Vector-based Navigation using Grid-like Representations in Artificial Agents.

Andrea Banino[1,2,3*], Caswell Barry[2*], Benigno Uria[1], Charles Blundell[1], Timothy Lillicrap[1], Piotr Mirowski[1], Alexander Pritzel[1], Martin J. Chadwick[1], Thomas Degris[1], Joseph Modayil[1], Greg Wayne[1], Hubert Soyer[1], Fabio Viola[1], Brian Zhang[1], Ross Goroshin[1], Neil Rabinowitz[1], Razvan Pascanu[1], Charlie Beattie[1], Stig Petersen[1], Amir Sadik[1], Stephen Gaffney[1], Helen King[1], Koray Kavukcuoglu[1], Demis Hassabis[1,4], Raia Hadsell[1], Dharshan Kumaran[1,3]

[1]DeepMind, 5 New Street Square, London EC4A 3TW, UK.

[2]Department of Cell and Developmental Biology, University College London, London, UK

[3]Centre for Computation, Mathematics and Physics in the Life Sciences and Experimental Biology (CoMPLEX), University College London, London, UK

[4]Gatsby Computational Neuroscience Unit, 25 Howland Street, London W1T 4JG, UK

[*]equal contribution.

Deep neural networks have achieved impressive successes in diverse areas ranging from object recognition to complex games such as Go[1,2]. Navigation, however, remains a substantial challenge for artificial agents, with deep neural networks trained by reinforcement learning (RL)[3–5] failing to rival the proficiency of mammalian spatial behavior, underpinned by grid cells in the entorhinal cortex[6]. Grid cells are viewed to provide a multi-scale periodic representation that functions as a metric for coding space[7,8] which is critical for integrating self-motion (path integration)[6,7,9] and planning direct trajectories to goals (vector-based navigation)[7,10,11]. We set out to leverage the computational functions of grid cells to develop a deep RL agent with mammalian-like navigational abilities. We first trained a recurrent network to perform path integration, leading to the emergence of representations resembling grid cells, as well as other entorhinal cell types[12]. We then showed that this representation

provided an effective basis for an agent to locate goals in complex, unfamiliar, and changeable environments — optimizing the primary objective of navigation through deep RL. The performance of agents endowed with grid-like representations surpassed that of an expert human and comparison agents, with the metric quantities necessary for vector-based navigation derived from grid-like units within the network. Further, grid-like representations enabled agents to conduct shortcut behaviours reminiscent of those performed by mammals. Our findings show that emergent grid-like representations furnish agents with a Euclidean spatial metric and associated vector operations, providing a foundation for proficient navigation. As such, our results support neuro-scientific theories that see grid cells as critical for vector-based navigation[7,10,11], demonstrating that the latter can be combined with path-based strategies to support navigation in complex environments.

The ability to self-localize in the environment and update one's position on the basis of self-motion are core components of navigation[13]. We trained a deep neural network to path integrate within a square arena (2.2m×2.2m), using simulated trajectories modelled on those of foraging rodents (see Methods). The network was required to update its estimate of location and head direction based on translational and angular velocity signals, mirroring those available to the mammalian brain[12,14,15] (see Methods, Fig. 1a&b). Velocity was provided as input to a recurrent network with a Long Short-Term Memory architecture (LSTM) which was trained using backpropagation through time (see Methods and Supplemental Discussion), allowing the network to dynamically combine current input signals with activity patterns reflecting past events (see Methods, Fig. 1a). The LSTM projected to place and head direction units via a linear layer — units with activity defined as a simple linear function of their input (see Extended Data Figure 1 for architecture). Importantly, the linear layer was subject to regularization, in particular dropout[16], such that 50% of the units were randomly silenced at each time step. The vector of activities in the place and head direction units, corresponding to the current position, was provided as a supervised training signal at each time step (see Methods and Extended Data Figure 1). This form of supervision follows evidence that in mammals, place and head direction representations exist in close anatomical proximity to entorhinal grid cells[12] and emerge in rodent pups prior to the appearance of mature grid cells[17,18]. Equally, in adult rodents, entorhinal grid cells are known to project to the hippocampus[19] and appear to contribute to the neural activity of place cells[19].

As expected, the network was able to path integrate accurately in this setting involving foraging behavior (mean error after 15s trajectory, 16cm vs 91cm for an untrained network, effect size = 2.83; 95% CI [2.80, 2.86], Fig. 1b&c). Strikingly, individual units within the linear layer of the network developed stable spatial activity profiles similar to neurons within the entorhinal network[6,12] (Fig. 1d, Extended Data Figure 2). Specifically, 129 of the 512 linear layer units (25.2%) resembled grid cells, exhibiting significant hexagonal periodicity (gridness[20]) versus a null distribution generated by a conservative fields shuffling procedure (see Methods). The scale of the grid-patterns, measured from the spatial autocorrelograms of the activity maps[20], varied between units (range 28cm to 115cm, mean 66cm) and followed a multi-modal distribution, consistent with empirical results from rodent grid cells[21,22] (Fig. 1e). To assess these clusters we fit mixtures of Gaussians, finding the most parsimonious number by minimizing the Bayesian Information Criterion (BIC). The distribution was best fit by 3 Gaussians (means 47cm, 70cm, and

106cm), indicating the presence of scale clusters with a ratio between neighbouring clusters of approximately 1.5, closely matching theoretical predictions[23] and lying within the range reported for rodents[21,22] (Fig. 1e, Extended Data Figure 3). The linear layer also exhibited units resembling head direction cells (10.2%), border cells (8.7%), and a small number of place cells[12] as well as conjunctions of these representations (Fig. 1d,f&g, Extended Data Figure 2).

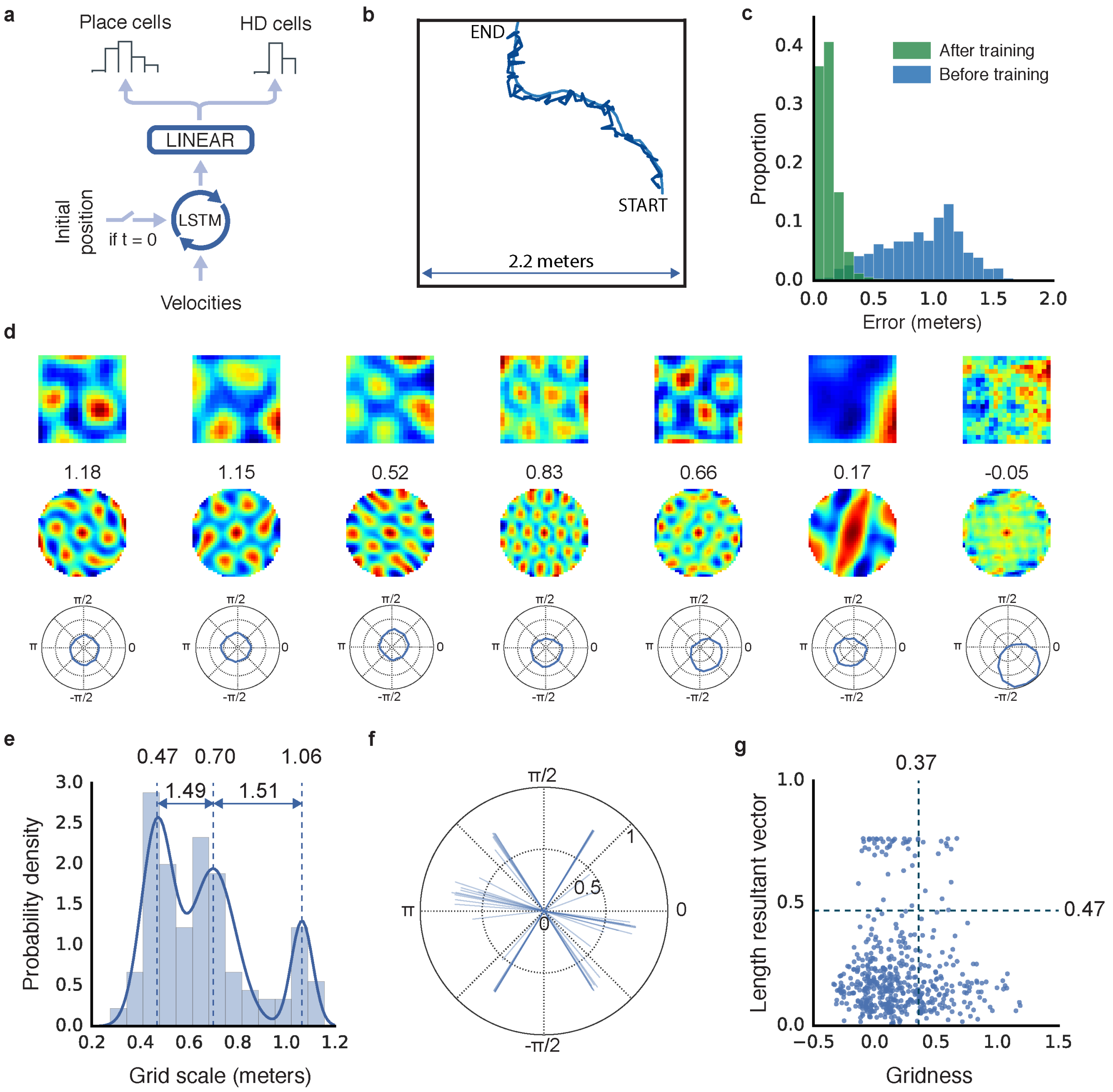

a
Place cells
HD cells
LINEAR
Initial position
if t = 0
LSTM
Velocities
b
END
START
2.2 meters
c
0.4
0.3
0.2
0.1
0.0
Proportion
After training
Before training
0.0
0.5
1.0
1.5
2.0
Error (meters)
d
1.18
1.15
0.52
0.83
0.66
0.17
-0.05
π/2
π
0
-π/2
e
0.47
0.70
1.06
1.49
1.51
3.0
2.5
2.0
1.5
1.0
0.5
0.0
Probability density
0.2
0.4
0.6
0.8
1.0
1.2
Grid scale (meters)
f
π/2
π
0
-π/2
1
0.5
0
g
0.37
1.0
0.5
0.0
0.47
Length resultant vector
−0.5
0.0
0.5
1.0
1.5
Gridness

Figure 1: **Entorhinal-like representations emerge in a network trained to path integrate.** a, Schematic of network architecture (see Extended Data Figure 1 for details) . b, Example trajectory (15s), self-location decoded from place cells resembles actual path (respectively, dark and light-blue). c, Accuracy of decoded location before (blue) and after (green) training. d, Linear layer units exhibit spatially tuned responses resembling grid, border, and head direction cells. Ratemap shows activity over location (top), spatial autocorrelogram of the ratemap with gridness indicated (middle), polar plot show activity vs. head direction (bottom). e, Spatial scale of grid-like units ($n = 129$) is clustered. Distribution is more discrete[22] than chance (effect size = 2.98, 95% CI [0.97, 4.91]) and best fit by a mixture of 3 Gaussians (centres 0.47, 0.70 & 1.06m, ratio=1.49 & 1.51). f, Directional tuning of the most strongly directional units ($n = 52$). Lines indicate length and orientation of resultant vector (see Methods), exhibiting a six-fold clustering reminiscent of conjunctive grid cells[24]. g, Distribution of gridness and directional tuning. Dashed lines indicate 95% confidence interval from null distributions (based on 500 data permutations), 14 (11%) grids exhibit directional modulation (see Methods). Similar results were seen in a circular environment (Extended Data Figure 3).

To ascertain how robust these representations were, we retrained the network 100 times, in each instance finding similar proportions of grid-like units (mean 23% SD 2.8%, units with significant gridness scores) and other spatially modulated units (Extended Data Figure 3). Conversely, grid-like representations did not emerge in networks without regularization (e.g. dropout, see Methods; also see [25], Extended Data Figure 4). Therefore, the use of regularization, including dropout which has been viewed to be a parallel of noise in neural systems[16], was critical to the emergence of entorhinal-like representations. Notably, therefore, our results show that grid-like representations reminiscent of those found in the mammalian entorhinal cortex emerge in a generic network trained to path integrate, contrasting with previous approaches using pre-configured grid cells (e.g. [26]; see Supplemental Discussion). Further our results are consistent with the view that grid cells represent an efficient and robust basis for a location code updated by self-motion cues[6–9].

Next, we sought to test the hypothesis that the emergent representations provide an effective basis function for goal-directed navigation in complex, unfamiliar, and changeable environments, when trained through deep RL. Entorhinal grid cells have been proposed to provide a Euclidean spatial metric and thus support the calculation of goal-directed vectors, enabling animals to follow direct routes to a remembered goal, a process known as vector-based navigation[7,10,11]. Theoretically, the advantage of decomposing spatial location into a multi-scale periodic code, as provided by grid cells, is that the relative position of two points can be retrieved by examining the difference in the code at the level of each scale — combining the modulus remainders to return the true vector[7,11] (Fig. 2a). However, despite the obvious utility of such a framework, experimental evidence for the direct involvement of grid representations in goal-directed navigation is still lacking.

To develop an agent with the potential for vector-based navigation, we incorporated the “grid network” described above, into a larger architecture that was trained using deep RL (Fig. 2d, Extended Data Figure 5). As before, the grid network was trained using supervised learning but, to better approximate the information available to navigating mammals, it now received velocity signals perturbed with random noise as well as visual input. Experimental evidence suggests that place cell input to grid cells corrects for drift and anchors grids to environmental cues[21]. To parallel this, visual input was processed by a ”vision module” consisting of a convolutional network that produced place and head direction cell activity patterns which were provided as input to the grid network 5% of the time – akin to a moving animal making occasional, imperfect observations of salient environmental cues[27] (see Methods, Fig. 2b&c and Extended Data Figure 5). The output of the linear layer of the grid network, corresponding to the agent’s current location, was provided as input to the “policy LSTM”, a second recurrent network controlling both the agent’s actions and outputting a value function. Additionally, whenever the agent reached the goal, the ”goal grid code” — activity in the linear layer — was subsequently provided to the policy LSTM during navigation as an additional input.

We first examined the navigational capacities of the agent in a simple setting inspired by the classic Morris water maze (Fig. 2b&c; 2.5m×2.5m square arena; see Methods and Supplemental Results). Notably, the agent was still able to self localize accurately in this more challenging setting where ground truth information about location was not provided and velocity inputs were noisy (mean error after 15s trajectory, 12cm vs 88cm for an untrained network, effect size = 2.82; 95% CI [2.79, 2.84], Fig. 2e). Further, the agent exhibited proficient goal-finding abilities, typically taking direct routes to the goal from arbitrary starting locations (Fig. 2h). Performance exceeded that of a control place cell agent (Fig. 2f, Supplemental Results and Methods), chosen because place cells provide a robust representation of self-location but are not thought to provide a substrate for long range vector calculations[11]. We examined the units in the linear layer, again finding a heterogeneous population resembling those found in entorhinal cortex, including grid-like units (21.4%) as well as other spatial representations (Fig. 2g, Extended Data Figure 6) — paralleling the dependence of mammalian grid cells on self-motion information[15,28] and spatial cues[6,21].

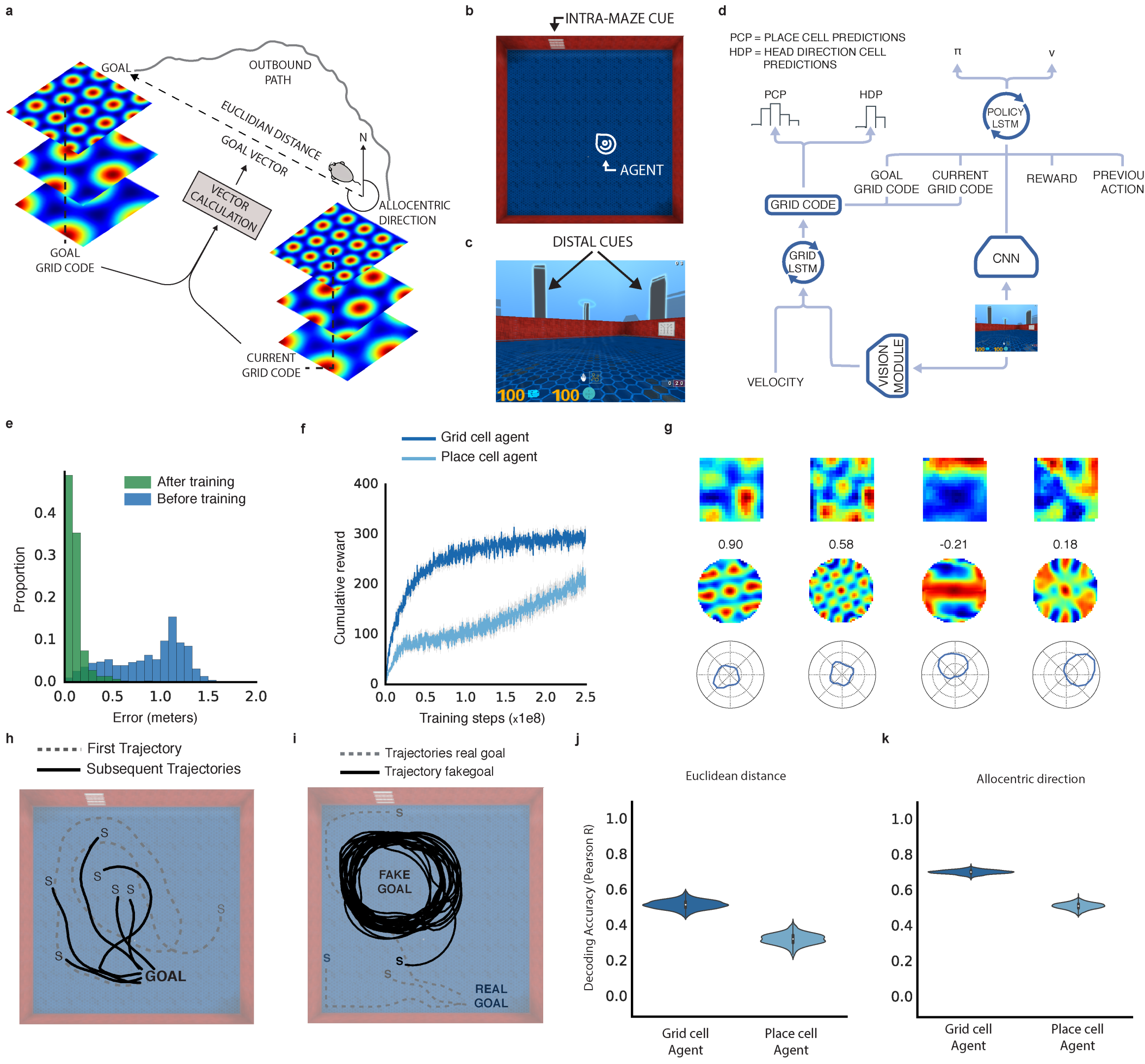

a
GOAL
OUTBOUND PATH
EUCLIDIAN DISTANCE
GOAL VECTOR
N
VECTOR CALCULATION
ALLOCENTRIC DIRECTION
GOAL GRID CODE
CURRENT GRID CODE
b
INTRA-MAZE CUE
AGENT
c
DISTAL CUES
100
100
d
PCP = PLACE CELL PREDICTIONS
HDP = HEAD DIRECTION CELL PREDICTIONS
π
v
PCP
HDP
POLICY LSTM
GOAL GRID CODE
CURRENT GRID CODE
REWARD
PREVIOUS ACTION
GRID CODE
GRID LSTM
CNN
VELOCITY
VISION MODULE
e
After training
Before training
Proportion
0.0
0.1
0.2
0.3
0.4
0.0
0.5
1.0
1.5
2.0
Error (meters)
f
Grid cell agent
Place cell agent
Cumulative reward
0
100
200
300
400
0.0
0.5
1.0
1.5
2.0
2.5
Training steps (x1e8)
g
0.90
0.58
-0.21
0.18
h
First Trajectory
Subsequent Trajectories
S
GOAL
i
Trajectories real goal
Trajectory fakegoal
FAKE GOAL
REAL GOAL
S
j
Euclidean distance
Decoding Accuracy (Pearson R)
0.0
0.2
0.4
0.6
0.8
1.0
Grid cell Agent
Place cell Agent
k
Allocentric direction
0.0
0.2
0.4
0.6
0.8
1.0
Grid cell Agent
Place cell Agent

Figure 2: **One-shot open field navigation to a hidden goal.** a, Schematic of vector-based navigation[11]. b, Overhead view of typical environment (icon indicates agent and facing direction). c, Agent view of (b). d, Schematic of Deep RL architecture (see Extended Data Figure 5) e, Accuracy of self-location decoded from place cell units. f, Performance of grid cell agent and place cell agent (y-axis shows reward obtained within a single episode, 10 points per goal arrival, gray band displays the 68% confidence interval based on 5000 bootstrapped samples). g, As before the linear layer develops spatial representations similar to entorhinal cortex. Left to right, 2 grid cells, 1 border cell, and 1 head direction cell. h, On the first trial of an episode the agent explores to find the goal and subsequently navigates directly to it. i, After successful navigation, the policy LSTM was supplied with a ”fake” goal grid-code, directing the agent to this location where no goal was present. j&k, Decoding of goal-directed metric codes (i.e. Euclidean distance and direction) from the policy LSTM of grid cell and place cell agents. The bootstrapped distribution (1000 samples) of correlation coefficients are each displayed with a violin plot overlaid on a Tukey boxplot.

