Supplementary Materials for "On Path Integration of Grid Cells: Group Representation and Isotropic Scaling"

Ruiqi Gao[1]* ruiqigao@ucla.edu
Jianwen Xie[2] jianwen@ucla.edu
Xue-Xin Wei[3] weixx@utexas.edu
Song-Chun Zhu[1,4,5] sczhu@stat.ucla.edu
Ying Nian Wu[1] ywu@stat.ucla.edu

[1]Department of Statistics, UCLA [2]Cognitive Computing Lab, Baidu Research
[3]Department of Neuroscience, UT Austin [4]Department of Computer Science, UCLA
[5]Beijing Institute for General Artificial Intelligence (BIGAI)

Contents

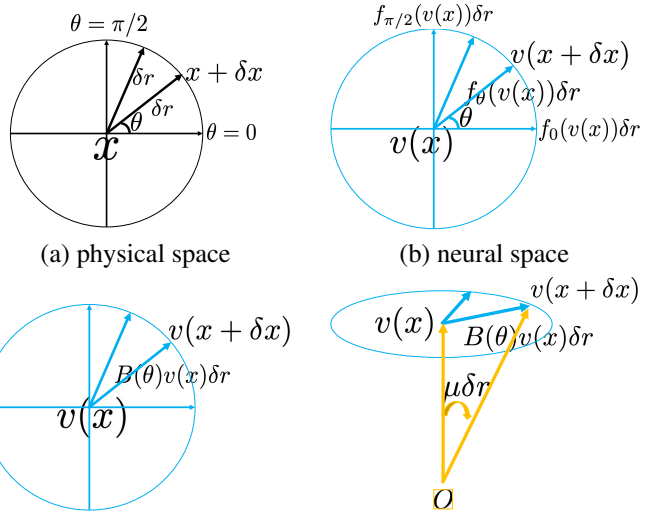
*The author is now a Research Scientist at Google Brain team.

35th Conference on Neural Information Processing Systems (NeurIPS 2021).

1 Theoretical analysis

1.1 Graphical illustrations of key equations

Fig. 1 illustrates key equations in the main text as well as in the supplementary materials.



(a) physical space (b) neural space

(c) linear transformation (d) as rotation

Figure 1: Color-coded illustration. (a) In the 2D physical space, the agent moves from $\boldsymbol{x}$ to $\boldsymbol{x}+\delta\boldsymbol{x}$, where $\delta\boldsymbol{x} = (\delta r\cos\theta, \delta r\sin\theta)$, i.e., the agent moves by δr along the direction θ. We also show a displacement of δr in a different direction. (b) In the d-dimensional neural space, the vector $\boldsymbol{v}(\boldsymbol{x})$ is changed to $\boldsymbol{v}(\boldsymbol{x}+\delta\boldsymbol{x}) = F(\boldsymbol{v}(\boldsymbol{x}), \delta r, \theta) = \boldsymbol{v}(\boldsymbol{x}) + f_\theta(\boldsymbol{v}(\boldsymbol{x}))\delta r + o(\delta r)$, where the displacement is $f_\theta(\boldsymbol{v}(\boldsymbol{x}))\delta r = f_0(\boldsymbol{v}(\boldsymbol{x}))\delta r\cos\theta + f_{\pi/2}(\boldsymbol{v}(\boldsymbol{x}))\delta r\sin\theta$. Under the isotropic condition that $\|f_\theta(\boldsymbol{v}(\boldsymbol{x}))\|$ is constant over θ, the local 2D self-motion $\delta\boldsymbol{x}$ at $\boldsymbol{x}$ in the 2D physical space is embedded conformally into the neural space as a 2D subspace around $\boldsymbol{v}(\boldsymbol{x})$. (c) Linear transformation, where $f_\theta(\boldsymbol{v}(\boldsymbol{x})) = \boldsymbol{B}(\theta)\boldsymbol{v}(\boldsymbol{x})$. (d) 3D perspective view of linear transformation as a rotation: $\boldsymbol{v}(\boldsymbol{x}+\delta\boldsymbol{x})$ is a rotation of $\boldsymbol{v}(\boldsymbol{x})$, and the angle of rotation is $\mu\delta r$, where $\mu = \|\boldsymbol{B}(\theta)\boldsymbol{v}(\boldsymbol{x})\|/\|\boldsymbol{v}(\boldsymbol{x}))\|$ (μ may depend on $\boldsymbol{x}$).

1.2 Proof of Theorem 1 on conformal embedding

Proof: See Fig. 1(a) and (b) for an illustration. Consider the self-motion $\delta\boldsymbol{x} = (\delta r\cos\theta, \delta r\sin\theta)$,

$$\boldsymbol{v}(\boldsymbol{x}+\delta\boldsymbol{x}) = F(\boldsymbol{v}(\boldsymbol{x}), \delta r, \theta) = \boldsymbol{v}(\boldsymbol{x}) + f_\theta(\boldsymbol{v}(\boldsymbol{x}))\delta r + o(\delta r). \tag{1}$$

We can decompose the self-motion $\delta\boldsymbol{x}$ into two steps. First move along the direction 0 by $\delta r\cos\theta$, and then move along the direction $\pi/2$ by $\delta r\sin\theta$. Then under the **group representation condition**:

$$\begin{aligned}
\boldsymbol{v}(\boldsymbol{x}+\delta\boldsymbol{x}) &= F[F(\boldsymbol{v}(\boldsymbol{x}), \delta r\cos\theta, 0), \delta r\sin\theta, \pi/2)] \\
&= F[\boldsymbol{v}(\boldsymbol{x}) + f_0(\boldsymbol{v}(\boldsymbol{x}))\delta r\cos\theta + o(\delta r), \delta r\sin\theta, \pi/2] \\
&= [\boldsymbol{v}(\boldsymbol{x}) + f_0(\boldsymbol{v}(\boldsymbol{x}))\delta r\cos\theta] + f_{\pi/2}[\boldsymbol{v}(\boldsymbol{x}) + f_0(\boldsymbol{v}(\boldsymbol{x}))\delta r\cos\theta + o(\delta r)]\delta r\sin\theta + o(\delta r) \\
&= \boldsymbol{v}(\boldsymbol{x}) + f_0(\boldsymbol{v}(\boldsymbol{x}))\delta r\cos\theta + f_{\pi/2}(\boldsymbol{v}(\boldsymbol{x}))\delta r\sin\theta + o(\delta r),
\end{aligned} \tag{2}$$

The last equation holds because assuming the derivative $f'_{\pi/2}(\boldsymbol{v}(\boldsymbol{x}))$ exists, then by first-order Taylor expansion,

$$f_{\pi/2}[\boldsymbol{v}(\boldsymbol{x}) + f_0(\boldsymbol{v}(\boldsymbol{x}))\delta r\cos\theta + o(\delta r)]\delta r\sin\theta \tag{3}$$

$$= [f_{\pi/2}(\boldsymbol{v}(\boldsymbol{x})) + f'_{\pi/2}(\boldsymbol{v}(\boldsymbol{x}))f_0(\boldsymbol{v}(\boldsymbol{x}))\delta r\cos\theta + o(\delta r)]\delta r\sin\theta \tag{4}$$

$$= f_{\pi/2}(\boldsymbol{v}(\boldsymbol{x}))\delta r\sin\theta + o(\delta r). \tag{5}$$

