \boldsymbol{x} + \boldsymbol{W}_{\text{rec}} \boldsymbol{r} + \boldsymbol{b}_{\text{rec}} + \eta, \tag{13}$$

$$\boldsymbol{r} = [\boldsymbol{z}]_+ \tag{14}$$

$$\boldsymbol{y} = \boldsymbol{W}_{\text{out}} \boldsymbol{r} + \boldsymbol{b}_{\text{out}}. \tag{15}$$

Here, $\boldsymbol{z}$ is the network potential, $\boldsymbol{r}$ is the 'firing rate', $\boldsymbol{x}$ is the input, $\theta = \{\boldsymbol{W}_{\text{in}}, \boldsymbol{W}_{\text{rec}}, \boldsymbol{W}_{\text{out}}, \boldsymbol{b}_{\text{rec}}, \boldsymbol{z}_0, \boldsymbol{b}_{out}\}$ are the network parameters, and $\eta \sim \mathcal{N}(0, \sigma = 10^{-3})$ is Gaussian noise. All simulations used a time constant of $\tau = 5$ iterations. The output policy was defined in global allocentric coordinates as the desired next state, $\pi \propto e^{\boldsymbol{y}}$. In Figure S7, we also analyse a network that produced a policy in 'local' coordinates that indicated the desired movement direction. The input consisted of (i) a one-hot representation of the current location in the environment; (ii) a binary representation of the location of all walls (Jensen et al., 2024); and (iii) the reward $R_{\delta i}$ as described for the tasks above. Gaussian noise with a standard deviation of $\sigma = 10^{-3}$ was added to the inputs before passing them to the RNN.

For all analyses in the main text, the RNN performed 10 network iterations per environment iteration (action). Reward input was only provided during the planning phase and set to 0 during the execution phase. This was done to avoid biasing the execution-time representation towards a 'relative' subspace representation by providing reward input in that format (Supplementary Note). For comparison with this RNN, we also trained an RNN with reward input during both the planning and execution phase, which learned qualitatively similar representations and dynamics (Figure S6). For this network, the reward input was given relative to the current time as in the handcrafted STA, and the number of network iterations was randomly sampled between 9 and 11 before each environment iteration.

We optimised all network parameters θ to minimise the loss function

$$\mathcal{L}(\theta) = \lambda_\theta \mathcal{L}_{\text{params}}(\theta) + \mathbb{E}_{\tau\sim\pi_{\theta,opt}}\left[\sum_t \mathcal{L}^t_{\text{acc}} + \lambda_r \mathcal{L}^t_{\text{rate}} + \lambda_e \mathcal{L}^t_{\text{ent}}\right] \tag{16}$$

$$\mathcal{L}_{\text{params}}(\theta) = |\theta|_2^2 \tag{17}$$

$$\mathcal{L}^t_{\text{acc}} = \sum_{a\in\mathcal{A}^t_{\text{opt}}} -\pi_t(a) \tag{18}$$

$$\mathcal{L}^t_{\text{rate}} = |\boldsymbol{r}_t|_2^2 \tag{19}$$

$$\mathcal{L}^t_{\text{ent}} = \sum_a \pi_t(a)\log\pi_t(a). \tag{20}$$

$\mathcal{A}^t_{\text{opt}}$ is the set of optimal actions at time t, and $\mathbb{E}_{\tau\sim\pi_{\theta,opt}}[\cdot]$ indicates an expectation over trajectories induced by the policy of the agent, renormalised over optimal actions. That is, we consider an imitation learning setting where ties between equally optimal actions are broken according to the actual policy of the agent. We used $\lambda_\theta = 2\times10^{-7}$, $\lambda_r = 10^{-5}$, and $\lambda_e = 10^{-4}$ for all analyses. The RNNs were trained using Adam (Kingma and Ba, 2015) with a learning rate of 3×10^{-4} for 200,000 batches of 200 trials (250,000 batches for the RNNs trained in changing mazes).

All results are reported as mean and standard deviation across 5 separate RNNs that were trained from different random seeds and with different environment transition structures.

Analyses

In this section, we describe the analyses used to compare different handcrafted models and trained RNNs.

Decoding analyses

To decode future locations from neural activity (Figure 4C-D), we used scikit-learn (Pedregosa et al., 2011) to train L2-regularised logistic regression models that predicted location at time t_L from neural activity at time t_N with an inverse regularisation strength of $C = 1.0$. We performed this analysis in crossvalidation across locations at time t_N. In other words, we trained a decoder on data where the agent was in any state $s_{t_N} \neq s$ to predict all locations at time t_L, and we then tested this decoder in trials where the agent was in state s at time t_N. We repeated this analysis across all test locations s and averaged performance across the resulting 16 folds. We did this to test whether the RNN had a generalisable representation of future location, rather than an encoding of e.g. current location and neighboring values.

In Figure 4C (left), we plot the performance of decoders trained on $t_N = -1$ to predict location at any $t_L \geq 1$. In Figure 4D, we trained decoders on every pair of $t_N \in [0, 5]$ and $t_L \in [0, 5]$. We then averaged the performance across every 'delay' $t_L - t_N$.

To investigate the generalisation of decoders in Figure 4E, we trained a single decoder using neural activity at time $t_N = 1$ to predict location at time $t_L = 3$. We then applied the same decoder to neural activity at all times t'_N and quantified how well it predicted location at all times t'_L. This analysis was again performed in crossvalidation. A separate decoder was trained while holding out each location $s_1 = s$, and then tested only on trials where $t'_N = s$. Performance was averaged across all held-out locations and across 5 independently trained RNNs. Figure 4E shows the time at which the average predictive performance was highest for each t'_N. Figure S2 shows the full generalisation behaviour of the decoder.

To predict the time at which the agent would be at a particular location in Figure 4C (right), we analysed

every state s separately, and then averaged over all s. For each s, we identified trials where the RNN passed through s exactly once. We trained a decoder to predict the time at which s was visited and computed the test performance as a function of the true time at which s was visited. As before, all decoders were trained in crossvalidation across current agent location.

Comparisons of RNN performance and efficiency

In Figure 4F-G, we compare three classes of RNNs, which were trained on either the reward landscape task, the moving goal task, or the static goal task with a goal that changed between trials. The performance of each RNN was quantified in each of the three tasks as the probability of choosing the optimal first action (see above). We also computed the average parameter magnitudes of all the networks, $|\theta|_2^2$. Finally, we computed the average firing rate of each network in the static goal task as $\mathbb{E}_{\tau\sim\pi}\left[\sum_t |\boldsymbol{r}_t|_2^2\right]$. All other RNN analyses apart from Figure S3 used networks trained on the reward landscape task.