We next turn to our central claim, that grid cells endow agents with the ability to perform vector-based navigation, enabling downstream regions to calculate goal directed vectors by comparing current activity with that of a remembered goal[7,10,11]. In the agent, we expect these calculations to be performed by the policy LSTM, which receives the current activity pattern over the linear layer (termed “current grid code”; see Fig. 2d and Extended Data Figure 5) as well as that present the last time the agent reached the goal (termed “goal grid code”), using them to control movement. Hence we performed several manipulations, which yielded four lines of evidence in support of the vector-based navigation hypothesis (see Supplemental Results for details).

First, to demonstrate that the goal grid code provided sufficient information to enable the agent to navigate to an arbitrary location, we substituted it with a ”fake” goal grid code sampled randomly from a location in the environment (see Methods). The agent followed a direct path to the newly specified location, circling the absent goal (Fig. 2i) — similar to rodents in probe trials of the Morris water maze (escape platform removed). Secondly, we demonstrated that withholding the goal grid code from the policy LSTM of the grid cell agent had a strikingly deleterious effect on performance (see Extended Data Fig. 6c). Thirdly, we demonstrated that the policy LSTM of the grid cell agent contained representations of key components of vector-based navigation (Figure 2j&k), and that both Euclidean distance (difference in r = 0.17; 95% CI [0.11, 0.24]) and allocentric goal direction (difference in r = 0.22; 95% CI [0.18, 0.26]) were represented more strongly than in the place cell agent. Notably, a neural representation of goal distance has recently been reported in mammalian hippocampus[29]. Finally, we provide evidence consistent with a prediction of the vector-based navigation hypothesis, namely that a targeted lesion (i.e. silencing) to the most grid-like units within the goal grid code should have a greater adverse effect on performance and the representation of vector-based metrics (e.g. Euclidean distance) than a sham lesion (i.e silencing of non-grid units; average score for 100 episodes: 126.1 vs. 152.5, respectively; effect size = 0.38, 95% CI [0.34, 0.42] see Supplemental Results).

Having demonstrated the effectiveness of grid-like representations in optimizing one-shot goal learning in a simple square arena, we assessed the agent’s performance in complex, procedurally-generated multi-room environments, termed “goal-driven” and “goal-doors” (see Methods). Notably, these environments are challenging for deep RL agents with external memory (see Extended Data Figure 7e,f,h&i and Supplemental Results). Again, the grid-cell agent exhibited high levels of performance, was strikingly robust across a range of network hyperparameters (see Extended Data Figure 7a,b&c), and reached the goal more frequently than either control agents or a human expert — a typical benchmark for the performance of deep RL agents in game playing scenarios[2] (Fig. 3e&f and see Supplemental Results). Further, when agents were tested, without retraining, in environments considerably larger than those seen previously, only the grid cell agent was able to generalise effectively (Fig. 3g&h and see Supplemental Results). Despite the complexity of the ”goal-driven” environment, we could still decode the key metric codes from the grid agent policy LSTM with high accuracy during the initial period of navigation – with decoding accuracy substantially higher in the grid cell agent than both the place cell and deep RL control agents (Figure 3j&k and Supplemental Results, Extended Data Figures 8 and 9 for control agent architectures).

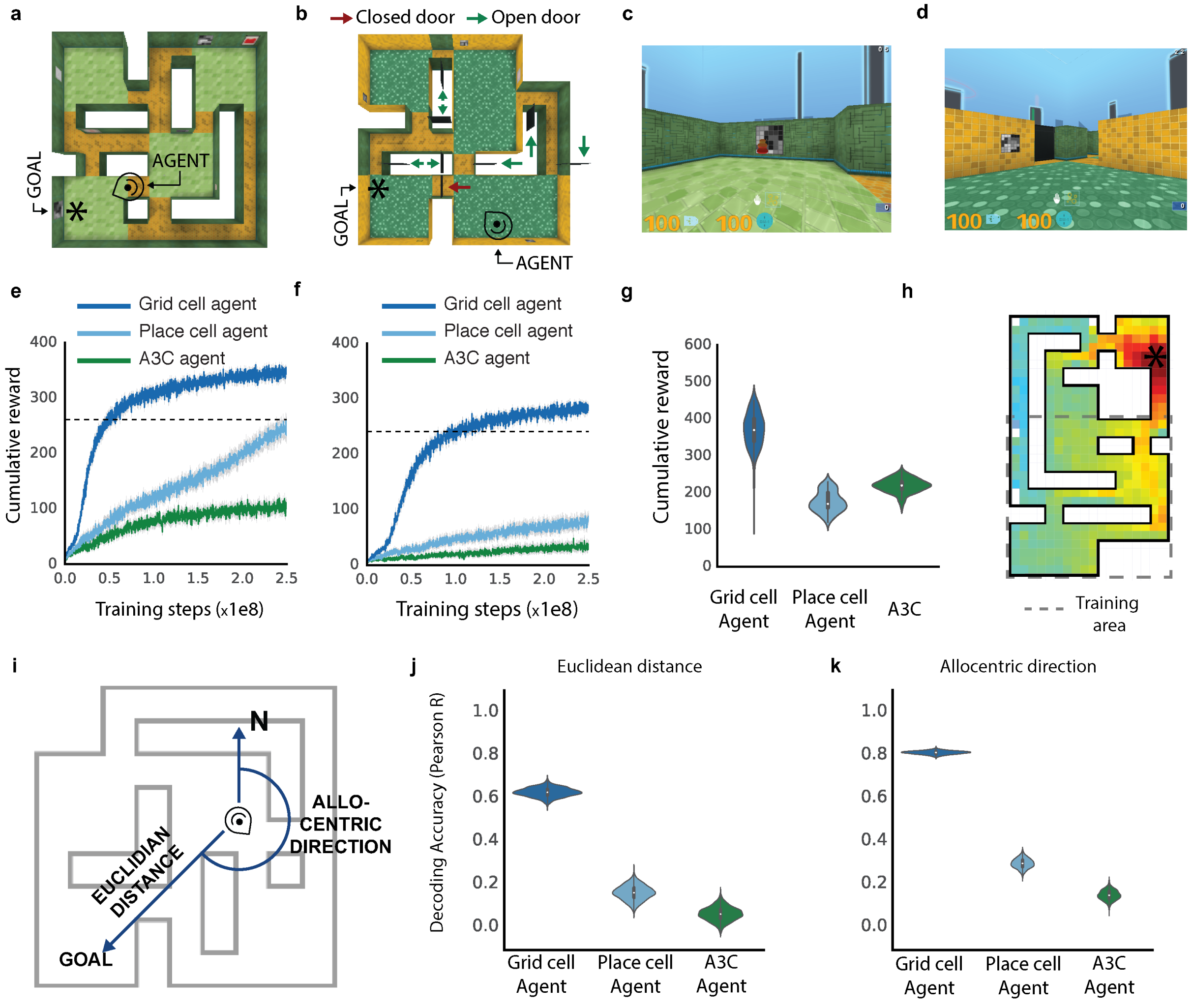

a
GOAL
AGENT
b
Closed door
Open door
GOAL
AGENT
c
100
100
d
100
100
e
Grid cell agent
Place cell agent
A3C agent
Cumulative reward
400
300
200
100
0
0.0
0.5
1.0
1.5
2.0
2.5
Training steps (x1e8)
f
Grid cell agent
Place cell agent
A3C agent
400
300
200
100
0
0.0
0.5
1.0
1.5
2.0
2.5
Training steps (x1e8)
g
Cumulative reward
600
500
400
300
200
100
0
Grid cell Agent
Place cell Agent
A3C
h
Training area
i
N
ALLO-CENTRIC DIRECTION
EUCLIDIAN DISTANCE
GOAL
j
Euclidean distance
Decoding Accuracy (Pearson R)
1.0
0.8
0.6
0.4
0.2
0.0
Grid cell Agent
Place cell Agent
A3C Agent
k
Allocentric direction
1.0
0.8
0.6
0.4
0.2
0.0
Grid cell Agent
Place cell Agent
A3C Agent

Figure 3: **Navigation in complex environments** a-b, Overhead view of multi-room environments “goal-driven” (a) and “goal-doors” (i.e. with stochastic doors) (b) Goal (*) and agent locations (head icon) are displayed. c-d, Agent views of (a) & (b) showing red goal and closed black door. e-f, Agent training performance curves for (a) & (b), and performance of human expert (dotted line). Performance is mean cumulative reward over 100 episodes. The gray band displays the 68% confidence interval based on 5000 bootstrapped samples g, Distribution of test performance over 100 episodes, showing ability of agents to generalize to a larger version of goal-driven environment, displayed with a violin plot overlaid on a Tukey boxplot for each agent. h, The value function of the grid cell agent is projected onto an example larger goal doors environment as a heatmap. Dotted lines show the extent of the original training environment. Despite the larger size, the value function clearly approximates Euclidean distance to goal. i, Schematic displaying the key metrics required for vector-based navigation to a goal. j-k, Decoding of vector-based metric codes from the policy LSTM of agents during navigation. The bootstrapped distribution (1000 samples) of correlation coefficients are displayed with a violin plot overlaid on a Tukey boxplot in each case.

Finally, a core feature of mammalian spatial behaviour is the ability to exploit novel shortcuts and traverse unvisited portions of space, a capacity thought to depend on vector-based navigation[9, 11]. Strikingly, the grid cell agent – but not comparison agents – robustly demonstrated these abilities in specifically designed neuroscience-inspired mazes, taking direct routes to the goal as soon as they became available (Fig. 4, Extended Data Figure 10 and Supplemental Results).

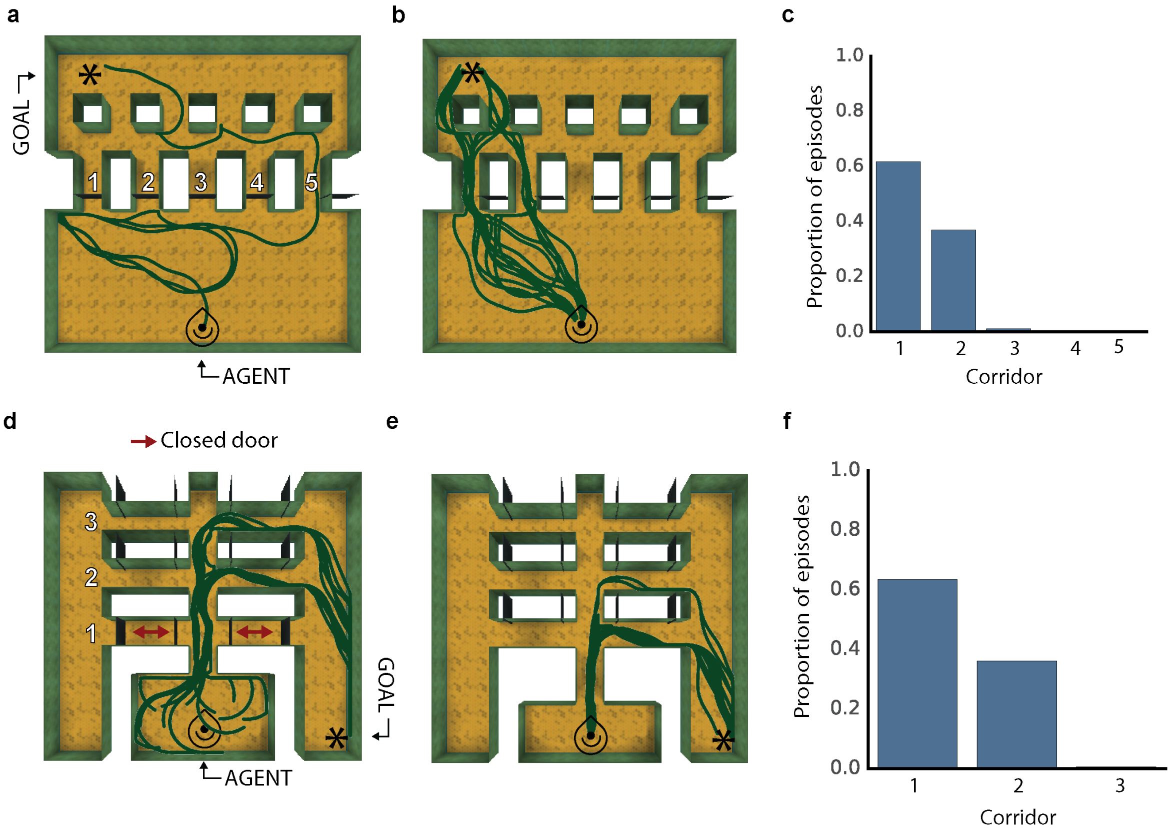

a
GOAL
1
2
3
4
5
AGENT
b
c
1.0
0.8
0.6
0.4
0.2
0.0
Proportion of episodes
1
2
3
4
5
Corridor
d
Closed door
3
2
1
GOAL
AGENT
e
f
1.0
0.8
0.6
0.4
0.2
0.0
Proportion of episodes
1
2
3
Corridor

Figure 4: **Flexible use of short-cuts** a, Example trajectory from grid cell agent during training in the linear sunburst maze (only door 5 open; icon indicates start location). b, Testing configuration with all doors open: grid cell agent uses the newly available shortcuts (100 episodes shown). c, Histogram showing agent's strong preference for most direct doors. d, Example grid cell agent trajectories (100) during training in the double E-maze (corridor 1 doors closed). e, Testing configuration with corridor 1 open, and 100 grid agent trajectories. f, Histogram analogous to panel c, agent prefers newely-available shortest route. See Extended Data Figure 10 for performance of place cell agent.

Conventional simultaneous localization and mapping (SLAM) techniques typically require an accurate and complete map to be built, with the nature and position of the goal externally defined[30]. In contrast, the deep reinforcement learning approach described in this work has the ability to learn complex control policies end-to-end from a sparse reward, taking direct routes involving shortcuts to goals in an automatic fashion - abilities that exceed previous deep RL approaches[3–5], and that would have to be hand-coded in any SLAM system.

Our work, in demonstrating that grid-like representations provide an effective basis for flexible navigation in complex novel environments, supports theoretical models of grid cells in vector-based navigation previously lacking strong empirical support[7,10,11]. We also show that vector-based navigation can be effectively combined with a path-based barrier avoidance strategy to enable the exploitation of optimal routes in complex multi-compartment environments. In sum, we argue that grid-like representations furnish agents with a Euclidean geometric framework — paralleling their proposed computational role in mammals as an early-developing Kantian-like spatial scaffold that serves to organize perceptual experience[17,18].

1. LeCun, Y., Bengio, Y. & Hinton, G. Deep learning. *Nature* **521**, 436–444 (2015).

2. Silver, D. *et al.* Mastering the game of go with deep neural networks and tree search. *Nature* **529**, 484–489 (2016).

3. Oh, J., Chockalingam, V., Singh, S. P. & Lee, H. Control of memory, active perception, and action in minecraft. In *Proc. of International Conference on Machine Learning, ICML* (2016).

4. Kulkarni, T. D., Saeedi, A., Gautam, S. & Gershman, S. J. Deep successor reinforcement learning. *CoRR* **abs/1606.02396** (2016). URL `http://arxiv.org/abs/1606.02396`.

5. Mirowski, P. *et al.* Learning to navigate in complex environments. *International Conference on Learning Representations* (2017).

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

7. Fiete, I. R., Burak, Y. & Brookings, T. What grid cells convey about rat location. *Journal of Neuroscience* **28**, 6858–6871 (2008).

8. Mathis, A., Herz, A. V. & Stemmler, M. Optimal population codes for space: grid cells outperform place cells. *Neural Computation* **24**, 2280–2317 (2012).

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

10. Erdem, U. M. & Hasselmo, M. A goal-directed spatial navigation model using forward trajectory planning based on grid cells. *European Journal of Neuroscience* **35**, 916–931 (2012).

11. Bush, D., Barry, C., Manson, D. & Burgess, N. Using grid cells for navigation. *Neuron* **87**, 507–520 (2015).

12. Barry, C. & Burgess, N. Neural mechanisms of self-location. *Current Biology* **24**, R330–R339 (2014).

13. Mittelstaedt, M.-L. & Mittelstaedt, H. Homing by path integration in a mammal. *Naturwissenschaften* **67**, 566–567 (1980).

14. Bassett, J. P. & Taube, J. S. Neural correlates for angular head velocity in the rat dorsal tegmental nucleus. *Journal of Neuroscience* **21**, 5740–5751 (2001).

15. Kropff, E., Carmichael, J. E., Moser, M.-B. & Moser, E. I. Speed cells in the medial entorhinal cortex. *Nature* **523**, 419–424 (2015).

16. Srivastava, N., Hinton, G. E., Krizhevsky, A., Sutskever, I. & Salakhutdinov, R. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research* **15**, 1929–1958 (2014).

17. Wills, T. J., Cacucci, F., Burgess, N. & O'keefe, J. Development of the hippocampal cognitive map in preweanling rats. *Science* **328**, 1573–1576 (2010).

18. Langston, R. F. *et al.* Development of the spatial representation system in the rat. *Science* **328**, 1576–1580 (2010).

19. Zhang, S.-J. *et al.* Optogenetic dissection of entorhinal-hippocampal functional connectivity. *Science* **340**, 1232627 (2013).

20. Sargolini, F. *et al.* Conjunctive representation of position, direction, and velocity in entorhinal cortex. *Science* **312**, 758–762 (2006).

21. Barry, C., Hayman, R., Burgess, N. & Jeffery, K. J. Experience-dependent rescaling of entorhinal grids. *Nature neuroscience* **10**, 682–684 (2007).