Since $\boldsymbol{v}(\boldsymbol{x}+\delta\boldsymbol{x}) = \boldsymbol{v}(\boldsymbol{x}) + f_\theta(\boldsymbol{v}(\boldsymbol{x}))\delta r + o(\delta r)$, by Eq. (2) we have $f_\theta(\boldsymbol{v}(\boldsymbol{x})) = f_0(\boldsymbol{v}(\boldsymbol{x}))\cos\theta + f_{\pi/2}(\boldsymbol{v}(\boldsymbol{x}))\sin\theta$, which is a 2D basis expansion. We are yet to prove that the two basis vectors $f_0(\boldsymbol{v}(\boldsymbol{x}))$ and $f_{\pi/2}(\boldsymbol{v}(\boldsymbol{x}))$ are orthogonal with equal norm.

For notational simplicity, let $\boldsymbol{v}_1 = f_0(\boldsymbol{v}(\boldsymbol{x}))$ and $\boldsymbol{v}_2 = f_{\pi/2}(\boldsymbol{v}(\boldsymbol{x}))$. Then under the **isotropic scaling condition**, $\|\boldsymbol{v}_1\| = \|\boldsymbol{v}_2\| = \|f_\theta(\boldsymbol{v}(\boldsymbol{x}))\| = s$, and $f_\theta(\boldsymbol{v}(\boldsymbol{x})) = \boldsymbol{v}_1 \cos\theta + \boldsymbol{v}_2 \sin\theta$ for any θ. Then we have that for any θ,

$$s^2 = \|f_\theta(\boldsymbol{v}(\boldsymbol{x}))\|^2 = \|\boldsymbol{v}_1 \cos\theta + \boldsymbol{v}_2 \sin\theta\|^2 = s^2 + 2\langle \boldsymbol{v}_1, \boldsymbol{v}_2\rangle \cos\theta \sin\theta. \tag{6}$$

Thus $\langle \boldsymbol{v}_1, \boldsymbol{v}_2\rangle = 0$, i.e., $f_0(\boldsymbol{v}(\boldsymbol{x})) \perp f_{\pi/2}(\boldsymbol{v}(\boldsymbol{x}))$. This leads to the conformal embedding of the local 2D polar system in the physical space as a 2D polar system in the d-dimensional neural space, with a scaling factor s (which may depend on $\boldsymbol{x}$). □

1.3 Proofs of Theorem 2 and Proposition 1 on error correction

Proof of Theorem 2: By Theorem 1, for a fixed self-position $\boldsymbol{x}$, we embed the 2D local neighborhood around $\boldsymbol{x}$ as a local 2D plane around $\boldsymbol{v}(\boldsymbol{x})$ in the d-dimensional neural space. A local perturbation in self-position, $\delta\boldsymbol{x}$, is translated into a local perturbation in $\boldsymbol{v}(\boldsymbol{v} + \delta\boldsymbol{x})$, so that

$$\|\delta\boldsymbol{v}\|^2 = \|f_\theta(\boldsymbol{v}(\boldsymbol{x}))\delta r + o(\delta r)\|^2 = s^2\|\delta\boldsymbol{x}\|^2, \tag{7}$$

where $\delta\boldsymbol{v} = \boldsymbol{v}(\boldsymbol{x} + \delta\boldsymbol{x}) - \boldsymbol{v}(\boldsymbol{x})$.

Suppose the agent infers its 2D position $\hat{\boldsymbol{x}}$ by $\hat{\boldsymbol{x}} = \arg\min_{\boldsymbol{x}'} \|\boldsymbol{v} - \boldsymbol{v}(\boldsymbol{x}')\|^2$, which amounts to projecting $\boldsymbol{v}$ onto the local 2D plane around $\boldsymbol{v}(\boldsymbol{x})$. The projected vector $\boldsymbol{v}(\hat{\boldsymbol{x}})$ on the local 2D plane is $\boldsymbol{v}(\boldsymbol{x}) + \delta\boldsymbol{v}$, where $\delta\boldsymbol{v}$ is the projection of $\boldsymbol{\varepsilon}$ onto the 2D plane. More specifically, let $(\boldsymbol{v}_1, \boldsymbol{v}_2)$ be an orthonormal basis of the local 2D plane centered at $\boldsymbol{v}(\boldsymbol{x})$. Then $\delta\boldsymbol{v}$ can be written as $e_1\boldsymbol{v}_1 + e_2\boldsymbol{v}_2$, where

$$\boldsymbol{e} = (e_1, e_2)^\top = (\boldsymbol{v}_1, \boldsymbol{v}_2)^\top \boldsymbol{\varepsilon} \sim \mathcal{N}(0, \tau^2 \boldsymbol{I}_2). \tag{8}$$

Let $\delta\boldsymbol{x} = \hat{\boldsymbol{x}} - \boldsymbol{x}$. Due to **isotropic scaling and conformal embedding**, the ℓ_2 squared error translate according to

$$\|\delta\boldsymbol{x}\|^2 = \|\delta\boldsymbol{v}\|^2/s^2 = (e_1^2 + e_2^2)/s^2, \tag{9}$$

whose expectation is $2\tau^2/s^2$. Thus $\mathbb{E}\|\hat{\boldsymbol{x}} - \boldsymbol{x}\|^2 = 2\tau^2/s^2$.□

Proof of Proposition 1: It is reasonable to assume $\tau^2 = \alpha^2(\|\boldsymbol{v}(\boldsymbol{x})\|^2/d)$, where α^2 measures the variance of noise relative to $\|\boldsymbol{v}(\boldsymbol{x})\|^2/d$, which is the average of $(v_i(\boldsymbol{x})^2, i = 1, ..., d)$. In other words, α^2 measures the noise level.