Subspace identification

Prior work has investigated the extent to which working memory subspaces are orthogonal (Xie et al., 2022; Dorrell et al., 2024). However, we are primarily interested in the *dynamics* between subspaces, which are easier to interpret in an orthonormal coordinate system. We therefore asked whether there exists a set of orthogonal subspaces that predict the location of the agent at every time in the future. To do so, we first simulated 6,000,000 trials and collected pairs of neural activity at time t and location at $t'(\delta)$ separately for each δ. We did this in two different ways. To estimate 'planning' subspaces, subspace δ was defined as a decoder that predicts location at time $t'(\delta) = \delta$ from neural activity at times $t \in \{-2, -1\}$. To estimate 'execution' subspaces, subspace δ was defined as a decoder that predicts location at $t'(\delta) = t + \delta$ from neural activity at any $t \geq 0$.

We then defined a predictive distribution parametrised by ϕ_δ for each δ:

$$p_{\phi_\delta}(s_{t'} = s_i) \propto \exp\left(\boldsymbol{c}_{\delta,i}\boldsymbol{r}_t + \boldsymbol{b}_\delta\right). \tag{21}$$

Finally, we minimised an objective function that combines the accuracy across δs and the overlap between subspaces:

$$\mathcal{L}(\theta) = \sum_\delta \left[\mathbb{E}_{\boldsymbol{r}_t, s_{t'(\delta)}\sim\mathcal{D}}\left[-\log p_{\phi_\delta}(s_{t'(\delta)})\right] + \alpha_1|\phi_\delta|_1 + \alpha_2|\phi_\delta|_2^2\right] + \alpha_{\text{orth}}\mathcal{L}_{\text{orth}}, \tag{22}$$

$$\mathcal{L}_{\text{orth}} := \sum_{\delta_k,\delta_l}\sum_{i,j} \hat{\boldsymbol{c}}_{\delta_k,i}^T \hat{\boldsymbol{c}}_{\delta_l,j}. \tag{23}$$

$\hat{\boldsymbol{c}}_{\delta_k,i}$ indicates the normalised parameter vector that predicts being at location s_i at a delay of δ_k. The parameters were optimised using ADAM with a learning rate of 5×10^{-3} until convergence or for a maximum of 2000 iterations. We used $\alpha_1 = 10^{-4}$, $\alpha_2 = 10^{-3}$, and α_{orth} was annealed from 0 to 2×10^{-3} over 500 iterations. These hyperparameters were chosen because they resulted in a good approximation to the 'true' subspaces in the handcrafted STA.

Estimating effective connectivity

To compute the effective connectivity between representations of different points in spacetime, we projected the learned network parameters into a coordinate system defined by the parameter vectors $\boldsymbol{C} \in \mathbb{R}^{NT\times N_{\text{rec}}}$. Each row of $\boldsymbol{C}$ is a normalised vector $\hat{\boldsymbol{c}}_{\delta,i}$ that predicts a particular point in spacetime. In this coordinate system, the input weights are given by $\boldsymbol{W}_{\text{in}}^{\text{eff}} = \boldsymbol{C}\boldsymbol{W}_{\text{in}}$; the output weights by $\boldsymbol{W}_{\text{out}}^{\text{eff}} = \boldsymbol{W}_{\text{out}}\boldsymbol{C}^T$; and the recurrent weights by $\boldsymbol{W}_{\text{rec}}^{\text{eff}} = \boldsymbol{C}\boldsymbol{W}_{\text{rec}}\boldsymbol{C}^T$. To analyse weights between 'adjacent' subspaces, we averaged the blocks of $\boldsymbol{W}_{\text{rec}}^{\text{eff}}$ that corresponded to weights from any subspace δ to $\delta + 1$ and from any subspace δ to $\delta - 1$. To avoid our analyses being biased by the fact that the networks were trained with supervised learning to predict optimal locations that could only be adjacent, we did not include the weights between subspaces 0 and 1 in this average.

To quantify the similarity between the recurrent weights in this spacetime coordinate system and different order adjacency matrices for the environment, we computed point-biserial correlations. The Δ^{th} order adjacency matrix $\boldsymbol{A}_\Delta \in \mathbb{R}^{N\times N}$ was defined as a binary matrix with elements equal to 1 for pairs of states

that can be reached from one another in Δ actions, and 0 for all other pairs of states. The 0^{th} order adjacency matrix was defined as the identity matrix.

Perturbation analyses

For the perturbation analyses in Figure 5F-H, we constructed an environment where the reward function $R(t, s)$ was (i) 1.0 for $(t, s) \in \{(0, 0), (1, 1), (2, 2), (3, 6), (4, 10), (5, 10), (6, 10)\}$, (ii) 0.7 for $(t, s) \in \{(1, 4), (2, 8), (3, 9)\}$, and -1 for all other points in spacetime. The RNN reliably converged to a representation of the optimal path. We first ran the network dynamics for 10 environment iterations without perturbation after the end of the normal planning period. We then continued to run the network dynamics for 10 environment iterations with a bias term defined by $\boldsymbol{b}_{\text{rec}}^{\text{stim}} = \boldsymbol{b}_{\text{rec}} + \alpha \hat{\boldsymbol{c}}_{2,8}$, where α is the stimulation strength. The perturbed representation was defined as the representation at the end of this perturbation period. The two example representations in Figure 5G used $\alpha = 0.3$ ('weak') and $\alpha = 10$ ('strong'). The quantification in Figure 5G used a range of α from 0 to 10 for the RNN, and from 0 to 500 for the handcrafted STA. This is because the attractor wells are deeper in the handcrafted STA, and a stronger perturbation is therefore required to push the network out of an attractor state. The control analyses (grey lines in Figure 5G and Figure S5C) were performed by running the same analysis on the RNNs, but with stimulation of the same magnitude in a random direction in neural state space. For the analysis in Figure 5H, we also removed the perturbation and ran the network dynamics for 10 environment iterations after the end of the perturbation period. All analyses in the main text focused on representations in the space of implied future trajectories, $p_{\phi_\delta}(s_{t'} = s_i)$. The 'representational change' was quantified as the L1 norm of the difference from the spacetime representation at the end of the normal planning period. See Figure S5 for an analysis of the raw firing rates.

RNNs trained in changing environments

For the analyses in Figure 6, we trained another set of RNNs in a version of the reward landscape task where the transition structure changed between trials. The locations of all walls in the environment were provided as a binary input to the agent (see Jensen et al., 2024 for details). For the performance comparison in Figure 6B (top), we evaluated the performance of these networks in the environments that the 'single structure' networks had been trained on.