22. Stensola, H. *et al.* The entorhinal grid map is discretized. *Nature* **492**, 72–78 (2012).

23. Stemmler, M., Mathis, A. & Herz, A. V. Connecting multiple spatial scales to decode the population activity of grid cells. *Science advances* **1**, e1500816 (2015).

24. Doeller, C. F., Barry, C. & Burgess, N. Evidence for grid cells in a human memory network. *Nature* **463**, 657–661 (2010).

25. Kanitscheider, I. & Fiete, I. Training recurrent networks to generate hypotheses about how the brain solves hard navigation problems. *arXiv preprint arXiv:1609.09059* (2016).

26. Milford, M. J. & Wyeth, G. F. Mapping a suburb with a single camera using a biologically inspired slam system. *IEEE Transactions on Robotics* **24**, 1038–1053 (2008).

27. Hardcastle, K., Ganguli, S. & Giocomo, L. M. Environmental boundaries as an error correction mechanism for grid cells. *Neuron* **86**, 827–839 (2015).

28. Chen, G., King, J. A., Burgess, N. & O'Keefe, J. How vision and movement combine in the hippocampal place code. *Proceedings of the National Academy of Sciences* **110**, 378–383 (2013).

29. Sarel, A., Finkelstein, A., Las, L. & Ulanovsky, N. Vectorial representation of spatial goals in the hippocampus of bats. *Science* **355**, 176–180 (2017).

30. Dissanayake, M. G., Newman, P., Clark, S., Durrant-Whyte, H. F. & Csorba, M. A solution to the simultaneous localization and map building (slam) problem. *IEEE Transactions on Robotics and Automation* **17**, 229–241 (2001).

Methods

Path integration: Supervised learning experiments.

Simplified 2D environment. Simulated rat trajectories of duration T were generated in square and circular environments with walls of length L (diameter in the circular case). The simulated rat started at a uniformly sampled location and facing angle within the enclosure. A rat-like motion model[31] was used to obtain trajectories that uniformly covered the whole environment by avoiding walls (see Table 1 in supplementary methods for the model's parameters).

Ground truth place cell distribution. Place cell activations, $\vec{c} \in [0,1]^N$, for a given position $\vec{x} \in \mathbb{R}^2$, were simulated by the posterior probability of each component of a mixture of two-dimensional isotropic Gaussians,

$$c_i = \frac{\mathrm{e}^{-\frac{\left\|\vec{x}-\vec{\mu}_i^{(c)}\right\|_2^2}{2\left(\sigma^{(c)}\right)^2}}}{\sum_{j=1}^{N} \mathrm{e}^{-\frac{\left\|\vec{x}-\vec{\mu}_j^{(c)}\right\|_2^2}{2\left(\sigma^{(c)}\right)^2}}}, \tag{1}$$

where $\vec{\mu}_i^{(c)} \in \mathbb{R}^2$, the place cell centres, are N two-dimensional vectors chosen uniformly at random before training, and $\sigma^{(c)}$, the place cell scale, is a positive scalar fixed for each experiment.

Ground truth head-direction cell distribution. Head-direction cell activations, $\vec{h} \in [0,1]^M$, for a given facing angle φ were represented by the posterior probability of a each component of a mixture of Von Mises distributions with concentration parameter $\kappa^{(h)}$,

$$h_i = \frac{\mathrm{e}^{\kappa^{(h)} \cos\left(\varphi-\mu_i^{(h)}\right)}}{\sum_{j=1}^{M} \mathrm{e}^{\kappa^{(h)} \cos\left(\varphi-\mu_j^{(h)}\right)}}, \tag{2}$$

where the M head direction centres $\mu_i^{(h)} \in [-\pi, \pi]$, are chosen uniformly at random before training, and $\kappa^{(h)}$ the concentration parameter is a positive scalar fixed for each experiment.

Supervised learning inputs. In the supervised setup the grid cell network receives, at each step t, the egocentric linear velocity $v_t \in \mathbb{R}$, and the sine and cosine of its angular velocity $\dot{\varphi}_t$.

Grid cell network architecture The grid cell network architecture (Extended Data Figure 1) consists of three layers: a recurrent layer, a linear layer, and an output layer. The single recurrent layer is an LSTM (*long short-term memory* [32]) that projects to place and head direction units via the linear layer. The linear layer implements regularisation through dropout[16]. The recurrent LSTM layer consists of one cell of 128 hidden units, with no peephole connections. Input to the recurrent LSTM layer is the vector $[v_t, \sin(\dot{\varphi}_t), \cos(\dot{\varphi}_t)]$. The initial cell state and hidden state of the LSTM, $\vec{l}_0$ and $\vec{m}_0$ respectively, are initialised by computing a linear transformation of the ground truth place and head-direction cells at time 0:

$$\vec{l}_0 = W^{(cp)}\vec{c}_0 + W^{(cd)}\vec{h}_0 \tag{3}$$

$$\vec{m}_0 = W^{(hp)}\vec{c}_0 + W^{(hd)}\vec{h}_0 \tag{4}$$

The parameters of these two linear transformations ($W^{(cp)}$, $W^{(cd)}$, $W^{(hp)}$, and $W^{(hd)}$) were optimised during training. The output of the LSTM, $\vec{m}_t$ is then used to produce predictions of the place cells $\vec{y}_t$ and head direction cells $\vec{z}_t$ by means of a linear decoder network.

The linear decoder consists of three sets of weights and biases: first, weights and biases that map from the LSTM hidden state $\vec{m}_t$ to the linear layer activations $\vec{g}_t \in \mathbb{R}^{512}$. The other two sets of weights map from the linear layer activations $\vec{g}_t$ to the predicted head directions, $\vec{z}_t$, and predicted place cells, $\vec{y}_t$, respectively via softmax functions[33]. Dropout [16] with drop probability 0.5 was applied to each $\vec{g}_t$ unit. Note that there is no intermediary non-linearity in the linear decoder.

Supervised learning loss. The grid cell network is trained to predict the place and head-direction cell ensemble activations, $\vec{c}_t$ and $\vec{h}_t$ respectively, at each time step t. During training, the network was trained in a single environment where the place cell centres were constant throughout. The parameters of the grid cell network are trained by minimising the cross-entropy between the network place cell predictions, $\vec{y}$, and the synthetic place-cells targets, $\vec{c}$, and the cross-entropy between head-direction predictions, $\vec{z}$, and their targets, $\vec{h}$:

$$L(\vec{y}, \vec{z}, \vec{c}, \vec{h}) = -\sum_{i=1}^{N} c_i \log(y_i) - \sum_{j=1}^{M} h_j \log(z_j), \tag{5}$$

Gradients of (5) with respect to the network parameters were calculated using backpropagation through time[34], unrolling the network into blocks of 100 time steps. The network parameters were updated using stochastic-gradient descent (RMSProp[35]), with weight decay[36] for the weights incident upon the bottleneck activations. Hyperparameter values used for training are listed in Table 1.

Gradient clipping In our simulations gradient clipping was used for parameters projecting from the dropout linear layer, g_t, to the place and head-direction cell predictions $\vec{y}_t$ and $\vec{z}_t$. Gradient clipping clips each element of the gradient vector to lie in a given interval $[-g_c, g_c]$. Gradient clipping is an important tool for optimisation in deep and recurrent artificial neural networks where it helps to prevent exploding gradients[37]. Gradient clipping also introduces distortions into the weight updates which help to avoid local minima[38].

Navigation through Deep RL

Environments and Task We assessed the performance of agents on three environments seen by the agent from a first-person perspective in the DeepMind Lab [39] platform.

Custom Environment: Square Arena This comprised a 10×10 square arena - which correspond to 2.5×2.5 meters assuming an agent speed of 15 cm/s (Fig. 2b, c). The arena contained a single, coloured, intra-arena cue whose position and colour changed each episode — as did the texture of the floor, the texture of the walls and the goal location. As in the goal-driven and goal-door environments described below, there were a set of distal cues (i.e. buildings) that paralleled the design of virtual reality environments used in human experiments[24]. These distal cues were rendered at infinity — so as to provide directional but not distance information — and their configuration was consistent across episodes. At the start of each episode the agent (described below) started in a random location and was required to explore in order to find an unmarked goal, paralleling the task of rodents in the classic Morris water maze. The agent always started in the central 6×6 grids (i.e. 1.5×1.5 meters) of the environment. Noise in the velocity input $\vec{u}_t$ was applied throughout training and testing (i.e. Gaussian noise ϵ, with $\mu = 0$ and $\sigma = 0.01$). The action space is discrete (six actions) but affords fine-grained motor control (i.e. the agent could rotate in small increments, accelerate forward/backward/sideways, or effect rotational acceleration while moving).

DeepMind Lab Environments: Goal-Driven and Goal-Doors Goal-driven and Goal-Doors are complex, visually-rich multi-room environments (see Fig. 3a-d). Mazes were formed within an 11×11 grid, corresponding to 2.7 × 2.7 meters, (see below for definition of larger 11×17 mazes).

Mazes were procedurally generated at the beginning of each episode; thus, the layout, wall textures, landmarks (i.e. intra-maze cues on walls) and goal location were different for each episode but consistent within an episode. Distal cues, in the form of buildings rendered at infinity, were as described for the square arena (see above).

The critical difference between goal-driven and goal-doors tasks is that the latter had the additional challenge of stochastic doors within the maze. Specifically, the state of the doors (i.e. open or closed) randomly changed during an episode each time the agent reached the goal. This meant that the optimal path to the goal from a given location changed during an episode – requiring the agent to recompute trajectories.

In both tasks the agent starts at a random location within the maze and its task is to explore to find the goal. The goal in both levels was always represented by the same object (see Fig. 3c). After getting to the goal the agent received a reward of 10 points after which it was teleported to a new random location within the maze. In both levels, episodes lasted a fixed duration of $5,400$ environment steps (90 seconds).

Generalisation on larger environments. We tested the ability of agents trained on the standard environment (11×11) to generalise to larger environments (11×17, corresponding to 2.7 × 4.25 meters). The procedural generation and composition of these environments was done as with the standard environments. Each agent was trained in the 11×11 goal-driven maze for a total of 10^9 environment steps, and the best performing replica (i.e. highest asymptotic performance averaged over 100 episodes in 11×11) was selected for evaluation in the larger maze. Note that the weights of the agent were frozen during evaluation on the larger maze. Evaluation was over 100 episodes of fixed duration $12,600$ environment steps (210 seconds).

Probe mazes to assess shortcut behaviour To test the agent's ability to follow novel, goal-directed routes, we created a series of environments inspired by mazes designed to test the shortcut abilities of rodents.

The first maze is a linearised version of Tolman's sunburst maze (Fig. 4a) used to determined if the agent was able to follow an accurate heading towards the goal when a path became available (see Supplementary Methods for details). In this maze, after reaching the goal, the agent was teleported to the original position with the same heading orientation. Here we tested agents trained in the "goal doors" environments. Specifically, the network weights were held frozen during testing and all the agents were tested for 100 episodes, each one lasting for a fixed duration of $5,400$ environment steps (90 seconds).

The second environment, the double E-maze (Fig. 4d), was designed to test the agent's ability to traverse an entirely novel portion of space (see Supplementary Methods for details). In this maze we had a training and a testing condition. During the former agents were trained as in the other mazes (e.g. goal-driven; training details given below), whereas at test time weights were frozen. The agent always started in the central room (e.g. see Fig. 4d). The maze had stochastic doors with two different configurations, one for the training phase and one for testing phase. During training the state of the doors (i.e. open or closed) randomly changed during an episode each time the agent reached the goal. Critically, during training the corridors presenting the shortest route to the goal (i.e. the ones closer to the central room) were closed at both ends, preventing access or observation of the interior. At test time, after the agent reached the goal the first time, all doors were opened. All the agents were tested for 100 episodes, each one lasting for a fixed duration of $5,400$ environment steps (90 seconds).

Agent Architectures

Architecture for the Grid Cell Agent. The agent architecture (see Extended Data Figure 5) was composed of a visual module, of the grid cell network (described above), and of an actor-critic learner[40]. The visual module was a neural network with input consisting of a three channel (RGB) 64×64 image $\phi \in [-1, 1]^{3\times 84\times 84}$. The image was processed by a convolutional neural network (see Supplementary Methods for the details of the convolutional neural network), which produced embeddings, $\vec{e}$, which in turn were used as input to a fully connected linear layer trained in a supervised fashion to predict place and head-direction cell ensemble activations, $\vec{c}$ and $\vec{h}$ (as specified above), respectively. The predicted place and head direction cell activity patterns were provided as input to the grid network 5% of the time on average, akin to occasional imperfect observations made by behaving animals of salient environmental cues[27]. Specifically, the output of the convolutional network $\vec{\vec{e}}$ is then passed through a masking layer which zeroed the units with a probability of 95%.

The grid cell network of the agent was implemented as in the supervised learning set up except that the LSTM ("GRID LSTM") was not initialised based upon ground truth place cell activations but rather set to zero. The input to the grid cell network were the two translational velocities, u and v, as in DeepMind Lab it is possible to move in a direction different from the facing direction, and the sine and cosine of the angular velocity, $\dot{\varphi}$, (these velocities are provided by DeepMind Lab) — and additionally the $\vec{y}$ and $\vec{z}$ output by the vision module. In contrast to the supervised learning case, here the grid cell network had to use $\vec{y}$ and $\vec{z}$ to learn how to reset its internal state each time it was teleported to an arbitrary location in the environment (e.g. after visit to goal). As in the supervised learning experiments described above, the configuration of place fields (i.e. location of place field centres in the 11×11 environments, "goal-driven" and "goal-doors", 10×10 square arena, and 13×13 double E) were constant throughout training (i.e. across episodes).

For the actor-critic learner the input was a three channel (RGB) 64×64 image $\phi_t \in [-1, 1]^{3 \times 84 \times 84}$, which was processed by a convolutional neural network followed by a fully connected layer (see Supplementary Methods for the details of the convolutional neural network). The output of the fully connected layer of the convolutional network $\vec{e}_t^1$ was then concatenated with the reward r_t, the previous action a_{t-1}, the current "grid code", $\vec{g}_t$, goal "grid code", $\vec{g}_*$ (i.e. linear layer activations observed last time the goal was reached) — or zeros if the goal had not yet been reached in the episode. Note we refer to these linear layer activations as "grid codes" for brevity, even though units in this layer comprise also units resembling head direction cells, and border cells (e.g. see Extended Figure 6a). This concatenated input was provided to an LSTM with 256 units. The LSTM had 2 different outputs. The first output, the actor, is a linear layer with 6 units followed by a softmax activation function, that represents a categorical distribution over the agent's next action. The second output, the critic, is a single linear unit that estimates the value function. Note that we refer to this as the ”policy LSTM” for brevity, even though it also outputs the value function.

Comparison agents We compared the performance of the grid cell agent against two agents specifically because they use a different representational scheme for space (i.e. place cell agent, place cell prediction agent), and relate to theoretical models of goal-directed navigation from the neuroscience literature (e.g. [41,42]). We also compared the grid cell agent against a baseline deep RL agent, Asynchronous Advantage Actor-Critic (A3C)[40].

Place cell agent. The place cell agent architecture is shown in Extended Data Figure 8b. In contrast to the grid cell agent, the place cell agent used ground truth information: specifically, the ground-truth place, $\vec{c}_t$, and head-direction, $\vec{h}_t$, cell activations (as described above). These activity vectors were provided as input to the policy LSTM in an analogous way to the provision of grid codes in the grid cell agent.

Specifically, the output of the fully connected layer of the convolutional network $\vec{e}_t$ was concatenated with the reward r_t, the previous action a_{t-1}, the ground-truth current place code, $\vec{c}_t$, and current head-direction code, $\vec{h}_t$ — together with the ground truth goal place code, $\vec{c}_*$, and ground truth head direction code, $\vec{h}_*$, observed last time the goal was reached — or zeros if the goal had not yet been reached in the episode (see Extended Data Figure 8b). The convolutional network had the same architecture described for the grid cell agent.

Place cell prediction agent. The architecture of the place cell prediction agent (Extended Data Figure 9a) is similar to the grid cell agent described above: the key difference is the nature of the input provided to the policy LSTM as described below. The place cell prediction agent had a grid cell network — with the same parameters as that of the grid cell agent. However, instead of using grid codes from the linear layer of the grid network $\vec{g}$, as input for the policy LSTM (i.e. as in the grid cell agent), we used the predicted place cell population activity vector $\vec{y}$ and the predicted head direction population activity vector $\vec{z}$ (i.e. the activations present on the output place and head direction unit layers of the grid cell network at each timestep) (see Supplementary Methods).

The critical difference between the place cell agent and the place cell prediction agent (see Extended Data Figure 8b and 9a respectively) is that the former used ground truth information (i.e. place and head direction cell activations for current location and goal location) - whereas the latter used the population activity produced across the output place and head direction cell layers (i.e. for current location and goal location) by the linear layer of the same grid network as utilised by the grid cell agent.

A3C We implemented the asynchronous advantage actor-critic architecture described in[40] with convolutional network having the same architecture described for the grid cell agent (Extended Data Figure 8a).

Other Agents We also assessed the performance of two deep RL agents with external memory (Extended Data Figure 9b), which served to establish the challenging nature of the multi-compartment environments (goal-doors and goal-driven). First, we implemented a memory network agent ("NavMemNet") consisting of the FRMQN architecture[3] but instead of Q-learning we used the Asynchronous Advantage Actor-Critic (A3C) algorithm described below. Further, the input to memory was generated as an output from the LSTM controller (Extended Data Figure 9b), rather than constituting embeddings from the convolutional network (i.e. as in[3]). The convolutional network had the same architecture described for the grid cell agent and the memory was formed of 2 banks (keys and values), each one with 1350 slots.

Second, we implemented a Differentiable Neural Computer ("DNC") agent which uses content-based retrieval and writes to the most recently used or least recently used memory slot.[43]

Training algorithms We used the Asynchronous Advantage Actor-Critic (A3C) algorithm[40], which implements a policy, $\pi(a|s,\theta)$, and an approximation to its value function, $V(s,\theta)$, using a neural network parameterised by θ. A3C adjusts the network parameters using n-step lookahead values, $\hat{R}_t = \sum_{i=0\ldots n-1} \gamma^i r_{t+i} + \gamma^n V(s_{t+n},\theta)$, to minimise: $\mathcal{L}_{\text{A3C}} = \mathcal{L}_\pi + \alpha\mathcal{L}_{\text{V}} + \beta\mathcal{L}_{\text{H}}$, where $\mathcal{L}_\pi = -\mathbb{E}_{s_t\sim\pi}\left[\hat{R}_t\right]$, $\mathcal{L}_{\text{V}} = \mathbb{E}_{s_t\sim\pi}\left[\left(\hat{R}_t - V(s_t,\theta)\right)^2\right]$, $\mathcal{L}_{\text{H}} = -\mathbb{E}_{s_t\sim\pi}\left[H(\pi(\cdot|s_t,\theta))\right]$. Where $\mathcal{L}_{\text{H}}$ is a policy entropy regularisation term (see Supplementary Methods for details of the reinforcement learning approach). The grid cell network and the vision module were trained with the same loss reported for supervised learning: $\mathcal{L}(\vec{y},\vec{z},\vec{c},\vec{h}) = -\sum_{i=1}^{N} c_i \log(y_i) - \sum_{j=1}^{M} h_j \log(z_j)$

Agent training details. We follow closely the approach of[40]. Each experiment used 32 actor-critic learner threads running on a single CPU machine. All threads applied updates to their gradients every 4 actions (i.e. action repeat of 4) using RMSProp with shared gradient statistics[40]. All the experiments were run for a total of 10^9 environment steps.