In the linear case, the metric is

$$\mu = \|f_\theta(\boldsymbol{v}(\boldsymbol{x}))\|/\|\boldsymbol{v}(\boldsymbol{x})\| = \|\boldsymbol{B}(\theta)\boldsymbol{v}(\boldsymbol{x}))\|/\|\boldsymbol{v}(\boldsymbol{x})\| = s/\|\boldsymbol{v}(\boldsymbol{x})\|, \tag{10}$$

which measures how fast $\boldsymbol{v}(\boldsymbol{x})$ rotates in the neural space as $\boldsymbol{x}$ changes. Then

$$\mathbb{E}\|\delta\boldsymbol{x}\|^2 = 2\alpha^2/(\mu^2 d). \tag{11}$$

The above scaling shows that error correction depends on two factors. One is the metric μ, and the other is the dimensionality d, i.e., the number of neurons. These correspond to two phases of error correction. One is to project the d-dimensional $\boldsymbol{\varepsilon}$ to the 2-dimensional $\delta\boldsymbol{v}$. The bigger d is, the better the error correction. The other is to translate $\|\delta\boldsymbol{v}\|^2$ to $\|\delta\boldsymbol{x}\|^2$. The bigger μ is, the better the error correction. □

1.4 Proof of Theorem 4 on hexagon grid patterns

Proof: Let $\boldsymbol{e}(\boldsymbol{x}) = (\exp(i\langle \boldsymbol{a}_j, \boldsymbol{x}\rangle), j = 1, 2, 3)^\top$, where $(\boldsymbol{a}_j, j = 1, 2, 3)$ are three 2D vectors of equal norm, and the angle between every pair of them is $2\pi/3$. Let $\boldsymbol{v}(\boldsymbol{x}) = \boldsymbol{U}\boldsymbol{e}(\boldsymbol{x})$, where $\boldsymbol{U}$ is an arbitrary unitary matrix, i.e., $\boldsymbol{U}^*\boldsymbol{U} = \boldsymbol{I}$. Then $\|\boldsymbol{v}(\boldsymbol{x})\|^2 = \|\boldsymbol{e}(\boldsymbol{x})\|^2 = 3$, $\forall \boldsymbol{x}$, and $\boldsymbol{e}(\boldsymbol{x}) = \boldsymbol{U}^*\boldsymbol{v}(\boldsymbol{x})$. For self-motion $\delta\boldsymbol{x} = (\delta r\cos\theta, \delta r \sin\theta) = \boldsymbol{q}(\theta)\delta r$, let

$$\begin{aligned}
\Lambda(\delta\boldsymbol{x}, \theta) &= \text{diag}(\exp(\langle \boldsymbol{a}_j, \delta\boldsymbol{x}\rangle), j = 1, 2, 3) \\
&= \text{diag}(\exp(\langle \boldsymbol{a}_j, \boldsymbol{q}(\theta)\rangle \delta r), j = 1, 2, 3) \\
&= \boldsymbol{I} + \text{diag}(i\langle \boldsymbol{a}_j, \boldsymbol{q}(\theta)\rangle), j = 1, 2, 3)\delta r + o(\delta r) \\
&= \boldsymbol{I} + \boldsymbol{D}(\theta)\delta r + o(\delta r).
\end{aligned} \tag{12}$$

Then

$$\begin{aligned}\boldsymbol{v}(\boldsymbol{x}+\delta\boldsymbol{x}) &= \boldsymbol{U}\boldsymbol{e}(\boldsymbol{x}+\delta x)\\ &= \boldsymbol{U}\Lambda(\delta\boldsymbol{x},\theta)\boldsymbol{e}(\boldsymbol{x})\\ &= \boldsymbol{U}\Lambda(\delta\boldsymbol{x},\theta)\boldsymbol{U}^*\boldsymbol{v}(\boldsymbol{x})\\ &= (\boldsymbol{I}+\boldsymbol{U}\boldsymbol{D}(\theta)\boldsymbol{U}^*\boldsymbol{v}(\boldsymbol{x})\delta r)\boldsymbol{v}(\boldsymbol{x})+o(\delta r)\\ &= (\boldsymbol{I}+\boldsymbol{B}(\theta)\delta r)\boldsymbol{v}(\boldsymbol{x})+o(\delta r),\end{aligned} \tag{13}$$

where $\boldsymbol{B}(\theta)=\boldsymbol{U}\boldsymbol{D}(\theta)\boldsymbol{U}^*$, and $\boldsymbol{B}(\theta)=-\boldsymbol{B}(\theta)^*$. For isotropic condition,

$$\begin{aligned}\|\boldsymbol{B}(\theta)\boldsymbol{v}(\boldsymbol{x})\|^2 &= \|\boldsymbol{D}(\theta)\boldsymbol{e}(\boldsymbol{x})\|^2\\ &= \sum_{j=1}^{3}\langle \boldsymbol{a}_j,\boldsymbol{q}(\theta)\rangle^2\\ &= \text{const}\|\boldsymbol{a}_j\|^2\|\boldsymbol{q}(\theta)\|^2=\text{const}\|\boldsymbol{a}_j\|^2,\end{aligned} \tag{14}$$

which is independent of θ, because $(\boldsymbol{a}_j, j=1,2,3)$ forms a **tight frame** in 2D.

One example of $\boldsymbol{U}$ is the following matrix:

$$\frac{1}{\sqrt{3}}\begin{pmatrix}1 & 1 & 1\\ 1 & \exp(i2\pi/3) & \exp(-i2\pi/3)\\ 1 & \exp(-i2\pi/3) & \exp(i2\pi/3)\end{pmatrix} \tag{15}$$

The resulting $(v_i(x), i=1,2,3)$ have the same orientation but different phases, i.e., they are spatially shifted versions of each other. □

The limitation of Theorem 4 is that we only show $\boldsymbol{v}(\boldsymbol{x})=\boldsymbol{U}\boldsymbol{e}(\boldsymbol{x})$ satisfies the linear model and the isotropic scaling condition, but we did not show that linear model with isotropic condition only has solutions that are hexagon grid patterns.

1.5 From group representation to orthogonal basis functions

Group representation is a central theme in modern mathematics and physics. In particular, it leads to a deep understanding and generalization of Fourier analysis or harmonic analysis.

For the set of $(\Delta\boldsymbol{x})$ that form a group, a matrix representation $\boldsymbol{M}(\Delta\boldsymbol{x})$ is equivalent to another representation $\tilde{\boldsymbol{M}}(\Delta\boldsymbol{x})$ if there exists an invertible matrix $\boldsymbol{P}$ such that $\tilde{\boldsymbol{M}}(\Delta\boldsymbol{x})=\boldsymbol{P}\boldsymbol{M}(\Delta\boldsymbol{x})\boldsymbol{P}^{-1}$ for each $\boldsymbol{x}$. A matrix representation is reducible if it is equivalent to a block diagonal matrix representation, i.e., we can find a matrix $\boldsymbol{P}$, such that $\boldsymbol{P}\boldsymbol{M}(\Delta\boldsymbol{x})\boldsymbol{P}^{-1}$ is block diagonal for every $\Delta\boldsymbol{x}$. Suppose the group is a finite group or a compact Lie group, and $\boldsymbol{M}$ is a unitary representation, i.e., $\boldsymbol{M}(\Delta\boldsymbol{x})$ is a unitary matrix. If $\boldsymbol{M}$ is block-diagonal, $\boldsymbol{M}=\text{diag}(\boldsymbol{M}_k, k=1,...,K)$, with non-equivalent blocks, and each block $\boldsymbol{M}_k$ cannot be further reduced, then the matrix elements $(M_{kij}(\Delta\boldsymbol{x}))$ are orthogonal basis functions of $\Delta\boldsymbol{x}$. Such orthogonality relations are proved by Schur [15] for finite group, and by Peter-Weyl for compact Lie group [13]. For our case, theoretically the group of displacements $\Delta\boldsymbol{x}$ in the 2D domain is $\mathbb{R}^2$, but we learn our model within a finite range, and we further discretize the range into a lattice. Thus the above orthogonal relations hold.