The effective connectivity in Figure 6C-D was computed as in the RNNs trained on a single structure. For these analyses, we identified the subspaces from 1,000,000 trials in a single environment, and repeated this analysis across 30 different environments. We computed similarities between (i) the effective connectivity estimated in an environment and the adjacency matrix of the same environment, and (ii) the effective connectivity and the adjacency matrix from a different control environment. In Figure 6E, the 'subspace similarity' was defined as the correlation between the set of vectors that define the subspaces, averaged over all points in spacetime. We computed the similarity between (i) subspaces identified from two sets of independent trials from the same maze, and (ii) the same number of trials from two different mazes. Recall that the effective recurrent weights between subspaces are given by $\boldsymbol{W}_{\text{rec}}^{\text{eff}} = \boldsymbol{C}\boldsymbol{W}_{\text{rec}}\boldsymbol{C}^T$. By using different subspaces in different environments, the RNN changes $\boldsymbol{C}$ between environments, which changes the effective connectivity between pairs of future subspaces (Figure 6E).

For the analyses in Figure 6G-H, we repeated the subspace identification procedure, but with two notable differences. First, we used trials across many environments to find generalised directions in neural state space that predict the future in *any* environment. Second, we defined the objective function in terms of future *transitions* τ_δ^{ij} instead of locations. Figure 6G quantifies the effective connectivity between future transitions in consecutive subspaces (τ_δ^{ij} and $\tau_{\delta+1}^{kl}$) that are either 'consistent' ($k = j$), 'adjacent' ($j \neq k$ but s_i and s_k are adjacent in an environment with no walls), or any other transitions. In Figure 6H, we first projected the input corresponding to a given wall location w_{ij} onto each subspace and normalised the projection within each subspace. We then calculated the dot product between this projection and the normalised directions in neural state space that predict either (i) transitions between s_i and s_j, (ii) transitions from s_i or s_j to some other state s_k, or (iii) transitions that do not originate at s_i or s_j. Projection magnitudes were averaged over transitions within each of these three groups, then across subspaces, and finally the mean and standard deviation were computed across 5 RNNs.

Supplementary figures

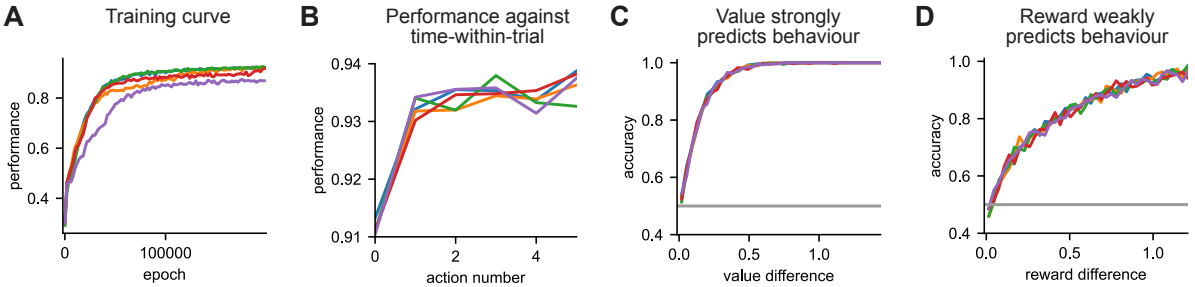

Figure S1: **RNN performance during and after training.** Each line in this figure corresponds to one of the five RNNs that were used for analyses in the main text. **(A)** Performance over the course of training, averaged over all actions within each trial. **(B)** Performance at the end of training as a function of the action number within the trial. When assessing performance at time t, trials were only included that had optimal choices up to time $t-1$. **(C)** Probability of choosing the action with highest value as a function of the value difference between the two actions with highest value. This analysis shows that errors are only made then the optimal action is close in value to the second best action. **(D)** Probability of choosing the action with highest reward as a function of the reward difference between the two actions with highest reward. Reward is less predictive of behaviour than value, confirming that the RNNs compute long-term value rather than relying on greedy reward.

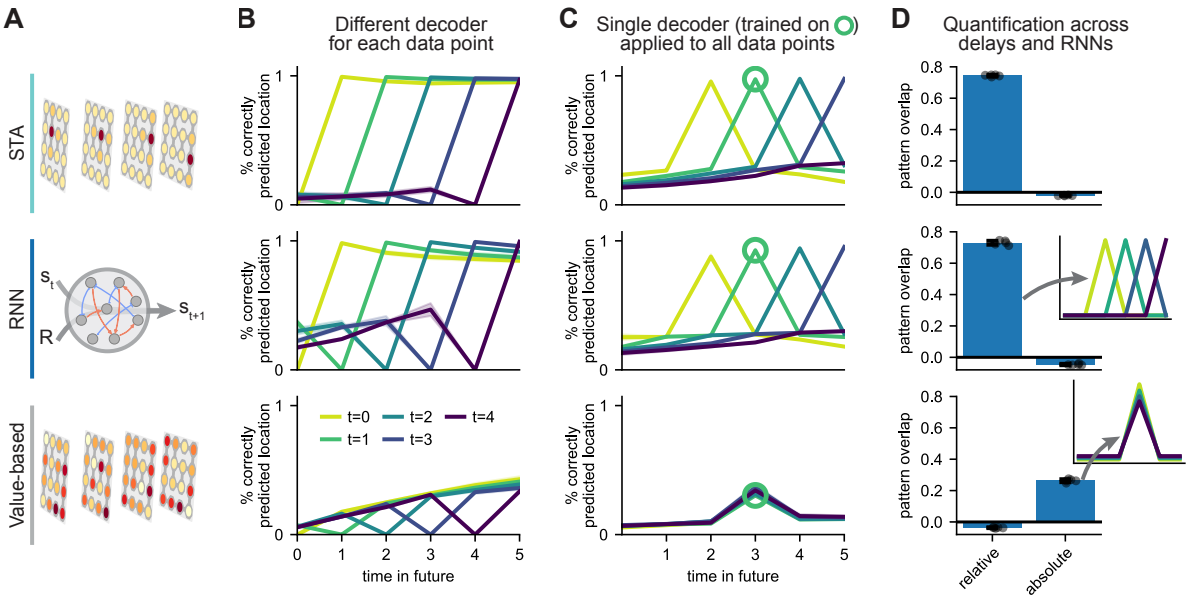

Figure S2: **Additional analyses of learned RNN representations.** **(A)** We compare a spacetime attractor; an RNN trained on the reward landscape task; and an agent that analytically computes a full spacetime value function. The value-based agent computes an optimal policy from 'neural activity' containing (i) the value function, (ii) the current location, and (iii) the time-within-trial (Methods). **(B)** Decoding accuracy of agent location at different times (x-axis) from neural activity at every other time (lines; legend). All decoders were trained in crossvalidation across the current agent location (Methods). This is why the accuracy is zero when decoding location from activity at the same time. **(C)** We trained a single decoder to predict location at time 3 from neural activity at time 1 (green circle). The same decoder predicts location at time $t+2$ (x-axis) from neural activity at any other t (lines). **(D)** Similarity of decoding patterns to idealised representations of future location in 'relative' or 'absolute' time (schematics).