In architectures where the grid cell network and the vision module were present we used a shared buffer [44,45] where we stored the agents experiences at each time-step, $e_t = (\phi_t, u_t, v_t)$, collected over many episodes. All the 32 actor-critic workers were updating the same shared buffer which had a total size of $20e6$ slots. The vision module was trained with mini batches of size 32 frames ($\vec{\phi}$) sampled randomly from the replay buffer. The grid cell network was trained with mini batches of size 10, randomly sample from the buffer, each one comprising a sequence of 100 consecutive observations, $[\vec{\phi}, \vec{u}, \vec{v}]$. These mini batches were firstly forwarded through the vision module to get $\vec{c}$, and $\vec{h}$, which were then passed trough a masking layer which masked them to 0 with a probability of 95% (i.e. as described above in section on grid cell architecture). The output of this masking layer was then concatenate with $\vec{u}, \ \ \vec{v}, \ \ \vec{sin}\dot{\varphi}, \ \ \vec{cos}\dot{\varphi}$, which were then used as inputs to the grid network, as previously described (see Extended Data Figure 5 for details). Both networks were trained using one single thread, one to train the vision module and another to train the grid network (so in total we used 34 threads). Also, there was no gradient sharing between the actor-critic learners, the vision module and the grid network.

The hyperparameters of the grid cell network were kept fixed across all the simulations and were derived from the best performing network in the supervised learning experiments. For the hyperparameter details of the vision module, the grid network and the actor-critic learner please refer to Table 2. For each of the agents in this paper, 60 replicas were run with hyperparameters sampled from the same interval (see Table 2) and different initial random seeds.

Details for lesion experiment To conduct a lesioning experiment in the agent we trained the grid cell agent with dropout applied on the goal grid code input $\vec{g}_*$. Specifically, every 100 training steps we generated a random mask to silence 20% of the units in the goal grid code ($\vec{g}_*$) - i.e. units were zeroed. This procedure was implemented to ensure that the policy LSTM would become robust through training to receiving a lesioned input (i.e. would not catastrophically fail), and still be able to perform the task.

We then selected the agent with the best performance over 100 episodes, and we computed the grid score of all units found in $\vec{g}$. The critical comparison to test the importance of grid-like units to vector-based navigation was as follows. In one condition we ran 100 testing episodes where we silenced the 25% units in $\vec{g}_*$ with the highest grid scores. In the other condition, we ran 100 testing episodes with the same agent with 25% random units in $\vec{g}_*$ silenced. In this second case we ensured head direction cells with a resultant vector length of more than 0.47 were not silenced, to preserve crucial head direction signals. We then compared the performance, and representation of metrics relating to vector-based navigation, of the agents under these two conditions.

Details of experiment using ”fake” goal grid code To demonstrate that the goal grid code provided sufficient information to enable the agent to navigate to an arbitrary location we took an agent trained in the square arena, we froze the weights and we ran it in the same square arena for $5,400$ steps. Critically, after the $6th$ time that the agent reach the goal, we sampled the grid code from a random point that the agent visited in the environment (called fake goal grid code). We then substituted the true goal grid code with this fake goal grid code, to show that this would be sufficient to direct the agent to a location where there was no actual goal.

Agent Performance For evaluating agent performance during training (as in Fig. 2f, Fig. 3e,f) we selected the 30 replicas (out of 60) which had the highest average cumulative reward across 100 episodes. Also we assessed the robustness of the architecture over different initial random seeds and the hyperparameters in Table 2 by calculating the area under the curve (AUC). To plot the AUC we ran 60 replicas with hyperparameters sampled from the same interval (see Table 2) and different initial random seeds (Extended Data Figure 7a-c).

Neuroscience-based analyses of network units

Generation of activity maps Spatial (ratemaps) and directional activity maps were calculated for individual units as follows. Each point in the trajectory was assigned to a specific spatial and directional bin based on its location and direction of facing. Spatial bins were defined as a 32×32 square grid spanning each environment and directional bins as 20 equal width intervals. Then, for each unit, the mean activity over all the trajectories points assigned to that bin was found. These values were displayed and analysed further without additional smoothing.

Inter-trial stability For each unit the reliability of spatial firing between baseline trials was assessed by calculating the spatial correlation between pairs of rate maps taken at 2 different logging steps in training ($t = 2e5$; $t' = 3e5$). The total training time was $3e5$ so the points were selected with enough time difference to minimise the chances of finding random correlations. The Pearson product moment correlation coefficient was calculated between equivalent bins in the two trials and unvisited bins were excluded from the measure.

Quantification of spatial activity Where possible, we assessed the spatial modulation of units using measures adopted from the neuroscience literature. The hexagonal regularity and scale of grid-like patterns were quantified using the gridness score[18,20] and grid scale[20], measures derived from the spatial autocorellogram[20] of each unit's ratemap. Similarly, the degree of directional modulation exhibited by each unit was assessed using the length of the resultant vector[46] of the directional activity map. Finally, the propensity of units to fire along the boundaries of the environment was quantified with the border score[47].

The gridness and border scores exhibited by units in the linear layer were benchmarked against the 95th percentile of null distributions obtained using a permutation procedure (spatial field shuffle[48]) applied to each unit's ratemap. This shuffling procedure aimed to preserve the local topography of fields within each ratemap while distributing the fields themselves at random[48]. The means, over units, of the thresholds obtained were gridness $>$ 0.37 and border score $>$ 0.50. Units exceeding these thresholds were considered to be grid-like and border-like, respectively. To identify directionally modulated cells we applied Rayleigh tests of directional uniformity to the binned directional activity maps. A unit was considered to be directionally modulated if the null hypothesis of uniform was rejected at the $\alpha = 0.01$ level - corresponding to units with resultant vector length in excess of 0.47 (See Supplementary Methods for further details).

Clustering of scale in grid-like units To determine if grid-like units exhibited a tendency to cluster around specific scales we applied two methods. First, following[22], to determine if the scales of grid-like units (gridness $>$ 0.37, 129/512 units) followed a continuous or discrete distribution we calculated the discreteness measure[22] of the distribution of their scales (see Supplementary Methods). The discreteness score of the real data was found to exceed that of all of the 500 shuffles. Second, to characterise the number and location of scale clusters, the distribution of scales from grid-like units was fit with Gaussian mixture distributions, 3 components were found to provide the most parsimonious fit, indicating the presence of 3 scale clusters. (See Supplementary Methods for further details.)

Multivariate decoding of representation of metric quantities within LSTM To test whether the grid agent learns to use the predicted vector based navigation (VBN) metric codes, we recorded the activation from the hidden units of the the Policy LSTM layer while the agent navigated 200 hundred episodes in the land maze. We used L2-regularized (ridge) regression to decode Euclidean distance and allocentric direction to the goal (see Supplementary Methods for full decoding details). We specifically focussed on twelve steps (steps 9-21) during the early portion of navigation, but after the agent has had time to accurately self-localize. It is this early period after the agent has reached the goal for the first time where a VBN strategy should be most effective. We conducted the same analysis on the place cell agent control which is not predicted to use vector-based navigation as efficiently. The decoding accuracy was measured as the correlation between predicted and actual metric values in held-out data. Decoding accuracy was compared across different agents by assessing the difference in decoding correlations between the agents. A bootstrap method (using 10,000 samples) was used to computed a 95% confidence interval on this correlation difference, and these are reported for each comparison. The same approach was used to decode and compare these two metrics in the lesioned agents on the land maze. Finally, to explore VBN metrics in a more complex environment, the same method was applied to the goal-driven task. In this case we also investigated metric decoding in the control A3C agent.

Data availability statement All reinforcement learning tasks described throughout the paper were built using the publicly available DeepMind Lab platform (https://github.com/deepmind/lab). We expect to release this set of tasks through this platform in the near future.

Code availability statement We will release the code for the supervised learning experiments within the next six months. The codebase for the deep RL agents makes use of proprietary components, and we are unable to publicly release this code. However, all experiments and agents are described in sufficient detail to allow independent replication.

31. Raudies, F. & Hasselmo, M. E. Modeling boundary vector cell firing given optic flow as a cue. *PLoS computational biology* **8**, e1002553 (2012).

32. Hochreiter, S. & Schmidhuber, J. Long short-term memory. *Neural computation* **9**, 1735–1780 (1997).

33. Bridle, J. S. Training stochastic model recognition algorithms as networks can lead to maximum mutual information estimation of parameters. In Touretzky, D. S. (ed.) *Advances in Neural Information Processing Systems 2*, 211–217 (Morgan-Kaufmann, 1990).

34. Elman, J. L. & McClelland, J. L. Exploiting lawful variability in the speech wave. *Invariance and variability in speech processes* **1**, 360–380 (1986).

35. Tieleman, T. & Hinton, G. Lecture 6.5—RmsProp: Divide the gradient by a running average of its recent magnitude. COURSERA: Neural Networks for Machine Learning (2012).

36. MacKay, D. J. A practical bayesian framework for backpropagation networks. *Neural computation* **4**, 448–472 (1992).

37. Pascanu, R., Mikolov, T. & Bengio, Y. On the difficulty of training recurrent neural networks. *ICML (3)* **28**, 1310–1318 (2013).

38. Ackley, D. H., Hinton, G. E. & Sejnowski, T. J. A learning algorithm for boltzmann machines. *Cognitive science* **9**, 147–169 (1985).

39. Beattie, C. *et al.* Deepmind lab. *CoRR* **abs/1612.03801** (2016). URL `http://arxiv.org/abs/1612.03801`.

40. Mnih, V. *et al.* Asynchronous methods for deep reinforcement learning. In *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, 1928–1937 (2016).

41. Touretzky, D. S., Redish, A. D. *et al.* Theory of rodent navigation based on interacting representations of space. *Hippocampus* **6**, 247–270 (1996).

42. Foster, D., Morris, R., Dayan, P. *et al.* A model of hippocampally dependent navigation, using the temporal difference learning rule. *Hippocampus* **10**, 1–16 (2000).

43. Graves, A. *et al.* Hybrid computing using a neural network with dynamic external memory. *Nature* **538**, 471–476 (2016).

44. Mnih, V. *et al.* Human-level control through deep reinforcement learning. *Nature* **518**, 529–533 (2015).

45. Lin, L.-J. Reinforcement learning for robots using neural networks. Tech. Rep., Carnegie-Mellon Univ Pittsburgh PA School of Computer Science (1993).

46. Knight, R. *et al.* Weighted cue integration in the rodent head direction system. *Philosophical Transactions of the Royal Society of London B: Biological Sciences* **369**, 20120512 (2014).

47. Solstad, T., Boccara, C. N., Kropff, E., Moser, M.-B. & Moser, E. I. Representation of geometric borders in the entorhinal cortex. *Science* **322**, 1865–1868 (2008).

48. Barry, C. & Burgess, N. To be a grid cell: Shuffling procedures for determining gridness. *BiorXiv* (2017).

There is Supplemental Information that contains additional results, discussion and details about the Methods.

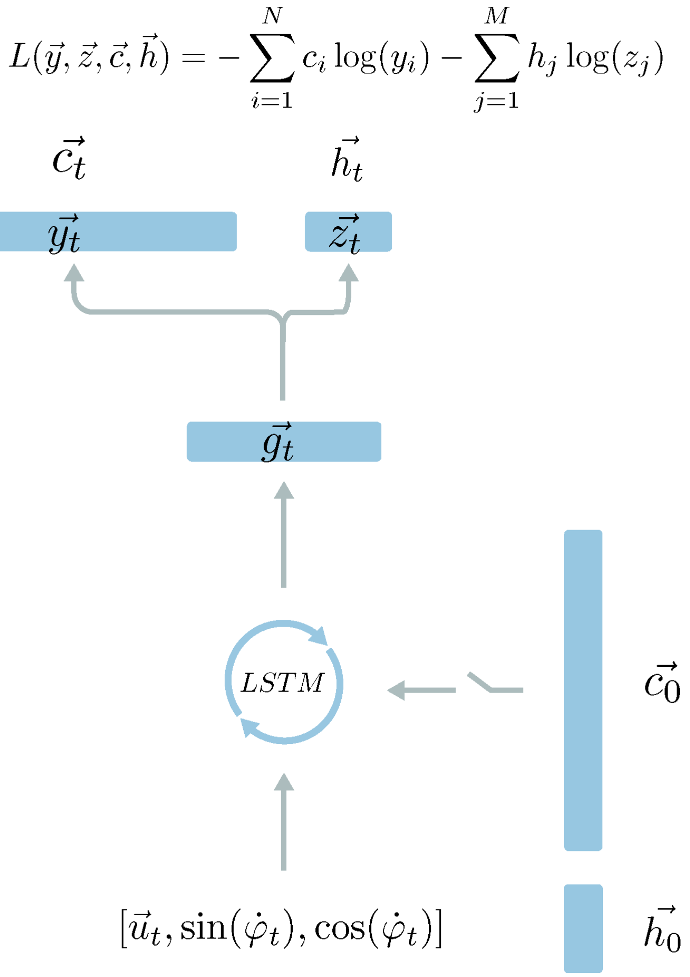
Acknowledgements We thank Max Jaderberg, Volodymyr Mnih, Adam Santoro, Tom Schaul, Kim Stachenfeld and Jason Yosinski for discussions; Matthew Botvinick, and Jane Wang for comments on an earlier version of the manuscript . C.Ba. funded by Royal Society and Wellcome Trust.

Competing Interests The authors declare that they have no competing financial interests.

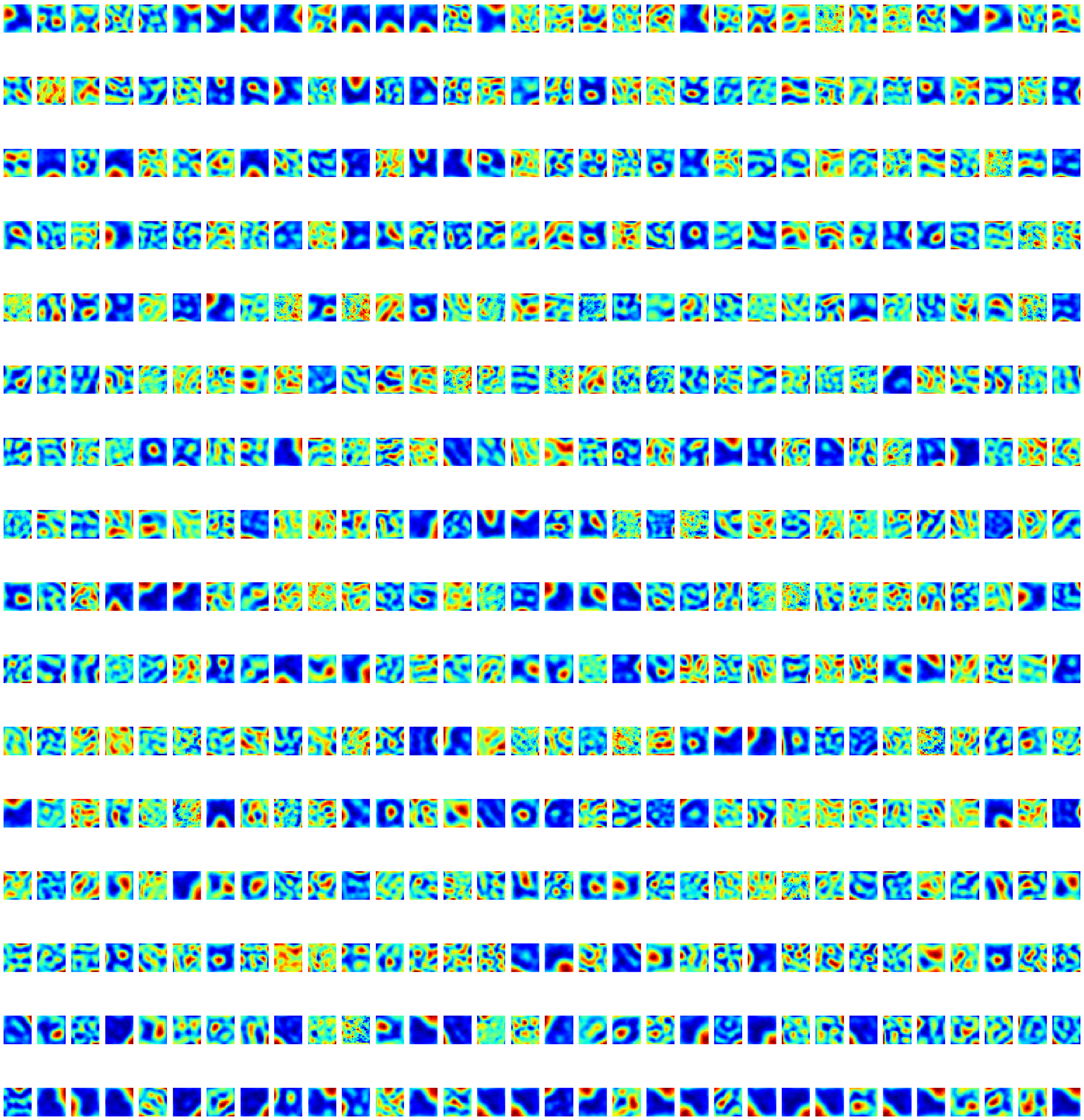
Correspondence Correspondence and requests for materials should be addressed to Andrea Banino, Caswell Barry, Dharshan Kumaran (email: abanino@google.com, caswell.barry@ucl.ac.uk, dkumaran@google.com).

Author Contributions Conceived project. A.B, D.K., C.Ba., R.H., P.M., B.U., Contributed ideas to experiments. A.B., D.K., C.Ba., B.U., R.H., T.L., C.Bl., P.M., A.P., T.D., J.M., K.K., N.R., G.W., R.G., D.H., R.P. Performed experiments and analysis: A.B., C.Ba., B.U., M.C., T.L., H.S., A.P., B.Z, F.V. Development of testing platform and environments. C.Be., S.P., R.H., T.L., G.W., D.K., A.B., B.U., D.H. Human expert tester. A.S. Managed project. D.K, R.H., A.B., H.K., S.G., D.H. Wrote paper. D.K., A.B., C.Ba., T.L., C.Bl., B.U., M.C., A.P., R.H., N.R., K.K., D.H.