In our model, we also assume block diagonal $\boldsymbol{M}$, and we call each block a module. However, we do not assume each module is irreducible, i.e., each module itself may be further diagonalized into a block diagonal matrix of irreducible sub-blocks. Thus the elements within the same module $\boldsymbol{v}_k(\boldsymbol{x})$ may be linear mixings of orthogonal basis functions of the irreducible sub-blocks, and the linear mixings themselves are not necessarily orthogonal.

Fig. 2 visualizes the correlation between pairs of the learned $\boldsymbol{v}_i(\boldsymbol{x})$ and $\boldsymbol{v}_j(\boldsymbol{x})$, $i,j=1,...,d$. For different i and j, the correlations between different $\boldsymbol{v}_i(\boldsymbol{x})$ and $\boldsymbol{v}_j(\boldsymbol{x})$ are close to zero; i.e., they are nearly orthogonal to each other. The average absolute value of correlation is 0.09, and the within-block average value is about the same as the between-block average value.

Unlike previous work on learning basis expansion model (or PCA-based model [6]), we **do not constrain the basis functions $\boldsymbol{v}(\boldsymbol{x})=(v_i(\boldsymbol{x}), i=1,...,d)$ to be orthogonal to each other**. Instead, we constrain them by our path integration model via the loss term L_1. Nonetheless, the learned $v_i(\boldsymbol{x})$ are close to being orthogonal in our experiments.

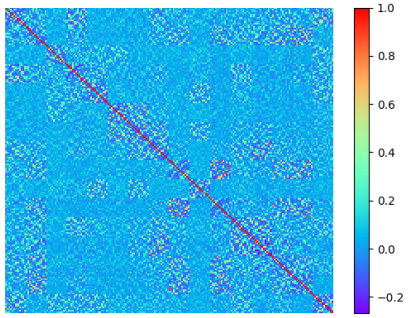


Figure 2: Correlation heatmap for each pair of the learned $v_i(\boldsymbol{x})$ and $v_j(\boldsymbol{x})$. The correlations are computed over 40×40 lattice of $\boldsymbol{x}$.

1.6 Decoding and re-encoding

In the above analysis, the projection of $\boldsymbol{v}$ onto the local 2D plane around $\boldsymbol{v}(\boldsymbol{x})$ is $\hat{\boldsymbol{x}} = \arg\min_{\boldsymbol{x}'} \|\boldsymbol{v} - \boldsymbol{v}(\boldsymbol{x}')\|^2$, which, for the linear model, amounts to decoding $\boldsymbol{v}$ to $\hat{\boldsymbol{x}}$ via

$$\hat{\boldsymbol{x}} = \arg\max_{x'} \langle \boldsymbol{v}, \boldsymbol{v}(\boldsymbol{x}') \rangle, \tag{16}$$

because $\|\boldsymbol{v}(\boldsymbol{x}')\|^2$ is constant. We project $\boldsymbol{v}$ to $\boldsymbol{v}(\hat{\boldsymbol{x}})$, which is an re-encoding of $\boldsymbol{v}$.

We can also perform decoding via the learned $\boldsymbol{u}(\boldsymbol{x}')$:

$$\hat{\boldsymbol{x}} = \arg\max_{x'} \langle \boldsymbol{v}, \boldsymbol{u}(\boldsymbol{x}') \rangle, \tag{17}$$

and re-encoding $\boldsymbol{v} \leftarrow \boldsymbol{v}(\hat{\boldsymbol{x}})$. For the above decoding, the heat map

$$\boldsymbol{h}(\boldsymbol{x}') = \langle \boldsymbol{v}, \boldsymbol{u}(\boldsymbol{x}') \rangle = \langle \boldsymbol{v}(\boldsymbol{x}), \boldsymbol{u}(\boldsymbol{x}') \rangle + \langle \boldsymbol{\varepsilon}, \boldsymbol{u}(\boldsymbol{x}') \rangle = A(\boldsymbol{x}, \boldsymbol{x}') + \boldsymbol{e}(\boldsymbol{x}'), \tag{18}$$

where $\boldsymbol{e}(\boldsymbol{x}') = \langle \boldsymbol{\varepsilon}, \boldsymbol{u}(\boldsymbol{x}') \rangle \sim \mathcal{N}(0, \sigma^2 \|\boldsymbol{v}(\boldsymbol{x})\|^2 \|\boldsymbol{u}(\boldsymbol{x}')\|^2 / d)$. For $A(\boldsymbol{x}, \boldsymbol{x}') = \exp(-\|\boldsymbol{x} - \boldsymbol{x}'\|^2/(2\sigma^2)) = \langle \boldsymbol{v}(\boldsymbol{x}), \boldsymbol{u}(\boldsymbol{x}') \rangle$, if σ^2 is small, $A(\boldsymbol{x}, \boldsymbol{x}')$ decreases to 0 quickly, i.e., if $\|\boldsymbol{x}' - \boldsymbol{x}\| > c$, then $A(\boldsymbol{x}, \boldsymbol{x}') < \exp(-c^2/(2\sigma^2))$, and the chance for the maximum of $\boldsymbol{h}(\boldsymbol{x}')$ to be achieved at an $\boldsymbol{x}'$ so that $\|\boldsymbol{x}' - \boldsymbol{x}\| > c$ can be very small. The above analysis also provides a justification for regularizing $\|\boldsymbol{u}(\boldsymbol{x}')\|^2$ in learning.

For error correction, we want to use small σ^2. However, for path planning, we need large σ^2 so that we can assess the adjacency as well as the change of the adjacency between the position on the path and the target position even if they are far apart.

In the experiments in the main text, we use Eq. (17) for decoding. In Fig. 3, we also show the results of path integration using Eq. (16) for decoding, whose performance is even better than Eq. (17). Especially the error would remain 0 over 300 time steps and 1,000 episodes using Eq. (16) with re-encoding. The advantage of (16) is that error correction is achieved within the grid cells system itself without interacting with the place cells.

1.7 Connection to continuous attractor neural network (CANN) defined on 2D torus

The CANN models [2, 4, 5, 9, 1] assume that the grid cells $\boldsymbol{v}(\boldsymbol{x}) = (v_i(\boldsymbol{x}), i = 1, ..., d)$ are placed on a finite 2D square lattice with periodic boundary condition, i.e., a 2D torus $\mathbb{T}$. If the lattice is $N \times N$, then $d = N^2$. Let $\boldsymbol{z} \in \mathbb{T}$ be the 2D coordinate of a pixel in $\mathbb{T}$, then each grid cell v_i is placed on a unique $\boldsymbol{z}_i \in \mathbb{T}$.