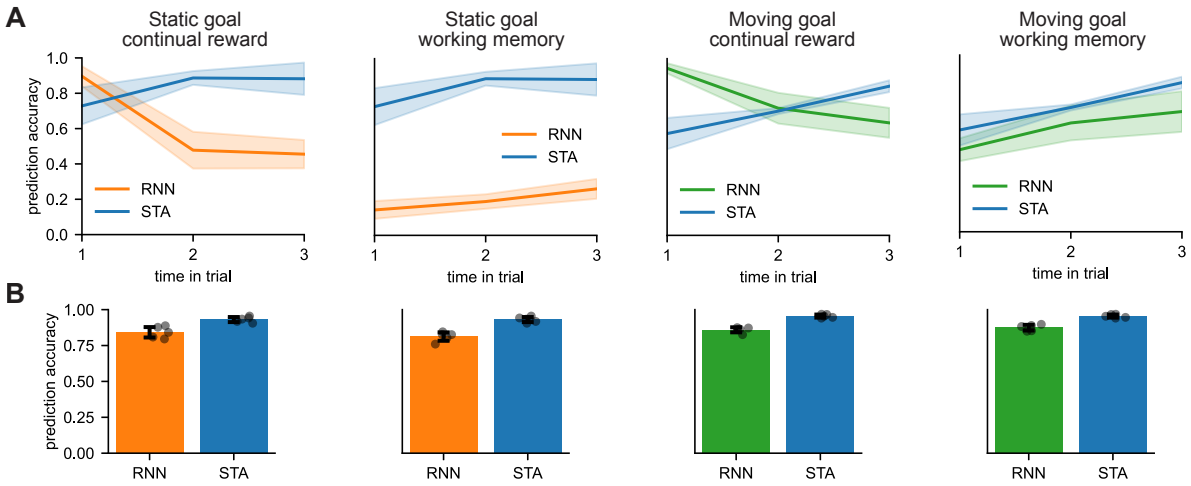

Figure S3: **RNNs trained on simpler tasks do not learn spacetime representations.** In this figure, we analyse the representations of four RNNs trained on all combinations of (i) the static goal or the moving goal task, and (ii) reward input throughout the task ('continual') or reward input only during the planning phase ('working memory'). For all analyses in this figure, we only included trials where an optimal agent would intercept the goal in 3 to 6 actions. **(A)** Decoding accuracy for agent location at different times (x-axis) from neural activity at the end of the planning period. Decoders were trained in crossvalidation across the current agent location. Only the RNN trained on the moving goal task in a working memory setting seems to learn a generalisable representation of the future. This network is also unlikely to have learned a full STA, since it fails catastrophically on the reward landscape task (Figure 4F). Note that the decoding accuracy generally *increases* slightly for the true STA as a function of time-within-trial. This is because the navigation tasks have stronger correlations between consecutive positions, which leads to some degree of overfitting on the training data. This overfitting is less prominent later in a trial, where the space of possible locations conditioned on the current location is larger. **(B)** In this analysis, we trained a decoder to predict whether the agent would be at a particular location at *any* time in the trial from neural activity at the end of planning. Binary decoders were trained for each possible future location in crossvalidation across the current agent location. Bars indicate the average predictive accuracy across all binary decoders and current locations. The simpler networks seem to learn a representation of whether they will be at a given location at some point in the future.