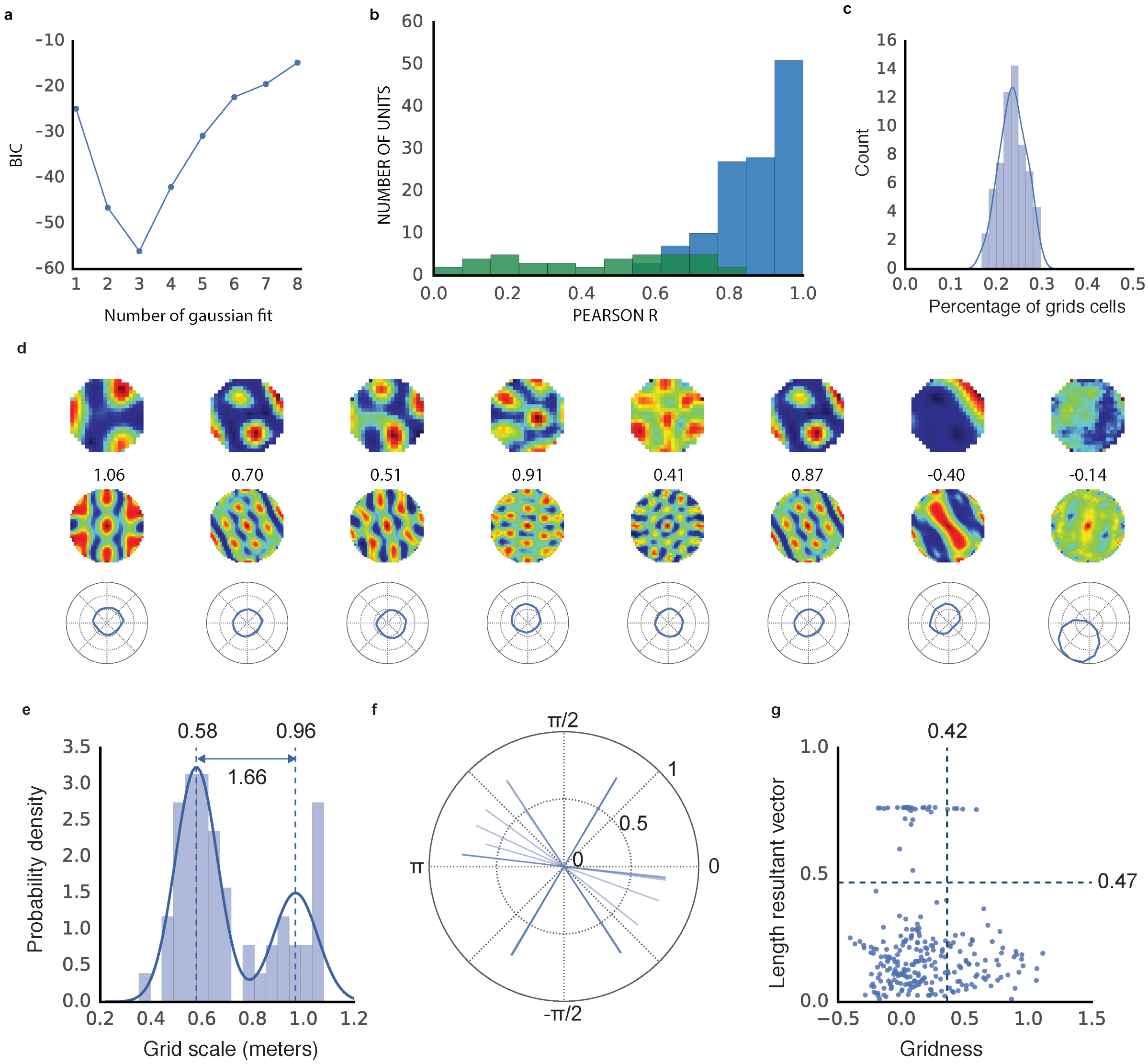
Extended data for *Vector-based Navigation using Grid-like Representations in Artificial Agents.*


$L(\vec{y}, \vec{z}, \vec{c}, \vec{h}) = -\sum_{i=1}^{N} c_i \log(y_i) - \sum_{j=1}^{M} h_j \log(z_j)$
$\vec{c}_t$
$\vec{h}_t$
$\vec{y}_t$
$\vec{z}_t$
$\vec{g}_t$
$LSTM$
$\vec{c}_0$
$[\vec{u}_t, \sin(\dot{\varphi}_t), \cos(\dot{\varphi}_t)]$
$\vec{h}_0$

Extended Data Figure 1: **Network architecture in the supervised learning experiment.** The recurrent layer of the grid cell network is an LSTM with 128 hidden units. The recurrent layer receives as input the vector $[v_t, \sin(\dot{\varphi}_t), \cos(\dot{\varphi}_t)]$. The initial cell state and hidden state of the LSTM, $\vec{l}_0$ and $\vec{m}_0$ respectively, are initialised by computing a linear transformation of the ground truth place $\vec{c}_0$ and head-direction $\vec{h}_0$ activity at time 0. The output of the LSTM is followed by a linear layer on which dropout is applied. The output of the linear layer, $\vec{g}_t$, is linearly transformed and passed to two softmax functions that calculate the predicted head direction cell activity, $\vec{z}_t$, and place cell activity, $\vec{y}_t$, respectively. We found evidence of grid-like and head direction-like units in the linear layer activations $\vec{g}_t$.



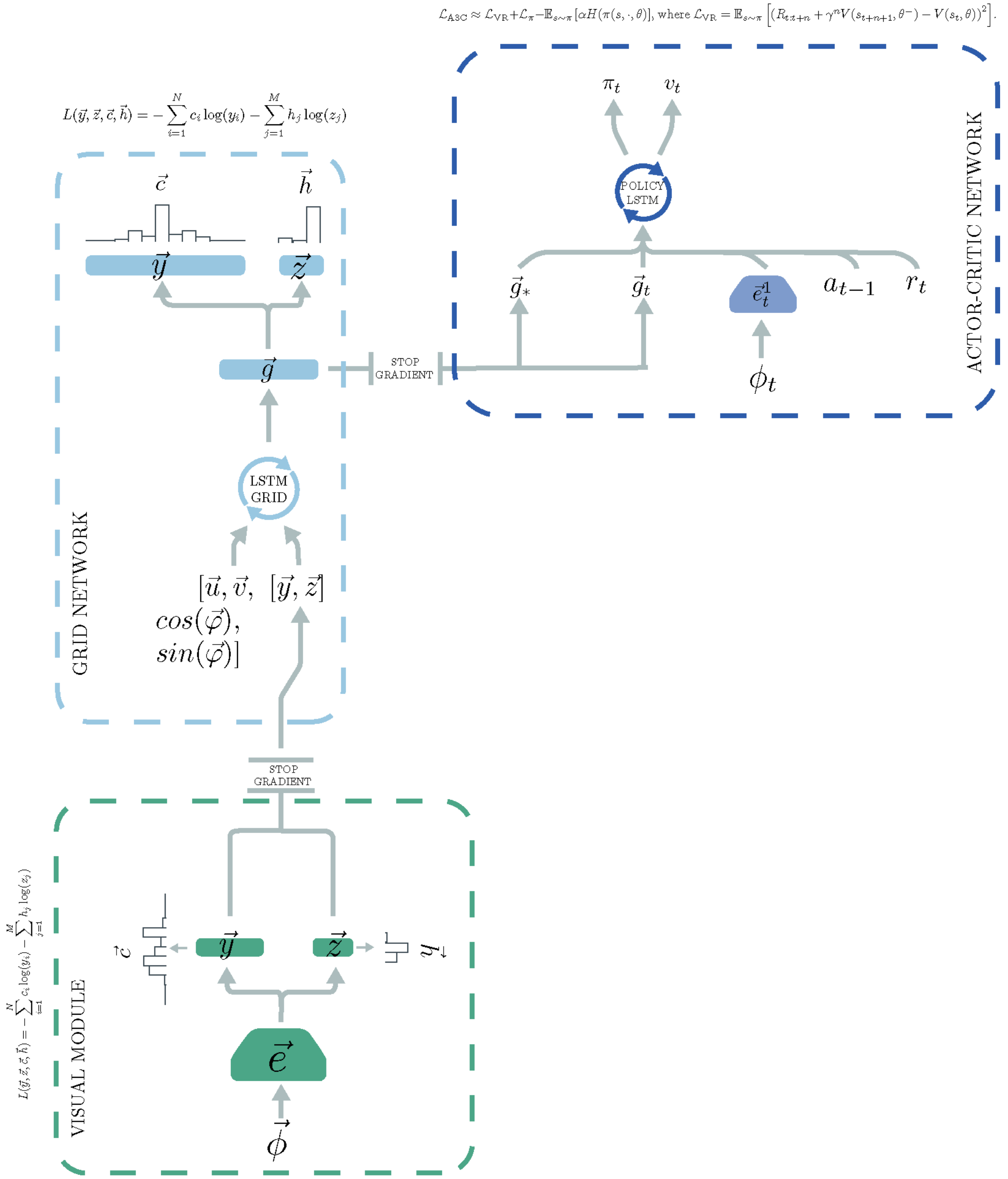
Extended Data Figure 2: **Linear layer spatial activity maps from the supervised learning experiment.** Spatial activity plots for all 512 units in the linear layer $\vec{g}_t$. Units exhibit spatial activity patterns resembling grid cells, border cells, and place cells — head direction tuning was also present but is not shown.


a
-10
-20
-30
-40
-50
-60
BIC
1 2 3 4 5 6 7 8
Number of gaussian fit
b
60
50
40
30
20
10
0
NUMBER OF UNITS
0.0 0.2 0.4 0.6 0.8 1.0
PEARSON R
c
16
14
12
10
8
6
4
2
0
Count
0.0 0.1 0.2 0.3 0.4 0.5
Percentage of grids cells
d
1.06
0.70
0.51
0.91
0.41
0.87
-0.40
-0.14
e
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
Probability density
0.58
0.96
1.66
0.2 0.4 0.6 0.8 1.0 1.2
Grid scale (meters)
f
π/2
1
0.5
0
π
0
-π/2
g
1.0
0.5
0.0
0.42
0.47
Length resultant vector
−0.5 0.0 0.5 1.0 1.5
Gridness

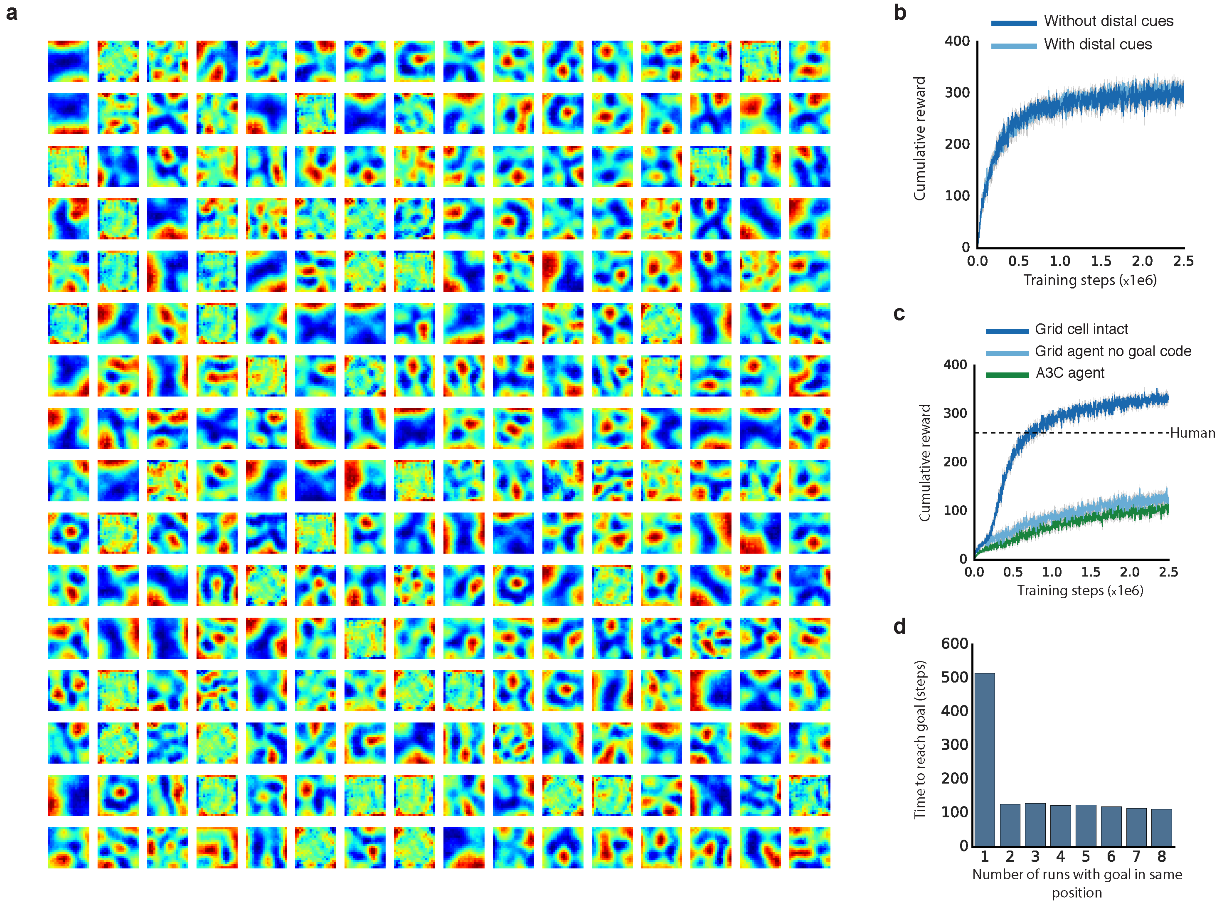
Extended Data Figure 3: **Characterization of grid-like units in Square environment and Circular environment.** a) The scale (assessed from the spatial autocorrelogram of the ratemaps) of grid-like units exhibited a tendency to cluster at specific values. The number of distinct scale clusters was assessed by sequentially fitting Gaussian mixture models with 1 to 8 components. In each case, the efficiency of the fit (likelihood vs. number of parameters) was assessed using Bayesian information criterion (BIC). BIC was minimized with three Gaussian components indicating the presence of three distinct scale clusters. b) Spatial stability of units in the linear layer of the supervised network was assessed using spatial correlations — bin-wise Pearson product moment correlation between spatial activity maps (32 spatial bins in each map) generated at 2 different points in training, $t = 2e5$ and $t' = 3e5$ training steps. That is, $\frac{2}{3}$ of the way through training and the end of training, respectively. This separation was imposed to minimise the effect of temporal correlations and to provide a conservative test of stability. Grid-like units (gridness $>$ 0.37) blue, directionally modulated units (resultant vector length $>$ 0.47) green. Grid-like units exhibit high spatial stability, while directionally modulated units do not. c) Robustness of the grid representation to starting conditions. The network was retrained 100 times with the same hyperparamters but different random seeds controling the initialisation of network weights, $\vec{c}$ and $\vec{h}$. Populations of grid-like units (gridness $>$ 0.37) were found to appear in all cases, the average proportion of grid-like units being 23$\%$ (SD of 2.8%). d) **circular environment**: the supervised network was also trained in a circular environment (diameter = 2.2m). As before, units in the linear layer exhibited spatially tuned responses resembling grid, border, and head direction cells. Eight units are shown. Top, ratemap displaying activity binned over location. Middle, spatial autocorrelogram of the ratemap, gridness[20] is indicated above. Bottom, polar plot of activity binned over head direction. e) Spatial scale of grid-like units ($n = 56$ (21.9%)) is clustered. Distribution is best fit by a mixture of 2 Gaussians (centres 0.58 & 0.96m, ratio = 1.66). f) Distribution of directional tuning for 31 most directionally active units, single line for each unit indicates length and orientation of resultant vector[46] g) Distribution of gridness and directional tuning. Dashed lines indicate 95$\%$ confidence interval derived from shuffling procedure (500 permutations), 5 grid units (9$\%$) exhibit significant directional modulation.

a

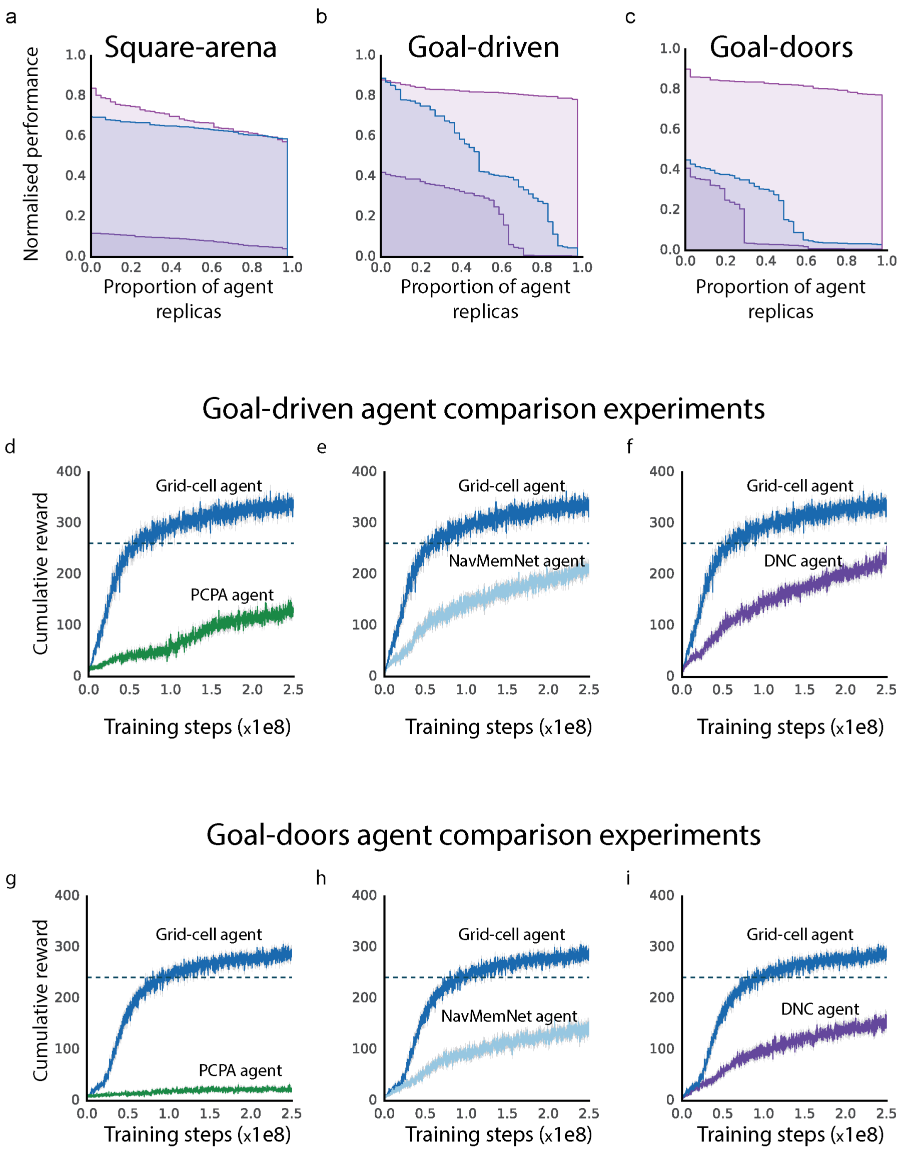
Extended Data Figure 4: **Grid-like units did not emerge in the linear layer when dropout was not applied.** Linear layer spatial activity maps (n=512) generated from a supervised network trained without dropout. The maps do not exhibit the regular periodic structure diagnostic of grid cells.