A CANN model hand-crafts the non-linear recurrent transformation $\boldsymbol{v}(\boldsymbol{x} + \Delta\boldsymbol{x}) = F(\boldsymbol{v}(\boldsymbol{x}), \Delta\boldsymbol{x})$ for some parametric form of F, and the coding manifold $(\boldsymbol{v}(\boldsymbol{x}), \forall \boldsymbol{x})$ consists of the attracting fixed points of $F(\cdot, 0)$. In CANN, the recurrent connection weights between a pair of grid cells (v_i, v_j) only depend on the relative positions of the two cells on the 2D torus, $\boldsymbol{z}_i - \boldsymbol{z}_j$, i.e., the connection weights

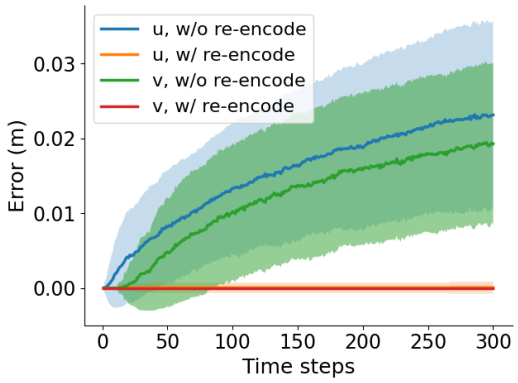


Figure 3: Path integration error over number of time steps. The mean and standard deviation band is computed over 1,000 episodes. "$\boldsymbol{v}$" means decoding by Eq. (16), and "$\boldsymbol{u}$" means decoding by Eq. (17). The squared domain is 1m $\times$ 1m.

are convolutional. Such a topographical arrangement may be physically realized on the 2D surface of the cortex in the brain, but it may also be the conceptual interpretation of the connection weights between the grid cells that are not necessarily placed on a physical 2D torus in the brain.

If we place the grid cells $\boldsymbol{v} = (v_i, i = 1, ..., d)$ on the $d = N \times N$ lattice of the 2D torus, either physically or conceptually, then their activities $\boldsymbol{v}(\boldsymbol{x}) = (v_i(\boldsymbol{x}), i = 1, ..., d)$ form an $N \times N$ "image" defined on the 2D torus. The pattern of the "image" may be a localized "bump", i.e., only a local subset of the pixels of the $N \times N$ lattice have non-zero activities. Suppose each self-position $\boldsymbol{x}$ of the agent can be mapped to a "bump" on the 2D torus centered at a corresponding $\boldsymbol{z} \in \mathbb{T}$. When the agent moves in the 2D physical space, i.e., when $\boldsymbol{x}$ changes to $\boldsymbol{x} + \Delta\boldsymbol{x}$, then the "bump" formed by $\boldsymbol{v}(\boldsymbol{x}) = (v_i(\boldsymbol{x}), i = 1, ..., d = N^2)$ moves on the 2D torus from $\boldsymbol{z}$ to $\boldsymbol{z} + \Delta\boldsymbol{z}$, while the shape of the "bump" remains the same. The connection weights of the CANN are hand-crafted so that the recurrent transformation of CANN realizes such a "mirroring" movement of the "bump".

If each displacement Δx of the agent in the 2D physical space can be mapped to a displacement $\Delta\boldsymbol{z}$ of the "bump" on the 2D torus $\mathbb{T}$, then the recurrent transformation of the CANN forms a representation of the 2D Euclidean group $\mathbb{R}^2$. If the local movement of the "bump" $\delta\boldsymbol{z}$ on the 2D torus is furthermore conformal to the local movement $\delta\boldsymbol{x}$ of the agent in the 2D physical space, then the local movement of the $d = N^2$ dimensional vector $\boldsymbol{v}(\boldsymbol{x}) = (v_i(\boldsymbol{x}), i = 1, ..., d = N^2)$ formed by the grid cells in the d-dimensional neural space, i.e., $\boldsymbol{v}(\boldsymbol{x} + \delta\boldsymbol{x}) - \boldsymbol{v}(\boldsymbol{x})$, is also conformal to the movement $\delta\boldsymbol{x}$ of the agent in the 2D physical space, and the isotropic scaling condition also holds.

In the above understanding, there are three types of movements. (1) The movement $\Delta\boldsymbol{x}$ of the agent in the 2D physical space $\mathbb{R}^2$. (2) The movement $\Delta\boldsymbol{z}$ of the "bump" on the $N \times N$ lattice of 2D torus $\mathbb{T}$. (3) The movement $\boldsymbol{v}(\boldsymbol{x} + \Delta\boldsymbol{x}) - \boldsymbol{v}(\boldsymbol{x})$ in the $d = N^2$-dimensional neural space.

Our model on either the general transformation or the linear transformation does not assume a 2D torus topography. In fact, no 2D topographical structure whatsoever is assumed in our model. The topographical arrangement is not part of our model. Instead, it may be treated as an implementation issue after the model is learned, i.e., how to arrange the grid cells physically on a 2D surface of cortex so that a pair of grid cells with strong connection weights are placed close to each other. It may also be treated as an interpretation issue after the model is learned, i.e., how to interpret the learned connection weights.

Even though our model does not make topographical assumptions, our linear transformation model appears to learn the torus topography automatically. Specifically, in our learned model, the response maps of the grid cells within each module are spatially shifted versions of the same hexagon periodic pattern. Therefore we can identify two directions in the 2D physical space that are $2\pi/3$ apart, so that $\boldsymbol{v}(\boldsymbol{x})$ rotates back to itself as $\boldsymbol{x}$ moves along these two directions for a certain distance. This implies that the codebook manifold $(\boldsymbol{v}(\boldsymbol{x}), \forall\boldsymbol{x})$ forms a 2D torus as assumed by CANN models. Moreover, the fact that the learned response maps of the grid cells within each module are spatially shifted versions of the same hexagon periodic pattern also agrees with the CANN model that moves the "bump" on the 2D torus by "mirroring" the motion in the 2D physical space. The learned hexagon

periodic patterns and the spatial shifts of the response maps may be related to the optimality of the hexagon grid in terms of sampling, interpolation and packing.

Even though the CANN model realizes the movement of the "bump" on the 2D torus by a non-linear recurrent model, such movement is a cyclic permutation of the activities of the grid cells, and the permutation can be realized by a permutation matrix, which is an orthogonal matrix. Thus the $\boldsymbol{v}(\boldsymbol{x})$ that satisfies the non-linear CANN model also satisfies our linear transformation model, where the linear rotation matrix is a cyclic permutation matrix.

The torus topology is hardly surprising, even for the general transformation model. The Lie group formed by $(F(\cdot, \Delta\boldsymbol{x}), \forall \Delta\boldsymbol{x})$ is abelian as it is a representation of the 2D additive Euclidean group $\mathbb{R}^2$. If a connected abelian Lie group is compact, then the group is automatically a torus. See [7].

Furthermore, if the scaling factor s is globally a constant for all $\boldsymbol{x}$, then the position embedding $(\boldsymbol{v}(\boldsymbol{x}), \forall \boldsymbol{x})$ is an isometric embedding up to a global scaling factor, and its intrinsic geometry remains Euclidean. It thus is a **flat torus**.