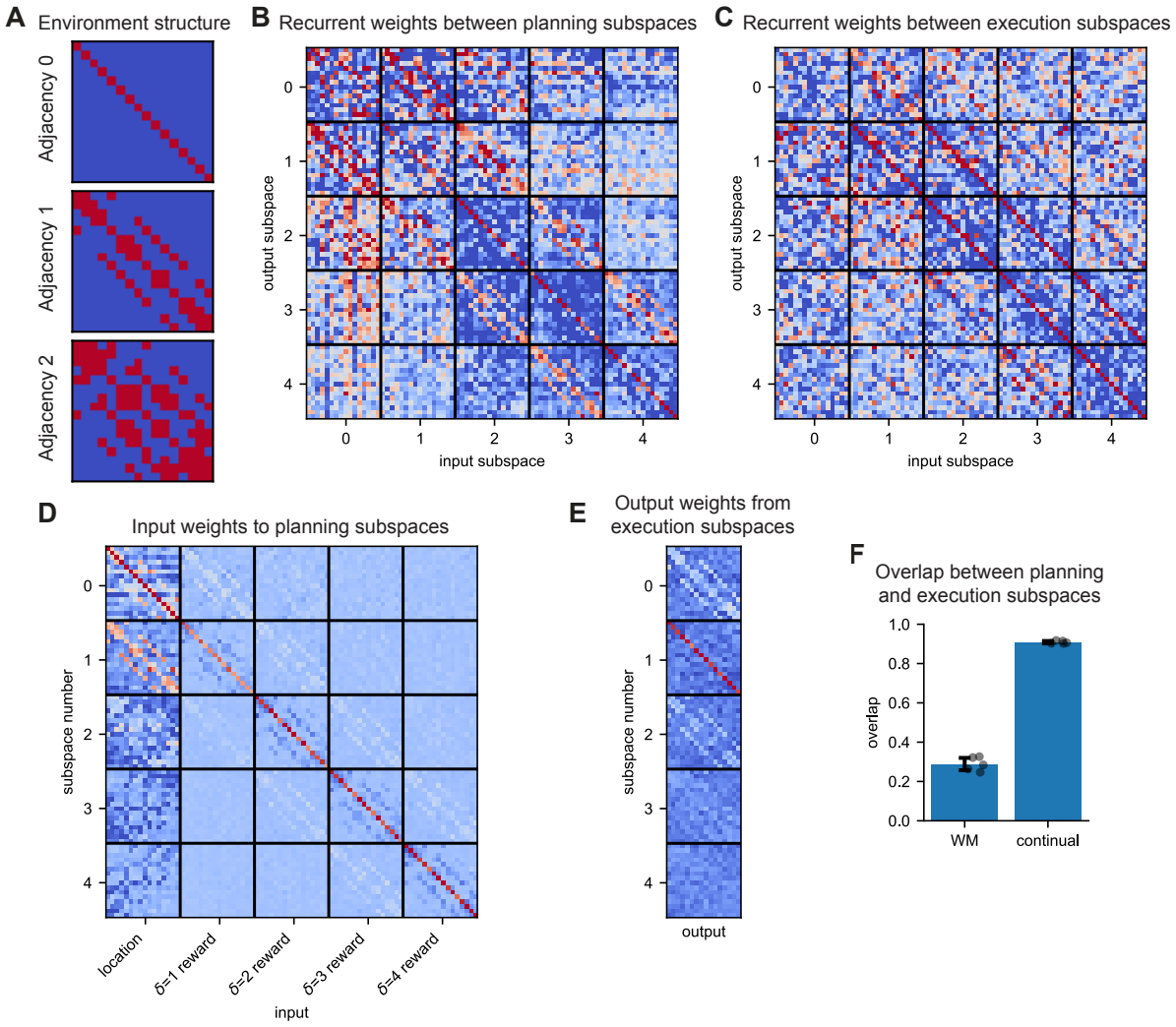

Figure S4: **Parameters learned by the reward landscape RNN.** Network weights are projected into an orthonormal coordinate system with axes that maximally predict future locations. All weight matrices are for a single example RNN, since the environment differs between networks, and the connectivity is therefore slightly different. **(A)** Structure of the environment that the RNN was trained in, illustrated as the 0^{th} order adjacency matrix (the identity matrix), the 1^{st} order adjacency matrix, and the 2^{nd} order adjacency matrix **(B)** Recurrent weight matrix estimated during the planning period, which shows structure resembling the environment adjacency matrix in the off-diagonal blocks. **(C)** Recurrent weight matrix estimated during the execution period, which shows an additional 'feedforward' component that copies information from later to earlier subspaces. We posit that this component of the connectivity matrix helps implement the conveyor belt dynamics identified in Figure 4E (Supplementary Note). **(D)** Input weight matrix estimated during the planning period. The 'current' subspace receives location input, and future subspaces receive reward corresponding to the appropriate time in the future. **(E)** Output weight matrix estimated during the execution period. The policy is read out from the 'immediate future' subspace as expected in a spacetime attractor. **(F)** Overlap between subspaces estimated during the planning and execution periods. This analysis was performed both for the standard RNN ('WM'), and for a network trained with continual reward input throughout the trial instead of only during the planning phase ('continual'; Figure S6). The WM RNN uses separate subspaces for computation of the plan and subsequent execution, consistent with the different connectivity patterns in (B) and (C). The continual RNN can use the same subspaces for planning and execution since it always receives the same type of input.

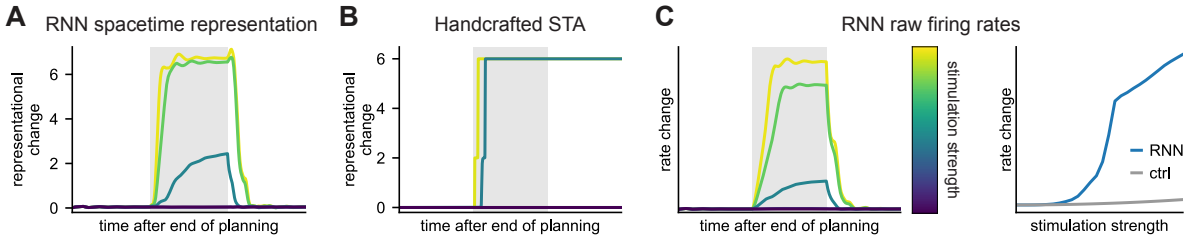

Figure S5: **Additional analyses of attractor dynamics. (A)** Change in implied spacetime representation over time in the trained RNN for different perturbation strengths (reproduced from Figure 5H). **(B)** Change in spacetime representation over time in the handcrafted spacetime attractor for different perturbation strengths. In contrast to the trained RNN, the 'low value' path is a fixed point of the perturbation-free network dynamics in the handcrafted network. At the end of a sufficiently strong perturbation, the representation can therefore stay in this new fixed point. **(C)** Change in RNN firing rates for different perturbation strengths. While the change in implied spacetime representation completely saturates with perturbation strength, the change in firing rates continues to increase with perturbation strength. This is expected because the network has a non-saturating ReLU nonlinearity. Small external perturbations are still quenched when quantifying the change in representation using the raw firing rates instead of the implied spacetime representation.

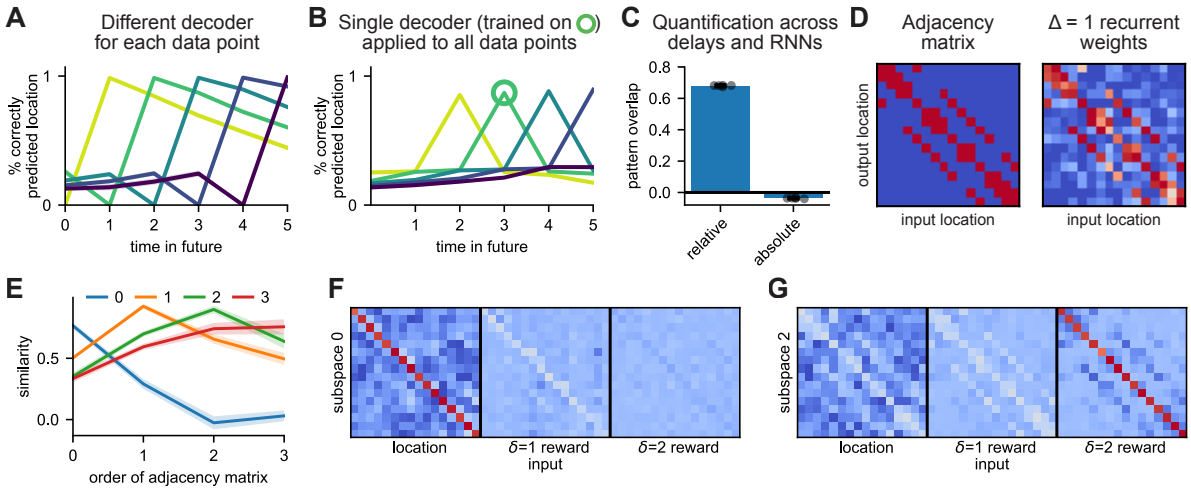

Figure S6: **RNNs trained with continual reward input also learn spacetime attractors.** In the main text, we focused on an RNN that was trained with reward input provided during an initial 'planning phase', while no information was given about the reward function during subsequent 'execution'. In this figure, we perform some of the same analyses on an RNN that receives reward input throughout the entire task. In this setting, the task could in theory be solved using a 'feedforward' strategy that does not rely on recurrent dynamics at all. However, the RNNs still learn a spacetime attractor-like solution. **(A)** Decoding accuracy of agent location at different times (x-axis) from neural activity at every other time. Each line corresponds to predictions from neural activity at a different time in the trial from $t = 0$ (yellow) to $t = 4$ (blue). Decoders were trained in crossvalidation across the current agent location. **(B)** We trained a single decoder to predict location at time 3 from neural activity at time 1 (green circle). The same decoder predicts location at time $t + 2$ (x-axis) from neural activity at any other t (lines). **(C)** Similarity of decoding patterns to idealised representations of future location in 'relative' or 'absolute' time (Figure S2). **(D)** The average recurrent weights between subspaces separated by a single action resemble the adjacency matrix of the environment. **(E)** Correlation between (i) the average connectivity between subspaces separated by Δ actions (lines; legend), and (ii) different order adjacency matrices (x-axis). **(F)** Input weights to the 'current' subspace ($\delta = 0$). **(G)** Input weights to a 'future' subspace ($\delta = 2$).