$\mathcal{L}_{\mathrm{A3C}} \approx \mathcal{L}_{\mathrm{VR}} + \mathcal{L}_{\pi} - \mathbb{E}_{s\sim\pi}[\alpha H(\pi(s,\cdot,\theta)]$, where $\mathcal{L}_{\mathrm{VR}} = \mathbb{E}_{s\sim\pi}\left[(R_{t:t+n} + \gamma^{n}V(s_{t+n+1},\theta^{-}) - V(s_t,\theta))^2\right]$.
π_t
v_t
POLICY LSTM
$\vec{g}_*$
$\vec{g}_t$
$\vec{e}_t^1$
a_{t-1}
r_t
ϕ_t
ACTOR-CRITIC NETWORK
STOP GRADIENT
$L(\vec{y},\vec{z},\vec{c},\vec{h}) = -\sum_{i=1}^{N} c_i \log(y_i) - \sum_{j=1}^{M} h_j \log(z_j)$
$\vec{c}$
$\vec{h}$
$\vec{y}$
$\vec{z}$
$\vec{g}$
LSTM GRID
$[\vec{u},\vec{v},$
$cos(\vec{\varphi}),$
$sin(\vec{\varphi})]$
$[\vec{y},\vec{z}]$
GRID NETWORK
STOP GRADIENT
$L(\vec{y},\vec{z},\vec{c},\vec{h}) = -\sum_{i=1}^{N} c_i \log(y_i) - \sum_{j=1}^{M} h_j \log(z_j)$
$\vec{c}$
$\vec{y}$
$\vec{z}$
$\vec{h}$
$\vec{e}$
$\vec{\phi}$
VISUAL MODULE

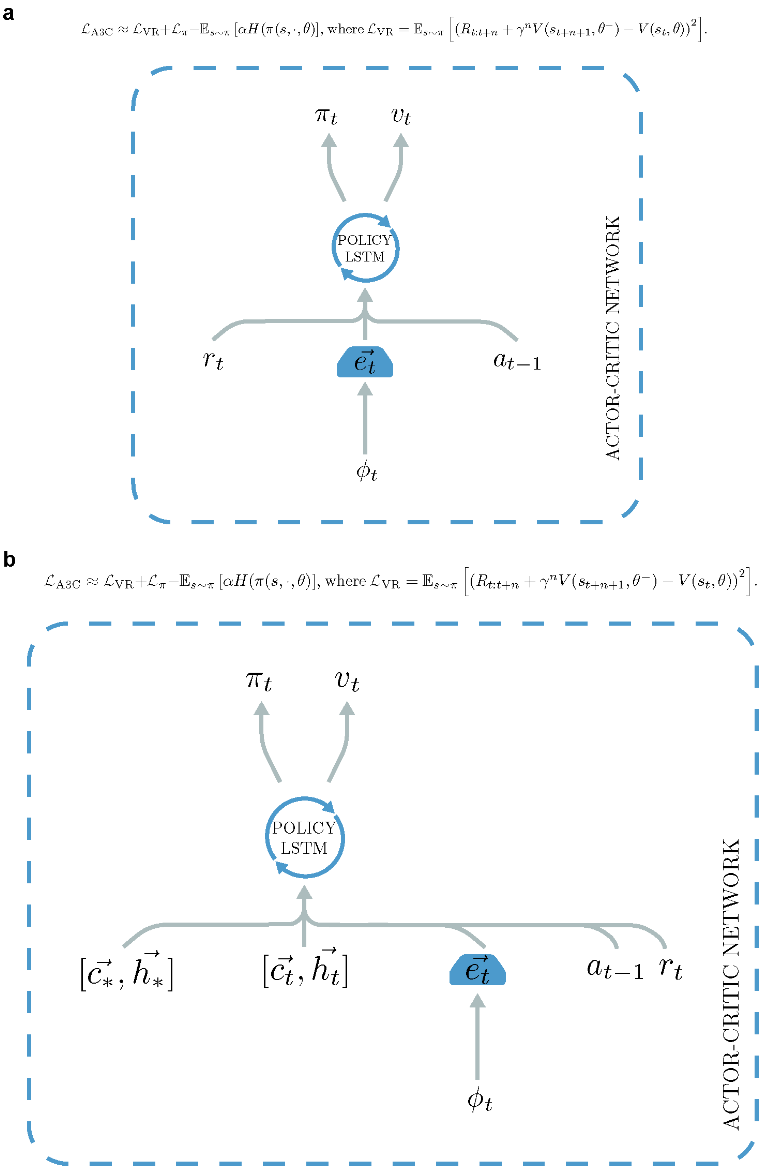
Extended Data Figure 5: **Architecture of the grid cell agent.** The architecture of the supervised network (grid network, light blue dashed) was incorporated into a larger deep RL network, including a visual module (green dashed) and an actor critic learner (based on A3C [40]; dark blue dashed). In this case the supervised learner does not receive the ground truth $\vec{c}_0$ and $\vec{h}_0$ to signal its initial position, but uses input from the visual module to self-localize after placement at a random position within the environment. Visual module: since experimental evidence suggests that place cell input to grid cells functions to correct for drift and anchor grids to environmental cues[21,27], visual input was processed by a convolutional network to produce place cell (and head direction cell) activity patterns which were used as input to the grid network. The output of the vision module was only provided 5% of the time to the grid network; see Methods for implementational details), akin to occasional observations made by behaving animals of salient environmental cues[27]. The output of the vision module was concatenated with $\vec{u},\ \ \vec{v},\ \ \vec{sin\varphi},\ \ \vec{cos\varphi}$ to form the input to the GRID LSTM, which is the same network as in the supervised case (see Methods and Extended Data Figure 1). The actor critic learner (light blue dashed) receives as input the concatenation of $\vec{e}^1_t$ produced by a convolutional network with the reward r_t, the previous action $a_t - 1$, the linear layer activations of the grid cell network $\vec{g}_t$ ("current grid-code"), and the linear layer activations observed last time the goal was reached, $\vec{g}_*$. $\vec{g}_*$ ("goal grid-code"), which is set to zeros if the goal has not been reached in the episode. The fully connected layer was followed by an LSTM with 256 units. The LSTM has 2 different outputs. The first output, the actor, is a linear layer with 6 units followed by a softmax activation function, that represents a categorical distribution over the agent's next action $\vec{\pi}_t$. The second output, the critic, is a single linear unit that estimates the value function v_t.


a
b
Without distal cues
With distal cues
Cumulative reward
400
300
200
100
0
0.0 0.5 1.0 1.5 2.0 2.5
Training steps (x1e6)
c
Grid cell intact
Grid agent no goal code
A3C agent
Cumulative reward
400
300
200
100
0
Human
0.0 0.5 1.0 1.5 2.0 2.5
Training steps (x1e6)
d
Time to reach goal (steps)
600
500
400
300
200
100
0
1 2 3 4 5 6 7 8
Number of runs with goal in same position

Extended Data Figure 6: **Characterisation of grid-like representations and robustness of performance for the grid cell agent in the square "land maze" environment.** a) Spatial activity plots for the 256 linear layer units in the agent exhibit spatial patterns similar to grid, border, and place cells. b) Cumulative reward indexing goal visits per episode (goal = 10 points) when distal cues are removed (dark blue) and when distal cues are present (light blue) — performance is unaffected, hence dark blue largely obscures light blue. Average of 50% best agent replicas (n=32) plotted (see Methods). The gray band displays the 68% confidence interval based on 5000 bootstrapped samples. c) Cumulative reward per episode when no goal code was provide (light blue) and when goal code was provided (dark blue). When no goal code was provided the agent performance fell to that of the baseline deep RL agent (A3C) (100 episodes average score ”no goal code” = 123.22 vs. A3C = 112.06 ,effect size = 0.21, 95% CI [0.18, 0.28]). Average of 50% best agent replicas (n=32) plotted (see Methods). The gray band displays the 68% confidence interval based on 5000 bootstrapped samples. d) After locating the goal for the first time during an episode the agent typically returned directly to it from each new starting position, showing decreased latencies for subsequent visits, paralleling the behaviour exhibited by rodents.


a
Square-arena
b
Goal-driven
c
Goal-doors
Normalised performance
0.0
0.2
0.4
0.6
0.8
1.0
Proportion of agent replicas
Goal-driven agent comparison experiments
d
e
f
Cumulative reward
0
100
200
300
400
Grid-cell agent
PCPA agent
NavMemNet agent
DNC agent
0.0
0.5
1.0
1.5
2.0
2.5
Training steps (x1e8)
Goal-doors agent comparison experiments
g
h
i

Extended Data Figure 7: **Robustness of grid cell agent and performance of other agents.** a-c) AUC performance gives the robustness to hyperparameters (i.e. learning rate, baseline cost, entropy cost - see Table 2 in Supplementary Methods for details of the range) and seeds (see Methods). For each environment we run 60 agent replicas (see Methods). Light purple is the grid agent, blue is the place cell agent and dark purple is A3C. a) Square arena b) Goal-driven c) Goal Doors. In all cases the grid cell agent shows higher robustness to variations in hyperparameters and seeds. d-i Performance of place cell prediction/NavMemNet/DNC agents (see Methods) against grid cell agent. Dark blue is the grid cell agent (Extended Data Figure 5), green is the place cell prediction agent (Extended Data Figure 9a), purple is the DNC agent, light blue is the NavMemNet agent (Extended Data Figure 9b). The gray band displays the 68% confidence interval based on 5000 bootstrapped samples. d-f) Performance in goal-driven. g-i) Performance in goal-doors. Note that the performance of the place cell agent (Extended Data Figure 8b, lower panel) is shown in Figure 3.



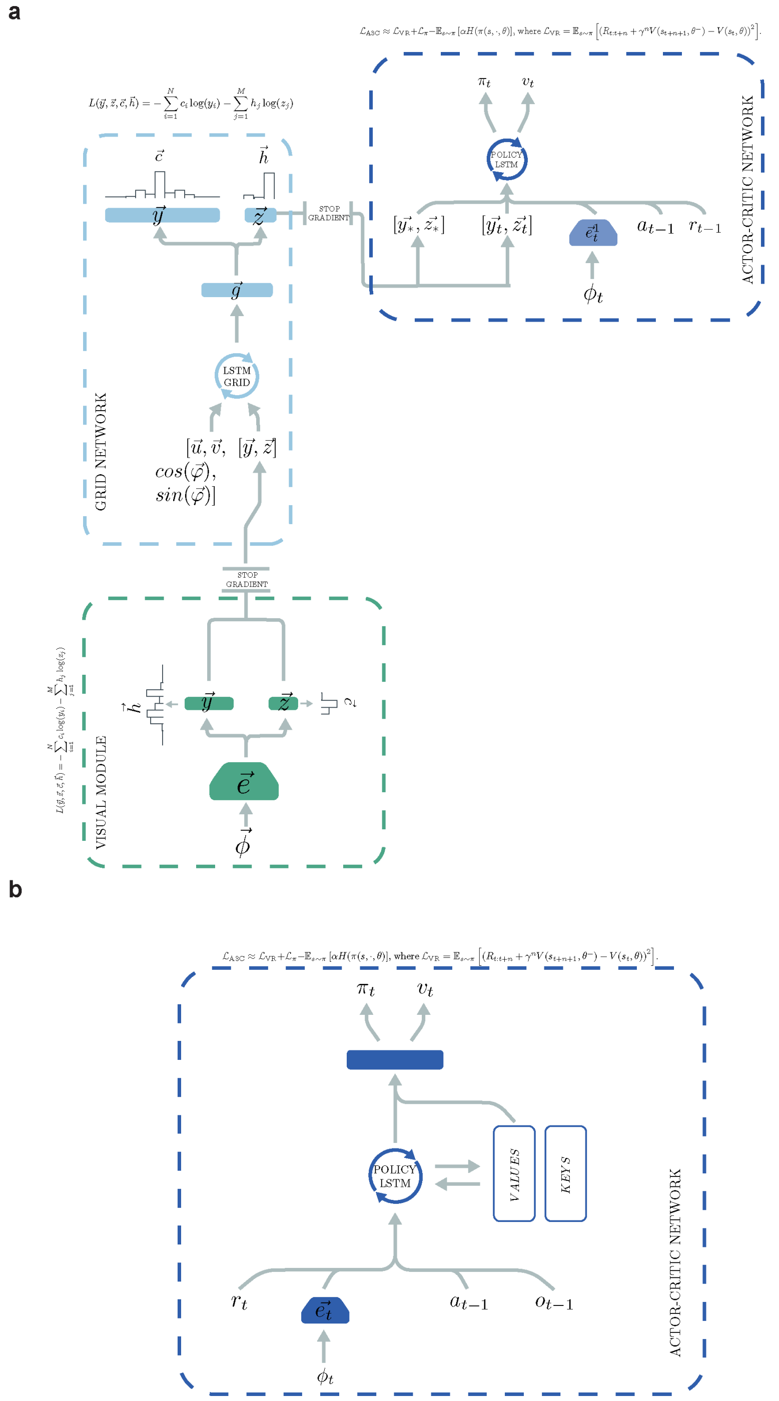
a

$\mathcal{L}_{\text{A3C}} \approx \mathcal{L}_{\text{VR}} + \mathcal{L}_{\pi} - \mathbb{E}_{s\sim\pi}\left[\alpha H(\pi(s,\cdot,\theta)\right]$, where $\mathcal{L}_{\text{VR}} = \mathbb{E}_{s\sim\pi}\left[\left(R_{t:t+n} + \gamma^n V(s_{t+n+1},\theta^-) - V(s_t,\theta)\right)^2\right]$.

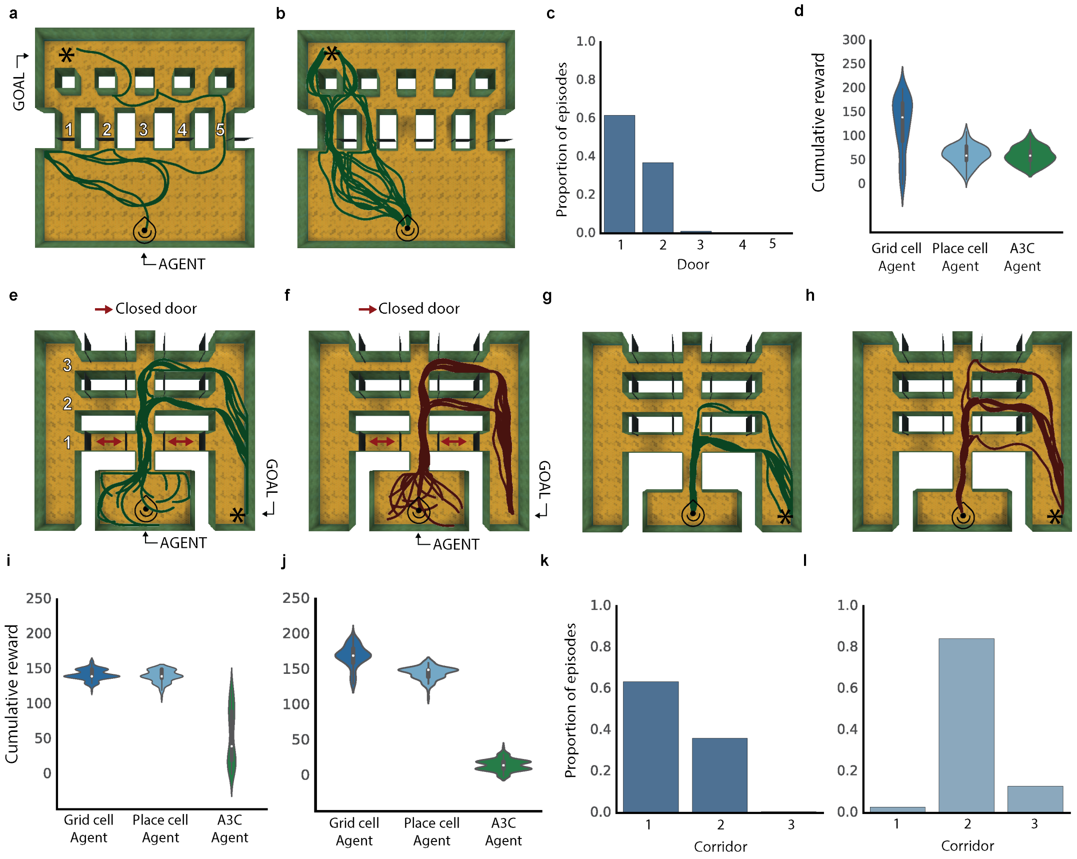
b

$\mathcal{L}_{\text{A3C}} \approx \mathcal{L}_{\text{VR}} + \mathcal{L}_{\pi} - \mathbb{E}_{s\sim\pi}\left[\alpha H(\pi(s,\cdot,\theta)\right]$, where $\mathcal{L}_{\text{VR}} = \mathbb{E}_{s\sim\pi}\left[\left(R_{t:t+n} + \gamma^n V(s_{t+n+1},\theta^-) - V(s_t,\theta)\right)^2\right]$.

Extended Data Figure 8: **Architecture of the A3C and place cell agent.** a) The A3C implementation is as described in[40]. b) The place cell agent was provided with the ground-truth place, $\vec{c}_t$, and head-direction, $\vec{h}_t$, cell activations (as described above) at each time step. The output of the fully connected layer of the convolutional network $\vec{e}_t$ was concatenated with the reward r_t, the previous action $a_t - 1$, the ground-truth current place code, $\vec{c}_t$, and current head-direction code, $\vec{h}_t$ — together with the ground truth goal place code, $\vec{c}_*$, and ground truth head direction code, $\vec{h}_*$, observed the last time the agent reached the goal (see Methods).


a
$\mathcal{L}_{\mathrm{A3C}} \approx \mathcal{L}_{\mathrm{VR}} + \mathcal{L}_{\pi} - \mathbb{E}_{s\sim\pi}\left[\alpha H(\pi(s,\cdot,\theta))\right]$, where $\mathcal{L}_{\mathrm{VR}} = \mathbb{E}_{s\sim\pi}\left[\left(R_{t:t+n} + \gamma^{n} V(s_{t+n+1},\theta^{-}) - V(s_t,\theta)\right)^{2}\right]$.
π_t
v_t
POLICY LSTM
$[\vec{y}_*, \vec{z}_*]$
$[\vec{y}_t, \vec{z}_t]$
$\vec{e}_t$
a_{t-1}
r_{t-1}
ϕ_t
ACTOR-CRITIC NETWORK
$L(\vec{y},\vec{z},\vec{c},\vec{h}) = -\sum_{i=1}^{N} c_i \log(y_i) - \sum_{j=1}^{M} h_j \log(z_j)$
$\vec{c}$
$\vec{h}$
$\vec{y}$
$\vec{z}$
STOP GRADIENT
$\vec{g}$
LSTM GRID
$[\vec{u}, \vec{v}, cos(\vec{\varphi}), sin(\vec{\varphi})]$
$[\vec{y}, \vec{z}]$
GRID NETWORK
STOP GRADIENT
$L(\vec{y},\vec{z},\vec{c},\vec{h}) = -\sum_{i=1}^{N} c_i \log(y_i) - \sum_{j=1}^{M} h_j \log(z_j)$
$\vec{h}$
$\vec{y}$
$\vec{z}$
$\vec{c}$
$\vec{e}$
$\vec{\phi}$
VISUAL MODULE
b
$\mathcal{L}_{\mathrm{A3C}} \approx \mathcal{L}_{\mathrm{VR}} + \mathcal{L}_{\pi} - \mathbb{E}_{s\sim\pi}\left[\alpha H(\pi(s,\cdot,\theta))\right]$, where $\mathcal{L}_{\mathrm{VR}} = \mathbb{E}_{s\sim\pi}\left[\left(R_{t:t+n} + \gamma^{n} V(s_{t+n+1},\theta^{-}) - V(s_t,\theta)\right)^{2}\right]$.
π_t
v_t
POLICY LSTM
VALUES
KEYS
r_t
$\vec{e}_t$
a_{t-1}
o_{t-1}
ϕ_t
ACTOR-CRITIC NETWORK

Extended Data Figure 9: **Architecture of the place cell prediction agent and of the NavMemNet agent**. a) The architecture of the place cell prediction agent is similar to the grid cell agent — having a grid cell network with the same parameters as that of the grid cell agent. The key difference is the nature of the input provided to the policy LSTM. Instead of using grid codes from the linear layer of the grid network $\vec{g}$, we used the predicted place cell population activity vector $\vec{y}$ and the predicted head direction population activity vector $\vec{z}$ (i.e. the activations present on the output place and head direction unit layers of the grid cell network, corresponding to the current and goal position) as input for the policy LSTM. As in the grid cell agent, the output of the fully connected layer of the convolutional network, $\vec{e_t}$, the reward r_t, and the previous action $a_t - 1$, were also input to the policy LSTM. The convolutional network had the same architecture described for the grid cell agent. b) NavMemNet agent. The architecture implemented is the one described in[3], specifically FRMQN but the Asynchronous Advantage Actor-Critic (A3C) algorithm was used in place of Q-learning. The convolutional network had the same architecture described for the grid cell agent and the memory was formed of 2 banks (keys and values), each one composed of 1350 slots.


a
GOAL
1
2
3
4
5
AGENT
b
c
Proportion of episodes
1.0
0.8
0.6
0.4
0.2
0.0
1
2
3
4
5
Door
d
Cumulative reward
300
250
200
150
100
50
0
Grid cell Agent
Place cell Agent
A3C Agent
e
Closed door
3
2
1
GOAL
AGENT
f
Closed door
GOAL
AGENT
g
h
i
Cumulative reward
250
200
150
100
50
0
Grid cell Agent
Place cell Agent
A3C Agent
j
250
200
150
100
50
0
Grid cell Agent
Place cell Agent
A3C Agent
k
Proportion of episodes
1.0
0.8
0.6
0.4
0.2
0.0
1
2
3
Corridor
l
1.0
0.8
0.6
0.4
0.2
0.0
1
2
3
Corridor

Extended Data Figure 10: **Flexible use of short-cuts** a) Overhead view of the linear sunburst maze in initial configuration, with only door 5 open. Example trajectory from grid cell agent during training (green line, icon indicates start location). b) Test configuration with all doors open: grid cell agent uses the newly available shortcuts (multiple episodes shown). c) Histogram showing proportion of times the agent uses each of the doors during 100 test episodes. The agent shows a clear preference for the shortest paths. d) Performance of grid cell agent and comparison agents during test episodes. e) Example grid cell agent and f) example place cell agent trajectory during training in the double E-maze (corridor 1 doors closed). g-h) in the test phase, with all doors open, the grid cell agent exploits the available shortcut (g), while the place cell agent does not (h). i-j) Performance of agents during training (i) and test (j). k-l, The proportion of times the grid (k) and place (l) cell agents use the doors on the 1st to 3rd corridor during test. The grid cell agent shows a clear preference for available shortcuts, while the place cell agent does not.