2 Experiments

2.1 Implementation details

Monte Carlo samples. The expectations in loss terms are approximated by Monte Carlo samples. Here we detail the generation of Monte Carlo samples. For $(\boldsymbol{x}, \boldsymbol{x}')$ used in $L_0 = \mathbb{E}_{\boldsymbol{x},\boldsymbol{x}'}[A(\boldsymbol{x}, \boldsymbol{x}') - \langle \boldsymbol{v}(\boldsymbol{x}), \boldsymbol{u}(\boldsymbol{x}') \rangle]^2$, $\boldsymbol{x}$ is first sampled uniformly within the entire domain, and then the displacement $d\boldsymbol{x}$ between $\boldsymbol{x}$ and $\boldsymbol{x}'$ is sampled from a normal distribution $\mathcal{N}(0, \sigma^2 \boldsymbol{I}_2)$, where $\sigma = 0.48$. This is to ensure that nearby samples are given more emphasis. We let $\boldsymbol{x}' = \boldsymbol{x} + d\boldsymbol{x}$, and those pairs $(\boldsymbol{x}, \boldsymbol{x}')$ within the range of domain (i.e., 1m $\times$ 1m, 40×40 lattice) are kept as valid data. For $(\boldsymbol{x}, \Delta\boldsymbol{x})$ used in $L_1 = \mathbb{E}_{\boldsymbol{x},\Delta\boldsymbol{x}} |\boldsymbol{v}(\boldsymbol{x} + \Delta\boldsymbol{x}) - \exp(\boldsymbol{B}(\theta)\Delta r)\boldsymbol{v}(\boldsymbol{x})|^2$, $\Delta\boldsymbol{x}$ is sampled uniformly within a circular domain with radius equal to 3 grids and $(0, 0)$ as the center. Specifically, Δr^2, the squared length of $\Delta\boldsymbol{x}$, is sampled uniformly from $[0, 3]$ grids, and θ is sampled uniformly from $[0, 2\pi]$. We take the square root of the sampled Δr^2 as Δr and let $\Delta\boldsymbol{x} = (\Delta r \cos\theta, \Delta r \sin\theta)$. Then $\boldsymbol{x}$ is uniformly sampled from the region such that both $\boldsymbol{x}$ and $\boldsymbol{x} + \Delta\boldsymbol{x}$ are within the range of domain. For $(\theta, \Delta\theta)$ used in $L_2 = \sum_{k=1}^{K} \mathbb{E}_{\boldsymbol{x},\theta,\Delta\theta}[\|\boldsymbol{B}_k(\theta + \Delta\theta)\boldsymbol{v}_k(\boldsymbol{x})\| - \|\boldsymbol{B}_k(\theta)\boldsymbol{v}_k(\boldsymbol{x})\|]^2$, we uniformly sample θ and $\theta + \Delta\theta$ from discretized angles, i.e., 144 directions discretized for circle $[0, 2\pi]$. We will study sampling only small $\Delta\theta$ in the future.

Training details. The model is trained for 14,000 iterations. At each iteration, the samples are generated online. For the first 8,000 iterations, we update all learnable parameters, while for the following iterations, we fix the learned $\boldsymbol{v}(\boldsymbol{x})$ and update the other learnable parameters. The initial learning rate is set as 0.003 and is decreased by a factor of 0.5 every 500 iterations after 8,000 iterations. We use Adam [8] optimizer. The model is trained on a single Titan XP GPU. We apply the maximum batch size that can fit into the single GPU, which is 90,000. It takes about 3.5 hours to train the model on a single Titan XP GPU.

Baseline methods. In Table 1 of the main text, we compare the learned neurons with the ones from other two optimization-based learning methods [3, 11]. For [3], we run the code released by the authors (`https://github.com/deepmind/grid-cells`) to learn the model and compute gridness scores for the learned neurons. For [11], we use the pre-trained weights released by the authors (`https://github.com/ganguli-lab/grid-pattern-formation`) to get the learned neurons and compute the gridness scores. Both the code of [3] and pre-trained weights of [11] use Apache License V2.

Usage of data. In this paper, we mainly use simulated trajectories as training data, and thus we do not think that the data contain any personally identifiable information or offensive content. The only existing data we use is the pre-trained weights of the baseline method [11]. Under Apache License V2, we believe it is fully approved by the authors to use the pre-trained weights.

2.2 Learned patterns

Fig. 4 displays the autocorrelograms of learned patterns of $\boldsymbol{v}(\boldsymbol{x})$.

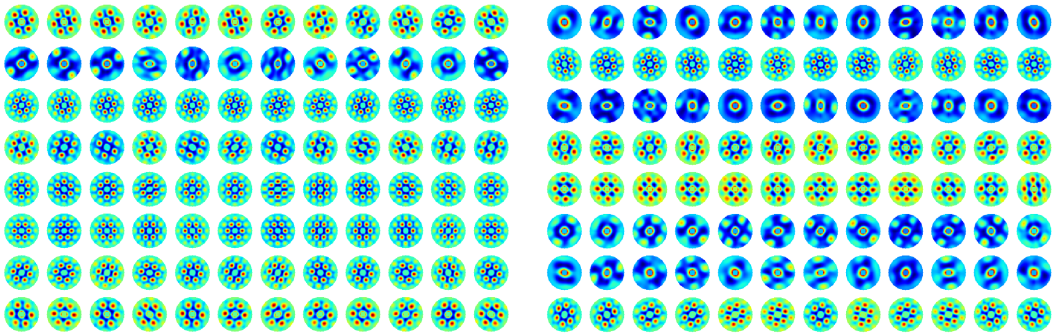


Figure 4: Autocorrelograms of the learned patterns of $\boldsymbol{v}(\boldsymbol{x})$.

Fig. 5 shows the learned patterns of $\boldsymbol{u}(\boldsymbol{x})$ with 16 blocks of 12 cells in each block. Regular hexagon patterns also emerge.

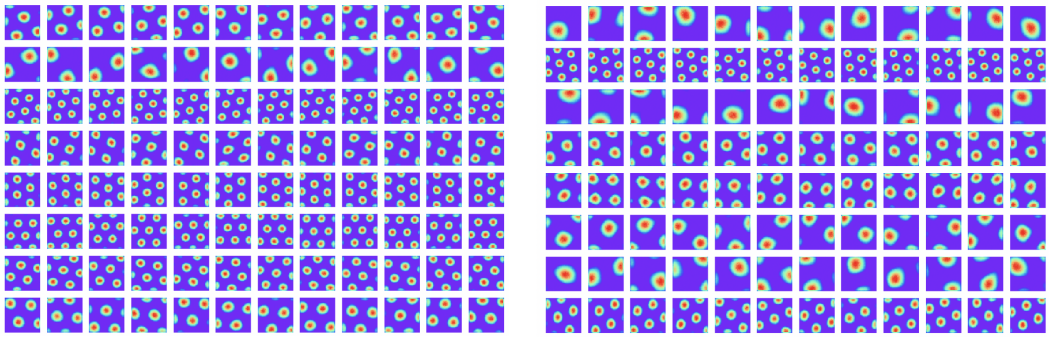


Figure 5: Learned patterns of $\boldsymbol{u}(\boldsymbol{x})$ with 16 blocks of size 12 cells in each block. Every row shows the learned patterns within the same block.