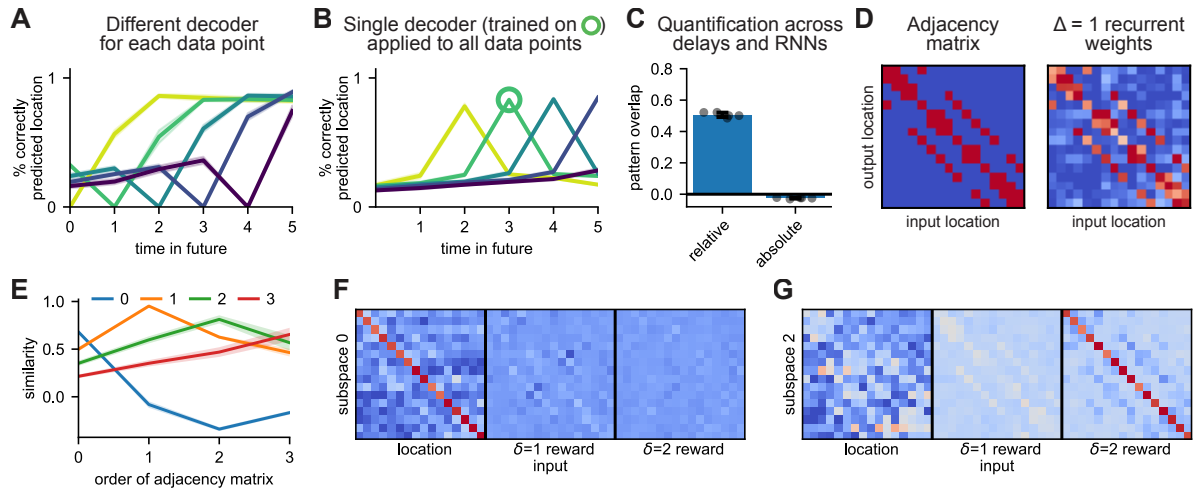

Figure S7: **RNNs with a local action space also learn spacetime attractors.** In the main text, we analysed an RNN that generated a global allocentric policy, consisting of a probability distribution over all locations that was renormalised over 'adjacent' locations before sampling an action. In this figure, we perform some of the same analyses on a network that outputs a 'local' policy in an action space consisting of 'north', 'south', 'east', and 'west'. This RNN also learns a spacetime attractor. **(A)** Decoding accuracy of agent location at different times (x-axis) from neural activity at every other time. Each line corresponds to predictions from neural activity at a different time in the trial from $t = 0$ (yellow) to $t = 4$ (blue). Decoders were trained in crossvalidation across the current agent location. **(B)** We trained a single decoder to predict location at time 3 from neural activity at time 1 (green circle). The same decoder predicts location at time $t + 2$ (x-axis) from neural activity at any other t (lines). **(C)** Similarity of decoding patterns to idealised representations of future location in 'relative' or 'absolute' time (Figure S2). **(D)** The average recurrent weights between subspaces separated by a single action resemble the adjacency matrix of the environment. **(E)** Correlation between (i) the average connectivity between subspaces separated by Δ actions (lines; legend), and (ii) different order adjacency matrices (x-axis). **(F)** Input weights to the 'current' subspace ($\delta = 0$). **(G)** Input weights to a 'future' subspace ($\delta = 2$).

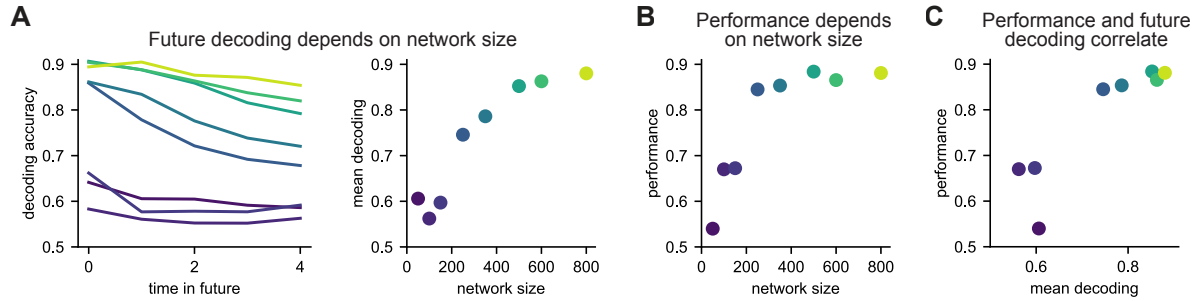

Figure S8: **RNN representations and performance across network sizes.** We trained a series of RNNs with different network sizes ranging from 50 (dark blue) to 800 (yellow) hidden units. **(A)** Networks with approximately 300 or more units learned a spacetime representation, and the future could be decoded from the hidden state of the network at the end of the planning period. **(B)** Task performance saturated as a function of network size at approximately 300 hidden units. **(C)** Task performance increased with the ability of the network to represent the entire future explicitly. These results mirror the findings of Whittington et al. (2023) that RNNs trained on working memory tasks learn a similar 'slot-like' solution only if the network is large enough.

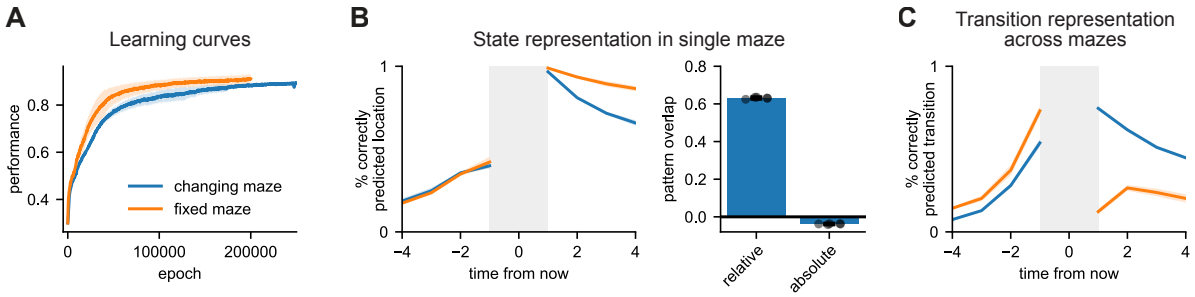

Figure S9: **Additional analyses of RNNs trained with changing environment structure. (A)** Learning curves of RNNs trained on the reward landscape task in a single maze ('fixed maze'; orange) or with a different structure on every trial ('changing maze'; blue). **(B)** We took the RNNs trained across many mazes and evaluated them in a single maze (blue). Future states could be decoded almost as well as in the RNNs trained in a single maze (orange). Additionally, the representations in each maze were more similar to idealised representations of future location in 'relative' than 'absolute' time (right). **(C)** When considering data across many mazes, future *transitions* could be decoded in a way that generalised across current transition and maze structure (blue). Such a representation did not exist in RNNs trained in a single maze (orange).