Supplemental Information for *Vector-based Navigation using Grid-like Representations in Artificial Agents.*

Andrea Banino[1*], Caswell Barry[2*], Benigno Uria[1], Charles Blundell[1], Timothy Lillicrap[1], Piotr Mirowski[1], Alexander Pritzel[1], Martin Chadwick[1], Thomas Degris[1], Joseph Modayil[1], Greg Wayne[1], Hubert Soyer[1], Fabio Viola[1], Brian Zhang[1], Ross Goroshin[1], Neil Rabinowitz[1], Razvan Pascanu[1], Charlie Beattie[1], Stig Petersen[1], Amir Sadik[1], Stephen Gaffney[1], Helen King[1], Koray Kavukcuoglu[1], Demis Hassabis[1], Raia Hadsell[1], Dharshan Kumaran[1]

[1]DeepMind, 5 New Street Square, London EC4A 3TW, UK.

[2]Department of Cell and Developmental Biology, University College London, London, UK

[*]equal contribution.

This section contains:

1. Supplementary Results
   (a) Assessing path integration and goal-finding in a square arena
   (b) Experimental manipulations to test the Vector-Based navigation hypothesis
   (c) Comparison of grid cell agent with other agents in complex, procedurally-generated multi-room environments
   (d) Probe mazes assessing ability to take novel shortcuts
2. Supplementary Discussion
   (a) Backpropagation through time (BPTT)
   (b) Relationship to previous models of grid cells
3. Supplementary Methods
   (a) Navigation through Deep RL
   (b) Additional information about Agent Architectures
   (c) Training algorithms
   (d) Neuroscience-based analyses of units
   (e) Multivariate decoding of representation of metric quantities within LSTM
   (f) Statistical reporting

1 - Supplementary Results for *Vector-based Navigation using Grid-like Representations in Artificial Agents.*

1a - Assessing path integration and goal-finding in a square arena To better understand the advantage conveyed by a grid-like representation, we trained the agent to navigate to an unmarked goal in a simple setting inspired by the classic Morris water maze (Fig. 2b&c; 2.5m×2.5m square arena; see Methods). The agent was trained in episodes to ensure it was able to generalize to arbitrary open field enclosures, each episode consisted of $5,400$ steps — corresponding to approximately 90 s in total — after which the goal location, floor texture, and cue location were randomized. An episode started with the agent in a random location, requiring it to first explore in order to find an unmarked goal. Upon reaching the goal the agent was teleported to another random location and continued to navigate with the aim of maximising the number of times it reached the goal before the episode ended. In this setting self-localisation was more challenging. Previously, in experiment described above, information about the ground truth initial location was provided to initialise the LSTM, here the grid network learned to use visual information to determine the agent's starting location and to correct for drift resulting from noise introduced to the velocity inputs (see Methods). Despite these differences the grid network continued to self-localize accurately, outputting place cell predictions consistent with the agent's location (Fig. 2e).

After locating the goal for the first time during an episode, the agent typically returned directly to it from each new starting position, showing decreased latencies for subsequent visits (average score for 100 episodes: grid cell agent = 289 vs place cell agent = 238, effect size = 1.80, 95% CI [1.63, 1.99], Fig. 2h, Extended Data Figure 6d). Performance of the grid cell agent was substantially better than that of a control place cell agent with homogeneous place fields tuned to maximize performance (see Supplemental Methods). Further, to additionally control for differences in the number and area of spatial fields between agents, we also generated two place cell agents – incorporating 256 and 660 heterogeneously sized place fields – that were explicitly matched to the grid cell agent (see Supplemental Methods for details). Again, the performance of the grid cell agent was found to be considerably better than these additional place cell agents (Average score over 100 episodes: grid cell agent = 289 vs. best place agent with 660 heterogeneous fields = 212, effect size = 3.93, 95% CI [3.54, 4.31]; best place agent with 256 heterogeneous fields = 225, effect size = 3.52, 95% CI [3.18, 3.87]).

1b - Experimental manipulations to test the Vector-Based navigation hypothesis First, to demonstrate that the goal grid code provided sufficient information to enable the agent to navigate to an arbitrary location, we substituted it with a ”fake” goal grid code sampled randomly from a location in the environment (see Methods). The agent followed a direct path to the newly specified location, circling the absent goal (Fig. 2i) — similar to rodents in probe trials of the Morris water maze (escape platform removed). As a second test, we trained a grid cell agent without providing the goal grid vector to the policy LSTM, effectively ”lesioning” this code. Performance of the grid agent drops to that of the baseline deep RL agent (A3C - a standard deep RL architecture, trained without any grid or place cell input), confirming that the goal grid code is critical for vector based navigation (see Extended Data Fig. 6c). Thirdly, to confirm the presence of a goal-directed vector, we attempted to decode the scalar quantities composing the vector from the policy LSTM. Reasoning that the goal directed vector would be particularly important at the start of a trajectory, we focused on the initial portion of navigation after the agent had reached the goal and was teleported to a new location. We found that the policy LSTM of the grid cell agent contained representations of two key components of vector-based navigation (Euclidean distance, and allocentric goal direction), and that both were more strongly present than in the place cell agent (Euclidean distance difference in r = 0.17; 95% CI [0.11, 0.24]; Goal direction difference in r = 0.22; 95% CI [0.18, 0.26]; Figure 2j&k). Notably, a neural representation of goal distance has recently been reported in mammalian hippocampus[29] (also see [49]). To determine the behavioral relevance of these two metric codes, we examined the goal-homing accuracy in each episode over several steps immediately following the period of metric decoding. We found that variation in both Euclidean distance ($r = 0.22$, 95% CI [-0.32, -0.09]) and allocentric goal direction ($r = 0.22$, 95% CI [-0.38, -0.15]) decoding error correlated with subsequent behavioral accuracy. This suggests that stronger metric codes are indeed important for accurate goal-homing behavior.

Finally, to determine the specific contribution of the grid-like units, we made targeted lesions to the goal grid code and reexamined performance and representation of the goal directed vector. When 25% of the most grid-like units were silenced (see Methods), performance was worse than lesioning 25% at random (average score for 100 episodes: 126.1 vs. 152.5, respectively; effect size = 0.38, 95% CI [0.34, 0.42]). Further, as expected, goal-directed vector codes were more strongly degraded (Euclidean distance: random lesions decoding accuracy $r = 0.45$, top-grid lesions decoding accuracy $r = 0.38$, difference in decoding accuracy = 0.08, 95% CI [0.03, 0.13]). We also performed an additional experiment where the effect of the targeted grid lesion was compared to that of lesioning non-grid units with patchy firing (see Supplemental Methods - section 3d for the details of the procedure). Our results show that the targeted grid cell lesion had a greater effect than the patchy non-grid cell lesion (average score for 100 episodes: 126.1 vs. 151.7, respectively; effect size = 0.38, 95% CI [0.34, 0.42]). These results support a role for the grid-like units in vector-based navigation, with the relatively mild impact on performance potentially accounted for by the difference in lesioning networks as compared to animals. Specifically, the procedure for lesioning networks differs in important respects from experimental lesions in animals — which bears upon the results observed. Briefly, networks have to be trained in the presence of an incomplete goal grid code and thus have the opportunity to develop a degree of robustness to the lesioning procedure – which would otherwise likely result in a catastrophic performance drop (see Methods). This opportunity is not typically afforded to experimental animals. This, therefore, may explain the significant but relatively small performance deficit observed in lesioned networks.

1c - Comparison of grid cell agent with other agents in challenging, procedurally-generated multi-room environments Our comparison agents for the grid cell agent included an agent specifically designed to use a different representational scheme for space (i.e. place cell agent, see Extended Data Figure 8b and see Methods), and a baseline deep RL agent (A3C [40], see Extended Data Figure 8a). The place cell agent relates to theoretical models of goal-directed navigation from the neuroscience literature (e.g.[41,42]). A key difference between grid and place cell based models is that the former are proposed to enable the computation of goal-directed vectors across large-scale spaces[7,10,11]and[50], whereas place cell based models are inherently limited in terms of navigational range (i.e. to the largest place field) and do not support route planning across unexplored spaces[11]. First, we test these three agents in the "goal-driven" maze (see Methods). The grid-cell agent exhibited high levels of performance, and over the course of 100 episodes, attained an average score of 346.5 (video: https://youtu.be/BWqZwLQfwlM), beating both the place cell agent (average score 258.76; contrast effect size = 1.98, 95% CI [1.79, 2.18]) and the A3C agent (average score 137.00; contrast effect size = 14.31, 95% CI [12.91, 15.71]). The grid cell agent showed markedly superior performance compared to the other agents in the “goal-doors” maze (average score over 100 episodes: grid cell agent = 284.30 vs place cell agent = 90.53, effect size = 7.86, 95% CI [7.09, 8.63]; A3C agent = 48.69, effect size = 7.73, 95% CI [6.97, 8.48]) (video of grid cell agent: https://youtu.be/BWqZwLQfwlM). Interestingly, therefore, the enhanced performance of the grid cell agent was particularly evident when it was necessary to recompute trajectories due to changes in the door configuration, highlighting the flexibility of vector-based navigation in exploiting ad hoc short-cuts (Fig. 3f).

The grid cell agent exhibited stronger performance than a professional human player in both “goal-driven” (average score: grid cell agent = 346.50 vs. professional human player = 261, effect size = 4.00, 95% CI [3.50, 4.52]) and “goal-doors” (average score: grid cell agent = 284.30 vs. professional human player = 240.5, effect size = 2.49, 95% CI [2.18, 2.81]). The human expert received 10 episodes worth of training in each environment before undergoing 20 episodes of testing. This is considerably less training than that experienced by the network. Importantly, however, the mammalian brain has evolved to path integrate and naturally the human expert had a lifetimes worth of relevant navigational experience. Hence, although directly drawing concrete conclusions from relative performance of human and agents is necessarily difficult, providing human-level performance is useful as a broad comparison and represents a commonly used benchmark in similar papers[44].

We also tested the ability of agents trained on the standard environment (11×11) to generalise to larger environments (11×17, corresponding to 2.7×4.25 meters) (see Methods). The grid cell agent exhibited strong generalistion performance compared to the control agents (average score over 100 episodes grid cell agent = 366.5 vs place cell agent = 175.7, effect size = 4.60, 95% CI [4.16, 5.06]; A3C agent = 219.4, effect size = 3.78, 95% CI [3.41, 4.15]).

We assessed the performance of two deep RL agents with external memory[3],[43] (see Extended Data Figure 9b). Whilst these agents were trained purely using RL — that is, they did not utilize supervised learning implemented by the grid cell agent — their relatively poor performance illustrates the challenge posed by the environments used (i.e. goal-driven and goal-doors) and shows that is not readily solved by the use of external memory alone. Importantly, this also serves to highlight the substantial advantage afforded to agents that can exploit vector-based mechanisms grounded in a grid-cell based Euclidean framework of space — and the potential for future work to examine the combination of such navigational strategies with more memory-intensive approaches. We also compare the grid cell agent with a variation of the place cell agent which used the predicted place cell and head direction cell as input to the Policy LSTM instead of the ground truth information (see Extended Data Figure 9a and Supplementary Methods). This agent exhibited substantially poorer performance than the grid agent.

Further, decoding accuracy was substantially and significantly higher in the grid cell agent than both the place cell (Euclidean distance difference in r = 0.44; 95% CI [0.37, 0.51]; Goal direction difference in r = 0.52; 95% CI [0.49, 0.56]) and deep RL (Euclidean distance difference in r = 0.57; 95% CI [0.5, 0.63]; Goal direction difference in r = 0.66; 95% CI [0.62, 0.70]) control agents

(Figure 3j&k).

1d - Probe mazes assessing ability to take novel shortcuts A core feature of mammalian spatial behaviour is the ability to exploit novel shortcuts and traverse unvisited portions of space[9], a capacity thought to depend on vector-based navigation[9,11]. To assess this, we examined the ability of the grid cell agent and comparison agents to use novel shortcuts when they became available in specifically configured probe mazes (see Methods for details). First, agents trained in the goal-doors environment were exposed to a linearized version of Tolman's sunburst maze. The grid cell agent, but not comparison agents, was reliably able to exploit shortcuts, preferentially passing through the doorways that offered a direct route towards the goal (Fig. 4a-c, and Extended Data Figure 10). The average testing score of the grid cell agent was higher than that of the place agent (124.1 vs 60.9, effect size = 1.46, 95% CI [1.32, 1.61]) and of the A3C agent (124.1 vs. 59.7, effect size = 1.51, 95% CI [1.36, 1.66]).

Next, to test the agents' abilities to traverse a previously unvisited section of an environment, we employed the "double-E shortcut" maze (Fig. 4d-f, and Extended Data Figure 10e-l). During training, the corridor presenting the shortest route to the goal was closed at both ends, preventing access or observation of the interior. In this simple configuration the grid and place cell agents performed similarly, exceeding the RL control agent (Extended Data Figure 10i). However, at test, when the doors were opened, the grid cell agent was able to exploit the short-cut corridor, whereas the control agents continued to follow the longer route they had previously learnt (Extended Data Figure 10j-l). In the "double-E shortcut" maze performance does not significantly differ between the grid and place cell agents, but both are significantly better than the A3C control (grid cell agent vs. place cell agent, effect size = 0.27, 95% CI [0.24, 0.29]; grid cell agent vs. A3C agent, effect size = 2.99, 95% CI [2.69, 3.29]; place cell agent vs. A3C agent, effect size = 2.92, 95% CI [2.63, 3.21]). When shortcuts become available in the test phase, the grid cell agent performs significantly better than the place agent (grid cell agent vs. place cell agent, effect size = 1.89, 95% CI [1.69, 2.09]; grid cell agent vs. A3C agent, effect size = 12.77, 95% CI [11.48, 14.07]; place cell agent vs. A3C agent, effect size = 14.87, 95% CI [13.35, 16.38]).

2 - Supplementary Discussion for *Vector-based Navigation using Grid-like Representations in Artificial Agents.*

2a - Backpropagation through time (BPTT) Whilst backpropagation provides a powerful mechanism for adjusting the weights within hierarchical networks analogous to those found in the brain (e.g. the ventral visual stream), it has long been thought to be biologically implausible for several reasons: for example, it requires access to information that is non-local to a synapse (i.e. information about errors many layers downstream). However, recent research in several directions have provided fresh new insights into how a process akin to backpropagation may be implemented in the brain [51]. Whilst less research has been conducted into how BPTT could be implemented in the brain, recent work points to potentially promising avenues that deserve further exploration [52].

2b - Relationship to previous models of grid cells Our work contrasts with previous approaches where grid cells have been hard-wired[53–56]and[57], derived through eigendecomposition of place fields[58,59], or arisen through self organization in the absence of an objective function[60]. It is worth noting that our experiments were not designed to provide insights into the development of grid cells in the brain — due to the limitations of the training algorithm used (i.e. backpropagation) in terms of biological plausibiliy (although see [61]). More generally, however, our findings accord with the perspective that the internal representations of individual brain regions such as the entorhinal cortex arise as a consequence of optimizing for specific ethologically important objective functions (e.g. path integration) — providing a parallel to the optimization process in neural networks[62].

3 - Supplementary Methods for *Vector-based Navigation using Grid-like Representations in Artificial Agents.*

3a - Navigation through Deep RL

Probe mazes to test for shortcut behavior The first maze used to test shortcut behaviour was a linearized version of Tolman's sunburst maze[63] (Fig. 4a). The maze was used to determined if the agent was able to follow an accurate heading towards the goal when a path became available. The maze was size 13×13 and contained 5 evenly spaced corridors, each of which had a door at the end closest to the start position of the agent. The agent always started on one side of the corridors with the same heading orientation (North; see Fig 4a) and the goal was always placed in the same location on the other side of the corridors. Until the agent reached the goal the first time only one door was open (door 5, Fig. 4a), but after that all the doors were opened for the remainder of the episode. After reaching the goal, the agent was teleported to the original position with the same heading orientation. This maze was used to test the shortcut capabilities of agents that had been previously trained in the ”goal doors” environment. All the agents were tested in the maze for 100 episodes, each one lasting for a fixed duration of $5,400$ environment steps (90 seconds).

The second maze, termed double E-maze, was designed to test the agents abilities to traverse a previously unvisited section of an environment. The maze was size 12×13 and was formed of 2 symmetric sides each one with 3 corridors. The goal location was always on the bottom right or left, and the location was randomized over episodes. During training, the left and right corridors closest to the bottom (i.e. those providing the shortest paths to the goals) were always closed from both sides to avoid any exploration down these corridors (see Extended Data Figure 10e&f). This ensured any subsequent shortcut behavior had to traverse unexplored space. Of the remaining corridors, at any time, on each side only one was accessible (top or middle, randomly determined). Each time the agent reached the goal, the doors were randomly configured again (with the same constraints). The agent always started in a random location in the central room with a random orientation. At test time, after the agent reached the goal for the first time, all corridors were opened, allowing potential shortcut behavior (see Extended Data Figure 10g&h). During the test phase, the agent always started in the center of the central room facing north. Each agent was trained for $1e9$ environment step divided into episodes of $5,400$ steps (90 seconds), and subsequently tested for 100 episodes, each one lasting for a fixed duration of $5,400$ environment steps (90 seconds).