For learned firing patterns of $\boldsymbol{v}(\boldsymbol{x})$, we also display the histogram of grid orientations in Fig. 6, where we do not observe clear clusters.

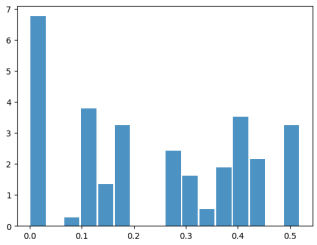


Figure 6: Histogram of grid orientations of the learned firing patterns of $\boldsymbol{v}(\boldsymbol{x})$.

In Fig. 7, we show the learned patterns of a block of $\boldsymbol{B}(\theta)$. Each element shows significant sine/cosine tuning over θ. For the other blocks, the patterns are all similar.

Gaussian kernel. Because $A(\boldsymbol{x}, \boldsymbol{x}')$ is a sharp Gaussian kernel, it contains a whole range of frequencies in the 2D Fourier domain. The learned response maps of the grid cells span a range of frequencies or scales too. Each module or block focuses on a certain frequency band, which corresponds to the metric of the module. We assume individual place field $A(\boldsymbol{x}, \boldsymbol{x}')$ to exhibit a Gaussian shape, rather than a Mexican-hat pattern (with balanced excitatory center and inhibitory

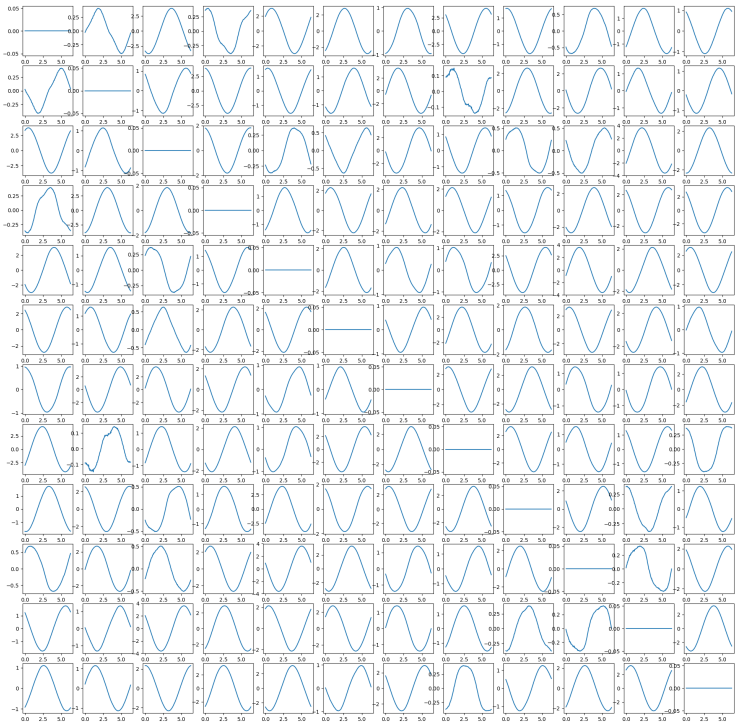


Figure 7: Learned patterns of a block of $\boldsymbol{B}(\theta)$. Each subfigure shows the value of an element in $\boldsymbol{B}(\theta)$ (vertical axis) over θ (horizontal axis).

surround) as assumed in previous basis expansion models [6, 11] of grid cells. The Mexican-hat or difference of Gaussians pattern occupies a ring in the 2D Fourier domain. It corresponds to a module in our model. But we use isotropic condition to enforce each module to be within a ring in the Fourier domain, and we use different modules to pave the whole Fourier domain.

2.3 Error correction

We begin by assessing the ability of error correction of the learned system following the setting in Proposition 1. Specifically, for a given location $\boldsymbol{x}$, suppose the neurons are perturbed by Gaussian noise: $\boldsymbol{v} = \boldsymbol{v}(x) + \varepsilon$, where $\varepsilon \sim \mathcal{N}(0, \tau^2 \boldsymbol{I}_d)$ and $\tau^2 = \alpha^2(\|\boldsymbol{v}(\boldsymbol{x})\|^2/d)$, so that α^2 measures the variance of noise relative to the average magnitude of $(v_i(\boldsymbol{x})^2, i = 1, ..., d)$ and α measures the relative standard deviation. We infer the 2D position $\hat{\boldsymbol{x}}$ from $\boldsymbol{v}$ by $\hat{\boldsymbol{x}} = \arg\min_{\boldsymbol{x}'} \|\boldsymbol{v} - \boldsymbol{v}(\boldsymbol{x}')\|^2$. Fig. 8 displays the inference error over the relative standard deviation α of the added Gaussian noise. We also show the results using the learned $\boldsymbol{u}(\boldsymbol{x}')$ for inference (Eq. (17)). The system works remarkably well even if $\alpha = 2$.

We further assess the ability of error correction in long distance path integration. Specifically, along the way of path integration, at every time step t, two types of errors are introduced to $\boldsymbol{v}_t$: (1) Gaussian noise or (2) dropout masks, i.e., certain percentage of units are randomly set to zero. Fig. 9 summarizes the path integration performance with different levels of injected errors for $T = 100$, using $\boldsymbol{v}(\boldsymbol{x}')$ (Eq. (16)) or $\boldsymbol{u}(\boldsymbol{x}')$ (Eq. (17)) for decoding. The results show that re-encoding at each step helps error correction, especially for dropout masks. For Gaussian noise, even without decoding and re-encoding at each step, decoding at the final step alone is capable of removing much of the noise. Notably, with re-encoding, the path integration works well even if Gaussian noise with $\alpha = 1$ is added or 50% units are randomly dropped out at each step, indicating that the learned system is robust to different sources of errors.

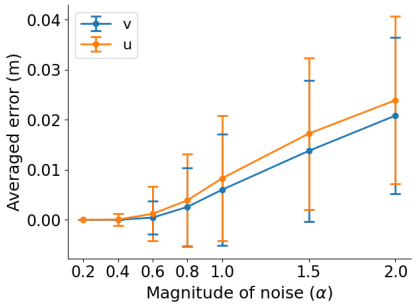


Figure 8: Error correction results following the setting in Proposition 1. The error bar stands for the standard deviation over 1,000 trials. "$\boldsymbol{v}$" means decoding by Eq. (16), and "$\boldsymbol{u}$" means decoding by Eq. (17). The squared domain is 1m $\times$ 1m.