Supplementary discussion of architecture and modelling choices

In this section, we discuss some of the many architectural and modelling choices that went into our work. As is the case for much work in modern computational neuroscience, the space of models was vast – and larger than we could feasibly explore fully in a single paper. In what follows, we aim to further motivate the choices that were made in the main paper and to provide additional intuition for the importance and effect of various architectural choices and hyperparameters in our work. This note is also not exhaustive, but we hope that it will be useful both for the reader looking to gain a deeper understanding of our work, and for those who want to draw inspiration from it in their own research.

Reward input format

When training RNNs with a reward function that changes in time, there are multiple different ways this could be provided as an input to the agent. The three most natural choices are:

1. A relative encoding, where a given input channel always reflects the reward in δ actions.
2. An absolute encoding, where a given input channel always reflects reward at time t.
3. A 'working memory' setting, where reward input is only provided during a planning period prior to the first action. At this time, the relative and absolute representations are identical, and no choice has to be made between the two.

One might expect a relative reward input to bias RNNs towards relative future representations, and 'conveyor belt' dynamics, and an absolute reward input to bias RNNs towards an absolute code. While RNNs trained with relative reward input did show strong relative coding of the future (Figure S6), RNNs trained with an absolute input learned somewhat mixed representations that were harder to interpret.

To avoid having to choose between these two, most of the analyses in the paper were done on an RNN trained in the 'working memory' setting. We also saw slightly stronger 'spacetime' representations in the working memory setting. In fact, Figure S6 shows that the effective planning horizon of the 'relative' model is only 4-5 steps, and predictive performance of future states drops off beyond that. This is presumably because a shorter planning horizon is sufficient for near-optimal performance in the task, and the regularisation encourages networks to not encode more information than necessary. The RNN therefore effectively re-plans at every step using a spacetime attractor with a depth of approximately 4.

The handcrafted spacetime attractor always received a 'relative' reward input. Otherwise, it would need to include either (i) an additional 'memory' component, (ii) a transformation from absolute to relative inputs, or (iii) a mechanism for changing the readout between subspaces. We expect that if biological agents use spacetime attractor-like dynamics, there may be settings where the code is relative and settings where it is absolute. This might depend on whether the plan is shaped around constraints in relative time (e.g. having to meet someone in 10 minutes) or absolute time (e.g. having to meet someone at 1 pm).

Planning period

When training the RNNs, we included a planning period prior to the first action. During this period, the output of the network had no effect on the environment. We did this (i) to separate the 'planning' period from the 'execution' period in the working memory setting, and (ii) because it slightly improved performance. In the handcrafted STA, there is no distinction between planning and execution, since the inputs are the same in both cases. However, convergence is slower before the initial action because the network state starts further from a fixed point. This means that the STA can be run for fewer iterations after the first action, which effectively corresponds to a longer 'planning phase' followed by faster execution.

Network iterations per action

For all RNNs, we used a variable number of planning iterations. This is because we were interested in stable representations of future behaviour, rather than networks learning to time their dynamics to initiation. In the RNNs with a relative reward input, we also varied the number of network iterations between each action for the same reason. However, this is not possible in the working memory RNN, since it would not know when it

had taken an action. There are two possible solutions to this problem. (i) we can provide a specific 'action' input that tells the network when the environment has updated. (ii) we can use a fixed number of network iterations between every action and allow the network to learn the speed of the environment. We opted for the second option to keep the inputs as simple as possible. We suspect that this could lead to changes in the subspaces used at different 'phases' of the dynamics, but we did not test this explicitly. Instead, we focused on the planning period when analysing the connectivity of the working memory RNNs to circumvent this potential complication. When analysing the activity during 'execution' in Figure 4D, we used only neural activity right before each action.

Execution-time dynamics

The fixed points of the STA dynamics are input-dependent, and the inputs to the handcrafted model change when it takes an action, because the location and/or future reward will be different. For this reason, the representation can automatically update to reflect the 'path-to-go' after each action. However, this involves recomputing the future to some extent, which is 'wasteful' since it has already been computed once. Additionally, the representation can get stuck in local minima because the barrier to switching between representations is non-zero. We therefore included an additional component in the STA weight matrix, which was 1 for all off-diagonal elements that implement a transfer of information to earlier subspaces. This transfer component was only active for 100 network iterations (two time constants) after each action to update the representation before settling into a new fixed point. This is similar to how the fruit fly head direction circuit uses populations of 'shift' (PEN-1) neurons to update the internal heading, and these PEN-1 neurons are gated by angular velocity input (Turner-Evans et al., 2017, 2020). Including an explicit shift component in the STA is not necessary for any of the main results in the paper, but it stabilises the dynamics to some extent. We posit that the RNNs learn something similar, as suggested by the feedforward component of the 'execution period' parameters in Figure S4C.

Noise sensitivity

The handcrafted STA is more or less susceptible to different types of noise. The model is very robust to noise added to the neural potential ('z'; Methods), which can be interpreted as a spacetime distribution in log probability space. This is because non-desirable locations in spacetime can have very negative values, which are not very noise sensitive. The STA is more sensitive to noise added to the firing rates ('r'; Methods), which can be interpreted as a spacetime distribution in actual probability space. This is because addition of positive noise to some location s_i in subspace δ will propagate through the adjacency matrix to all neighboring locations at time $\delta + 1$. This can lead to representations that 'teleport' between distant locations if s_i is near the reward location and therefore receives strong input from the reward function or future subspaces.

The STA is sensitive to structural noise for a similar reason. Adding a small positive value to weights corresponding to elements of the adjacency matrix that are meant to be zero leads to dynamics that include a finite probability of teleporting between these distant locations. In this work, we mitigate the structural noise sensitivity by using a 'pessimistic' estimate of the adjacency matrix as the base weights before adding noise (Methods). In other words, we subtract a small constant from all weights to ensure that distant locations are connected with weights that are zero or negative, even though they are noisy. This may be less of a problem in biological networks, since Dale's law ensures that synapses are either excitatory, inhibitory, or absent.

The sensitivity to both rate noise and structural noise is higher when the input strength is larger (β; Methods), which biases the representation more strongly towards rewarded locations. Conversely, the converged representation is more diffuse if the input strength is weaker, because there is a smaller bias towards rewarded locations. This is particularly true when planning towards distant rewards. The strength of the reward input therefore has to balance the planning depth with the susceptibility to teleporting representations. If there is no noise in the system, the input strength can safely be very large, which leads to robust performance for long planning horizons. If there is more noise in the system, the input strength should be smaller, which increases noise robustness at the expense of a shorter effective planning horizon. When training RNNs across tasks, they will naturally learn to balance the robustness of the representation with the required planning depth. Indeed, the analyses in Figure S5 suggest that RNNs learn to do so better than our handcrafted models.