3b - Additional information about Agent Architectures

Details of vision module in the grid cell agent The convolutional neural network had four convolutional layers. The first convolutional layer had 16 filters of size 5×5 with stride 2 and padding 2. The second convolutional layer had 32 filters of size 5×5 with stride 2 and padding 2. The third convolutional layer had 64 filters of size 5×5 with stride 2 and padding 2. Finally, the fourth convolutional layer with 128 filters of size 5×5 with stride 2 and padding 2. All convolutional hidden layers were followed by a rectifier nonlinearity. The last convolution was followed by a fully connected layer with 256 hidden units. The same convolutional neural network was used for the actor-critic learner. The weights of the two network were not shared.

Further details about the place cell agent Place cell agent with homogeneously sized place fields: we tested agents with fields — modelled as regular 2D Gaussians — having standard deviations of 7.5cm, 25cm, and 75cm bins. The agent with fields of size 7.5cm was found to perform best (highest cumulative reward on the Morris water maze task; see Supplemental Results) and hence was chosen as the primary place cell control agent (see main text for score comparisons).

Place cell agent with heterogeneously sized place fields: to control for differences in the number and area of spatial fields between agents, we also generated two further place cell agents that were explicitly matched to the grid cell agent. Specifically, we used a watershedding algorithm[64] to detect 660 individual grid fields in the grid-like units of the grid cell agent. The distribution of the areas of these fields were found to exhibit 3 peaks — based on a Gaussian fitting procedure — having means equivalent to 2D Gaussians with standard deviations of 8.2cm, 15.0cm, and 21.7cm. Hence we generated a further control agent having 395 place cells of size 8.2cm, 198 of size 15.0cm, and 67 of 21.7cm — 660 place cells in total, the relative numbers reflecting the magnitudes of the Gaussians fit to the distribution. A final control agent was also generated having 256 place cell units in total — the same number of linear layer units as the grid agent — distributed across the same three scales in a similar ratio. Additionally, we note that from a machine learning perspective, the place cell and grid cell agents with the same number of linear layer units are in principle well matched since they are provided with the same input information and have an identical number of parameters.

Place cell prediction agent. The architecture of the place cell prediction agent (Extended Data Figure 9a) is similar to the grid cell agent described in the Methods : the key difference is the nature of the input provided to the policy LSTM as described below. Specifically, the output of the fully connected layer of the convolutional network, $\vec{e}_t$, was concatenated with the reward r_t, the previous action $a_t - 1$, the current predicted place cell activity vector, $\vec{y}_t$, and the current predicted head direction cell activity vector $\vec{h}_t$ — and the goal predicted place cell activity vector , $\vec{y}_*$, and goal head direction activity vector, $\vec{h}_*$, observed the last time the agent had reached the goal — or zeros if the agent had not yet reached the goal within the episode. The convolutional network had the same architecture described for the grid cell agent.

3c - Training algorithms

We assume the standard reinforcement learning setting where an agent interacts with an environment over a number of discrete time steps. As previously defined the at time t the agent receives an observation o_t along with a reward r_t and produces an action a_t. The agent's state s_t is a function of its experience up until time t, $s_t = f(o_1, r_1, a_1, ..., o_t, r_t)$ (The specifics of o_t are defined in the architecture section). The n-step return $R_{t:t+n}$ at time t is defined as the discounted sum of rewards, $\hat{R}_t = \sum_{i=0...n-1} \gamma^i r_{t+i} + \gamma^n V(s_{t+n}, \theta)$. The value function is the expected return from state s, $V^\pi(s) = \mathbb{E}[R_{t:\infty} | s_t = s, \pi]$, under actions selected accorded to a policy $\pi(a|s)$. See main methods for the details of the loss functions.

3d - Neuroscience-based analyses of units

Gridness score and grid scale calculation Following [20] and [18] spatial autocorrelograms of ratemaps were used to assess the gridness and grid scale of linear layer units. First, for each unit, the spatial autocorrelogram was calculated as defined in [20]. To calculate gridness[20], a measure of hexagonal periodicity, we followed the 'expanding gridness' method introduced by [18]. Briefly, a circular annulus centred on the origin of the autocorrelogram was defined, having radius of 8 bins and with the central peak excluded. The annulus was rotated in 30° increments and, at each increment, the Pearson product moment correlation coefficient with the unrotated version of itself found. An interim gridness value was then defined as the highest correlation obtained from rotations of 30, 90 and 150° subtracted from the lowest at 0, 60 and 120°. This process was then repeated, each time expanding the annuls by 2, up to a maximum of 20. Finally, the gridness value was taken as the highest interim score.

Grid scale[20], a simple measure of the wavelength of spatial periodicity, was defined from the autocorrelogram as follows. The six local maxima closest to but excluding the central peak were identified. Grid scale was then calculated as the median distance of these peaks from the origin.

Directional measures Following[46] the degree of directional modulation exhibited by each unit was assessed using the length of the resultant vector of the directional activity map. Vectors corresponding to each bin of the activity map were created:

$$r_i = \begin{bmatrix} \beta_i \cos \alpha_i \\ \beta_i \sin \alpha_i \end{bmatrix}, \tag{6}$$

where α and β are, respectively, the centre and intensity of angular bin i in the activity map. These vectors were averaged to generate a mean resultant vector:

$$\vec{r} = \frac{\sum_{n=1}^{N} r_i}{\sum_{n=1}^{N} \beta_i}, \tag{7}$$

and the length of the resultant vector calculated as the magnitude of $\vec{r}$. We used 20 angular bins.

Border score To identify units that were preferentially active adjacent to the edges of the enclosure we adopted a modified version of the border score[47]. For each of the four walls in the square enclosure, the average activation for that wall, b_i, was compared to the average centre activity c obtaining a border score for that wall, and the maximum was used as the border-score for the unit:

$$b_s = \max_{i \in \{1,2,3,4\}} \frac{b_i - c}{b_i + c} \tag{8}$$

where b_i is the mean activation for bins within d_b distance from the i-th wall and c the average activity for bins further than d_b bins from any wall. In all our experiments 20 by 20 bins where used and d_b took value 3.

Threshold setting for gridness, border score, and directional measures The hexagonality of the spatial activity map (gridness), directional modulation (length of resultant vector), and propensity to be active against environmental boundaries (border scale) exhibited by units in the linear layer were benchmarked against null distributions obtained using permutation procedures[65][48].

For the gridness measure and border score, null distributions were constructed using a ’field shuffle’ procedure equivalent to that specified by[48]. Briefly, a watershedding algorithm[64] was applied to the ratemap to segment spatial fields. The peak bin of each field was found and allocated to a random position within the ratemap. Bins around each peak were then incrementally replaced, retaining as far as possible their proximity to the peak bin. This procedure was repeated 100 times for each of the units present in the linear layer and the gridness and border score of the shuffled ratemaps assessed as before. In each case the 95th percentile of the resulting null distribution was found and used as a threshold to determine if that unit exhibited significant grid or border-like activity.

To validate the thresholds obtained using shuffling procedures we calculated alternative null distributions by analysing the grid and border responses of linear units from 500 untrained networks. Again, in each case, a grid score and border score for each unit was calculated, these were pooled, and the 95th percentile found. In all cases the thresholds obtained by the first method were found to be most stringent and these were used for all subsequent analyses

To establish a significance threshold for directional modulation we calculated the length of the resultant vector that would demonstrate statistically significance under a Rayleigh test of directional uniformity at $\alpha = 0.01$. The resultant vector was calculated by first calculating the average activation for 20 directional bins. A threshold length of 0.47 for the resultant vector was obtained. The most stringent of these two thresholds was used.

Clustering of scale in grid-like units To determine if grid-like units exhibited a tendency to cluster around specific scales we applied two methods.

First, following [22], to determine if the scales of grid-like units (gridness $>$ 0.37, 129/512 units) followed a continuous or discrete distribution we calculated the 'discreteness measure'[22] of the distribution of their scales. Specifically, scales were binned into a histogram with 13 bins distributed evenly across a range corresponding to scales 10 to 36 spatial bins. 'Discreteness' was defined as the standard deviation of the counts in each bin. Again following[22], statistical significance for this value was obtained by comparing it to a null distribution generated from a shuffled version of the same data. Specifically, shuffles were generated as follows: For each unit, a random number was drawn from a flat distribution between -1/2 and +1/2 of the smallest grid scale in this case between -7 and +7 spatial bins. The random number was added to the grid scales, the population was binned as before, and the discreteness score calculated. This procedure was completed 500 times. The discreteness score of the real data was found to exceed that of all the 500 shuffles ($p < 0.002$).

Second, to characterise the number and location of scale clusters, the distribution of scales from grid-like units was fit with Gaussian mixture distributions containing 1 to 8 components. Fits were made using an Expectation-Maximization approach implemented with fitgmdist (Matlab 2016b, Mathworks, MA). The efficiency of fits made with different numbers of components was compared using Bayesian Information Criterion (BIC)[66] the model (3 components) with the lowest BIC score was selected as the most efficient.

Lesioning experiment: comparison of targeted grid unit lesion vs lesion of patchy non-grid units We lesioned a random subset of patchy multi-field spatial cells that were non-grid units (i.e. grid score lower than 0.37 threshold). The units chosen had a head-direction score lower than 0.47 and the number of spatial fields was in the same range as grid-like units (3 to 13). The number of fields in each ratemap was calculated by applying a watershedding algorithm[64] to their ratemap – ignoring fields with area smaller than 4 bins. This procedure identified 174 units as multi-field patchy spatial cells (out of 256 units in the linear layer). We then selected 64 random units from these 174 and we ran 100 episodes in which these units were silenced (see Supplemental Results section 1b). We also ran another variant of the experiment where we ran 100 episodes and in each episode we selected a different subset of 64 random units from the 174 identified by the watershedding procedure, and these units were silenced. The results were not qualitatively different from the former experiment (data not shown).

3e - Multivariate decoding of representation of metric quantities within LSTM

A key prediction of the vector-based navigation hypothesis is that grid codes should allow downstream regions to compute a set of metric codes relevant to accurate goal-directed navigation. Specifically, Euclidean distance and allocentric direction to the goal should both be computed by an agent using vector-based navigation (see Fig. 2j&k also 3i-k). To test whether the same representations can be found in the grid cell agent, and thereby provide additional evidence that it is indeed using a vector-based navigation strategy, we recorded the activity in the policy LSTM of the grid cell agent while it navigated in the land-maze and goal-driven environments. For each environment and agent, we collected data from 200 separate episodes. In each episode, we recorded data from the time period following the first time the agent reached the goal and was teleported to a new location in the maze. After an initial period to allow self-localization (8 steps), we examined the representation of the metric quantities over the next 12 steps, where the LSTM activity was sampled at 4 even points over those steps. We focussed on this time period because the agent potentially has knowledge of the goal location, but has not yet been able to learn the optimal path to the goal. Thus it is this initial period of time where the computation of the vector-based navigation metrics should be most useful, as this allows accurate navigation right from the start of being teleported to a new location. In the land maze task, we additionally collected the same data from a place cell agent control, and the two lesioned grid cell agents. In the goal driven task, we collected data from the place cell agent and A3C. For each agent, we applied a decoding analysis to the

LSTM dictating the policy and value function. We ran two separate decoding analyses, looking for evidence of each of the two metric codes (i.e. Euclidean distance, allocentric goal direction). For each decoding analysis we trained a L2-regularized (ridge) regression model on all data apart from the first 21 time-steps of each episode. The model was then tested on the four early sampling steps of interest, where accuracy was assessed as the Pearson correlation between the predicted and actual values over the 200 episodes. The penalization parameter was selected by randomly splitting the training data into internal training and validation sets (90% and 10% of the episodes respectively). The optimal parameter was selected from 30 values, evenly spaced on a log scale between 0.001 and 1000, based on the best performance on the validation set. This parameter was then used to train the model on the full training set, and evaluated on the fully independent test set. As the allocentric direction metric is circular, we decomposed the vector into two target variables: the cosine and sine of the polar angle. All reported allocentric decoding results are the average of the cosine and sine results. For the purpose of comparing decoding accuracy across agents, we report the difference in accuracy, along with a 95% bootstrapped confidence interval on this difference, based on 10,000 samples.

3f - Statistical reporting

We followed the guidelines outlined by[67]. Specifically reporting effect sizes and confidence intervals. Unless otherwise stated, the effect sizes are calculated using the following formula:

$$effect\ size = \frac{\mu_{group1} - \mu_{group2}}{\sigma_{pooled}}, \tag{9}$$

and the σ_{pooled} was calculated accordingly to[68] using:

$$\sigma_{pooled} = \sqrt{\frac{(N_{group1} - 1) \times \sigma^2_{group1} + (N_{group2} - 1) \times \sigma^2_{group2}}{N_{group1} + N_{group2} - 2}} \tag{10}$$

The confidence interval for the effect size was calculated accordingly to[69] using:

$$ci_{effectsize} = \sqrt{\frac{N_{group1} + N_{group2}}{N_{group1} \times N_{group2}} + \; + \frac{effect\ size^2}{2 \times (N_{group1} + N_{group2})}}. \tag{11}$$

Parameter name	Value	Description
T	15	Duration of simulated trajectories (seconds)
L	2.2	Width and height of environment, or diameter for circular environment (meters)
d	0.03	Perimeter region distance to walls (meters)
$\sigma^{(v)}$	0.13	Forward velocity Rayleigh distribution scale (m/sec)
$\mu^{(\phi)}$	0	Rotation velocity Gaussian distribution mean (deg/sec)
$\sigma^{(\phi)}$	330	Rotation velocity Gaussian distribution standard deviation (deg/sec)
ρ_{R_H}	0.25	Velocity reduction factor when located in the perimeter
Δ_{R_H}	90	Change in angle when located in the perimeter (deg)
Δt	0.02	Simulation-step time increment (seconds)
N	256	Number of place cells
$\sigma^{(c)}$	0.01	Place cell standard deviation parameter (meters)
M	12	Number of target head direction cells
$\kappa^{(h)}$	20	Head direction concentration parameter
g_c	10^{-5}	Gradient clipping threshold
minibatch size	10	Number of trajectories used in the calculation of a stochastic gradient
trajectory length	100	Number of time steps in the trajectories used for the supervised learning task
learning rate	10^{-5}	Step size multiplier in the RMSProp algorithm
momentum	0.9	Momentum parameter of the RMSProp algorithm
L2 regularisation	10^{-5}	Regularisation parameter for linear layer
parameter updates	300000	Total number of gradient descent steps taken

Table 1: Supervised learning hyperparameters.

Parameter name	Value	Description
Learning rate	$[0.000001, 0.0002]$	Step size multiplier in the shared RMSProp algorithm of the actor-critic learner with a break
Gradient momentum	0.99	Momentum parameter of the shared RMSProp algorithm
Baseline cost [α]	$[0.48, 0.52]$	Cost applied on the gradient of v
Entropy regularisation [β]	$[0.00006, 0.0001]$	Entropy regularization term with respect to the policy parameters
Discount	0.99	Discount factor gamma used in the value function estimation
Back-propagation step in the actor-critic learner	100	Number of backpropagation step used to unroll the LSTM
Action repeat	4	Repeat each action selected bu the agent this many times
Learning rate grid network	0.001	Step size multiplier in the RMSProp algorithm of the supervised learner
$\sigma^{(c)}$	40	Place cell scale
M	12	Number of target head direction cells
$\kappa^{(h)}$	20	Head direction concentration parameter
Back-propagation step in the supervised learner	100	Number of time steps in the trajectories used for the supervised learning task
L2 regularization	0.0001	Regularization parameter for linear layers in bottleneck
Gradient momentum	0.9	Momentum parameter of the RMSProp algorithm in the supervised learner

Table 2: Hyperparameters of all the agents presented. Values in square bracket are sampled from a categorial distribution in that range

49. Chadwick, M. J., Jolly, A. E., Amos, D. P., Hassabis, D. & Spiers, H. J. A goal direction signal in the human entorhinal/subicular region. *Current Biology* **25**, 87–92 (2015).

50. Kubie, J. L. & Fenton, A. A. Linear look-ahead in conjunctive cells: an entorhinal mechanism for vector-based navigation. *Frontiers in neural circuits* **6**, 20 (2012).

51. Scellier, B. & Bengio, Y. Towards a biologically plausible backprop. *arXiv preprint arXiv:1602.05179* **914** (2016).

52. Ke, N. R. *et al.* Sparse attentive backtracking: Long-range credit assignment in recurrent networks. *arXiv preprint arXiv:1711.02326* (2017).

53. Burgess, N., Barry, C. & O'keefe, J. An oscillatory interference model of grid cell firing. *Hippocampus* **17**, 801–812 (2007).

54. Hasselmo, M. E., Giocomo, L. M. & Zilli, E. A. Grid cell firing may arise from interference of theta frequency membrane potential oscillations in single neurons. *Hippocampus* **17**, 1252–1271 (2007).

55. Burak, Y. & Fiete, I. R. Accurate path integration in continuous attractor network models of grid cells. *PLoS Comput Biol* **5**, e1000291 (2009).

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

57. Gustafson, N. J. & Daw, N. D. Grid cells, place cells, and geodesic generalization for spatial reinforcement learning. *PLoS Comput Biol* **7**, e1002235 (2011).

58. Stachenfeld, K. L., Botvinick, M. & Gershman, S. J. Design principles of the hippocampal cognitive map. In *Advances in neural information processing systems*, 2528–2536 (2014).

59. Dordek, Y., Soudry, D., Meir, R. & Derdikman, D. Extracting grid cell characteristics from place cell inputs using non-negative principal component analysis. *eLife* **5**, e10094 (2016).

60. Widloski, J. & Fiete, I. How does the brain solve the computational problems of spatial navigation? In *Space, Time and Memory in the Hippocampal Formation*, 373–407 (Springer, 2014).

61. Bengio, Y., Lee, D.-H., Bornschein, J., Mesnard, T. & Lin, Z. Towards biologically plausible deep learning. *arXiv preprint arXiv:1502.04156* (2015).

62. Marblestone, A. H., Wayne, G. & Kording, K. P. Toward an integration of deep learning and neuroscience. *Frontiers in Computational Neuroscience* **10** (2016).

63. Tolman, E. C. *et al.* Cognitive maps in rats and men (1948).

64. Beucher, S. Use of watersheds in contour detection. In *Proceedings of the International Workshop on Image Processing* (CCETT, 1979).

65. Yartsev, M. M., Witter, M. P. & Ulanovsky, N. Grid cells without theta oscillations in the entorhinal cortex of bats. *Nature* **479**, 103–107 (2011).

66. Schwarz, G. Estimating the dimension of a model. *The Annals of Statistics* **6**, 461–464 (1978).

67. Halsey, L. G., Curran-Everett, D., Vowler, S. L. & Drummond, G. B. The fickle p value generates irreproducible results. *Nature methods* **12**, 179 (2015).

68. Olejnik, S. & Algina, J. Measures of effect size for comparative studies: Applications, interpretations, and limitations. *Contemporary educational psychology* **25**, 241–286 (2000).

69. Hedges, L. & Olkin, I. *Statistical Methods for Meta-analysis* (Academic Press, 1985). URL `https://books.google.co.uk/books?id=brNpAAAAMAAJ`.