Choice of RNN learning algorithm

We used supervised learning to train all RNNs in this paper. In other words, the RNNs were trained to predict the behaviour of an optimal agent. An alternative would have been to train the networks end-to-end using reinforcement learning. We opted for the supervised setting for two reasons. Firstly, supervised learning is more stable, which leads to more robust results that are less sensitive to hyperparameters and use less compute. These features make it much simpler for others to build on our work. Secondly, we do not think cortical representations are learned from scratch via reinforcement learning. Instead, we are of the opinion that cortical representations are likely learned via predictive or 'semi-supervised' learning. The basal ganglia can then use reinforcement learning to map these representations onto actions (Blanco-Pozo et al., 2024; Zintgraf et al., 2019). We expect that training the RNNs with reinforcement learning would yield similar results, but we have not tested this explicitly.

Convergence of RNN training

Convergence was in general fairly good, but it did vary with hyperparameters to some extent. For some combinations of regularisation strengths, the networks sometimes got trapped in local minima that corresponded to incomplete structural learning. These networks would partially learn the structure of the environment, but they would fail to learn connections between some pairs of states that were actually connected, which impaired performance. Similarly, some RNNs trained on the changing maze task erroneously 'hard coded' some transitions instead of having them flexibly modulated by the inputs. In general, the accuracy of the learned structure was correlated with model performance for both the networks trained in a single maze and networks trained in changing mazes. Additionally, the overall loss was higher for the networks that failed to learn the full task structure, suggesting that it is an issue of convergence rather than regularisation making the partially learned solution optimal.

Long distance parameters in the RNN

In Figure 5 and Figure S4, we saw that the trained RNNs learn some long-range connections between subspaces separated by more than one action. This differs somewhat from the handcrafted STA, which only has connections between adjacent subspaces. However, the presence of long-range connections in the RNN is not too surprising. In particular, we expect this additional structure to stabilise fixed points corresponding to possible trajectories, since it inhibits any trajectory that includes impossible n-step transitions. We suspect that stabilising the dynamics through weaker connectivity between all subspaces is cheaper than strong connectivity exclusively between adjacent subspaces. This may be a consequence of the L2 regularisation used to train the networks, which favors many small parameters over few large parameters. In future work, it could be interesting to explore whether the degree of long range connectivity is lower when using L1 regularisation instead. The result of such an analysis may also depend on how disentangled the spacetime representations are, which we did not explore in the present paper.

Discrete space and time

We have discretised space and time throughout this work. This makes the models and analyses simpler, because the action space is discrete and one action always leads to a step change in the environment. However, we expect that the basic ideas extend to continuous space and time as well. In this case, neurons would still represent particular points in spacetime. Pairs of neurons would be connected as a function of their difference in preferred location in a way that reflects which locations can be reached in a unit time. This is similar to the mechanism used for angular velocity integration in ring attractors and path integration in grid attractors. If the speed of the agent can vary, it might be necessary to represent different speeds in different connections or neurons, and the desired speed at every point in time could be inferred together with the trajectory.

Stochastic environments

We have worked with deterministic environments throughout this paper. This choice greatly simplifies the spacetime attractor, since the deterministic adjacency matrix can be built into the network connectivity. When working in a stochastic environment, we would intuitively want the connections to represent $\max_a p(s_{t+1}|s_t, a)$, which is a generalisation of the adjacency matrix for deterministic environments. We have not explored this explicitly but consider it an important extension for future work. One potential challenge in the stochastic case is that the spacetime attractor as formulated here effectively performs planning as inference under the

assumption that the posterior distribution over locations factorises across time. This assumption may have more severe consequences in stochastic environments, where correlations are more important. In particular, this assumption generally leads to a collapse to a 'modal' representation of a single trajectory rather than representations of entire distributions of trajectories. In stochastic environments, that means it is possible to converge to a single representation of a trajectory that is unlikely to happen in reality, even with the correct choice of actions.

Relationship to sequence working memory

The spacetime attractor is strongly inspired by representations identified for sequence working memory (El-Gaby et al., 2023; Xie et al., 2022; Chen et al., 2024; Tian et al., 2024; Whittington et al., 2023). These sequence memory tasks have notable similarities to the reward landscape task studied in this work. In particular, sequence working memory often involves presenting an animal with a sequence of 2-4 options (e.g. 'a', 'b', and 'c') that are sampled from a set of N possible states or actions (Xie et al., 2022; El-Gaby et al., 2023). The animal is then rewarded for repeating the sequence at response time. The task therefore has a reward function that is categorical (only one element is rewarded at a given moment in time) and changes in time (first 'a' is rewarded, then 'b'...). This is very similar to the changing reward function in the reward landscape task. A notable difference is that the sequence memory tasks have no constraints on the possible transitions – any element can be produced before or after any other element. For this reason, each sequence element can be treated independently, and there is no need to pass reward or value information between subspaces. The independent sequence working memory task can therefore be seen as a special case of the reward landscape task, where (i) only one state is rewarded at each point in time, and (ii) any state can be reached from any other state, so the adjacency matrix is uniform. Interestingly, theoretical work shows that explicit representations of the future preferentially emerge for sequence working memory when correlations (or more precisely, 'range dependence') between subsequent elements are weak, and the space of possible sequences is therefore large (Dorrell et al., 2024). This is reminiscent of our finding that spacetime representations for planning preferentially emerge in RNNs trained on the reward landscape task, where the space of possible optimal trajectories is larger than in the static and moving goal tasks.

Comparisons with TD and SR agents

In Figure 3, we compare the representations and performance of the STA to temporal difference learners and successor representation agents. We show that the STA solves 'dynamic' problems that these algorithms struggle with. This is in some sense a property of the state space rather than the decision making algorithm. It would be possible to construct TD and SR agents in a 'space-by-time' state space, which would allow them to solve these dynamic problems. Our message is not that this is not possible. Instead we are highlighting that the way these algorithms are usually implemented involves representations that are 'flat' across time, and we use them as a comparison to show why spacetime representations can be useful. If a TD learner was implemented with a space-by-time state space, it would be able to solve tasks where the reward changes within a trial but the same pattern is seen across all trials. The SR agent with a space-by-time state space could solve the general reward landscape task. In the simplest implementation, this would require inversion of a matrix $\boldsymbol{T} \in \mathbb{R}^{NT \times NT}$, which is computationally expensive. However, it is possible that the structure of $\boldsymbol{T}$ could be exploited to invert it more efficiently, which would be an interesting avenue for future research.

Summary

As is evident from this discussion, many choices went into this work that could have been different. We do not claim to have explored the full space of models and tasks, and we are not trying to argue that the spacetime attractor is a silver bullet for planning and decision making. Instead, we have tried to argue that STA-like models are interesting solutions to a range of problems that are relevant to prefrontal cortex and not widely studied in systems neuroscience. However, many open questions remain, some of which we have briefly motivated here. We therefore hope that this paper will inspire others to further explore these questions both experimentally and computationally.