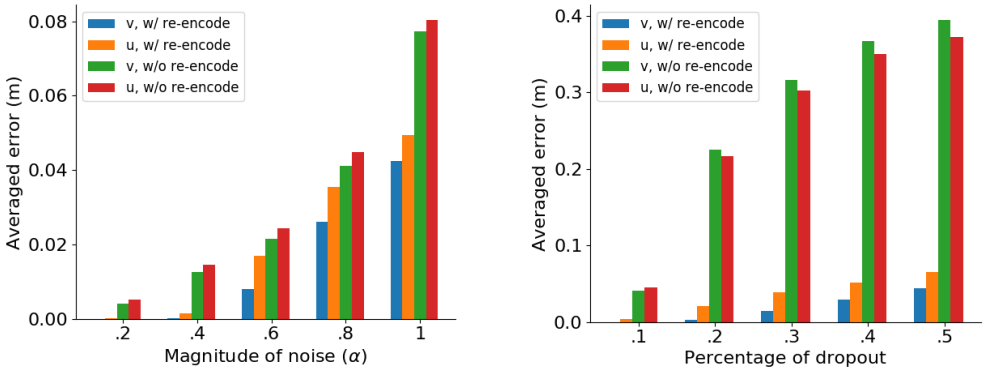


Figure 9: Path integration results with different levels of injected errors. *Left*: Gaussian noise. The magnitude of noise is measured using the average of the squared magnitudes of the units in $\boldsymbol{v}(\boldsymbol{x})$ as the reference. *Right*: dropout masks. Certain percentage of units are randomly set to zero at each step. "$\boldsymbol{v}$" means decoding by Eq. (16), and "$\boldsymbol{u}$" means decoding by Eq. (17). The squared domain is 1m $\times$ 1m.

2.4 Non-linear transformation model

We test our method with a non-linear transformation model:

$$F(\boldsymbol{v}(\boldsymbol{x}), \Delta r, \theta) = \text{ReLU}(\exp(\boldsymbol{B}(\theta)\Delta r)\boldsymbol{v}(\boldsymbol{x})), \tag{19}$$

where we insert $\text{ReLU}(a) = \max(0, a)$ into the linear transformation model.

We use numerical differentiation to define directional derivative

$$f_\theta(\boldsymbol{v}(\boldsymbol{x})) = [\boldsymbol{v}(\boldsymbol{x} + \boldsymbol{\delta x}) - \boldsymbol{v}(\boldsymbol{x})]/\delta r, \tag{20}$$

where $\boldsymbol{\delta x} = (\delta r \cos\theta, \delta r \sin\theta)$, with pre-defined δr. The reason for numerical differentiation is because the derivative of ReLU is an indicator function, which is not differentiable. $f_\theta(\boldsymbol{v}(\boldsymbol{x}))$ needs to be differentiable for minimizing the loss function (an alternative to numerical differentiation is to use sigmoid function to approximate the indicator function).

We continue to use the same loss function except with the above two changes. Interestingly, regular hexagon patterns continue to emerge (average gridness score 0.83, percentage of grid cells 70.21%). See Fig. 10 for the learned patterns of $\boldsymbol{v}(\boldsymbol{x})$.

2.5 Path planning

Our grid cells model can be applied to path planning. Specifically, according to [12], the adjacency kernel can be modeled by

$$A_\gamma(\boldsymbol{x}, \boldsymbol{x}') = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t 1(\boldsymbol{x}_t = \boldsymbol{x}') | \boldsymbol{x}_0 = \boldsymbol{x}\right] = \langle \boldsymbol{v}(\boldsymbol{x}), \boldsymbol{u}_\gamma(\boldsymbol{x}') \rangle, \tag{21}$$

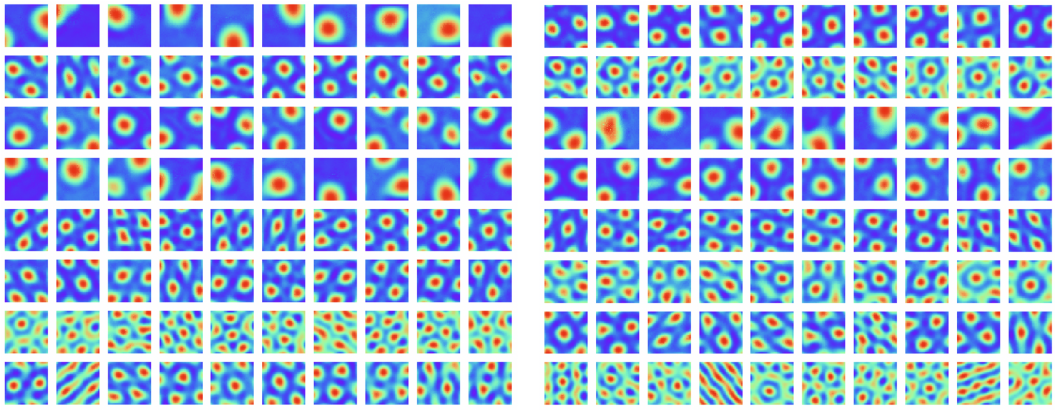


Figure 10: Learned patterns of $v(x)$ with the non-linear transformation model (Eq. (19)). Every row shows the learned patterns within the same block.

where γ is the discount factor that controls the temporal and spatial scales, $\mathbb{E}$ is with respect to a random walk exploration policy, and $1(\cdot)$ is the indicator function. For random walk in open field, $A_\gamma(x, x') \propto \exp(-\|x - x'\|^2/2\sigma_\gamma^2)$, where σ_γ^2 depends on γ.

To enable path planning, we need kernels of both big and small spatial scales to account for long and short distance planning respectively. To this end, we discretize γ into a finite list of scales, and learn a list of corresponding $u_\gamma(x')$ together with $v(x)$ and $B(\theta)$ using the loss function in Section 5 of the main text.

With the learned model, path planning can be accomplished by steepest ascent on the adjacency to the target position. Specifically, let $\hat{x}$ be the target or destination. Let $x^{(t)}$ be the current position in the path planning process, encoded by $v(x^{(t)})$. The agent plans the next displacement by steepest ascent on

$$A_\gamma(x^{(t)} + \Delta x, \hat{x}) = \langle v(x^{(t)} + \Delta x), u_\gamma(\hat{x}) \rangle = \langle M(\Delta x) v(x^{(t)}), u_\gamma(\hat{x}) \rangle, \tag{22}$$

over allowed Δx within a single step, where $M(\Delta x) = \exp(B(\theta)\Delta r)$, with $\Delta x = (\Delta r \cos\theta, \Delta r \sin\theta)$. We plan

$$\Delta x^{(t+1)} = \arg\max_{\Delta x} A_\gamma(x^{(t)} + \Delta x, \hat{x}), \tag{23}$$

and let $x^{(t+1)} = x^{(t)} + \Delta x^{(t+1)}$.

The scale γ is selected as the smallest one that satisfies $\max_{\Delta x} \langle M(\Delta x) v(x^{(t)}), u_\gamma(\hat{x}) \rangle > .2$. We can also use $\max_\gamma \max_{\Delta x} \langle M(\Delta x) v(x^{(t)}), u_\gamma(\hat{x}) \rangle$ for scale selection.

We test path planning in the open field environment. The model is first learned using a single-scale kernel function $A_\gamma(x, x') = \exp(-\|x - x'\|^2/2\sigma_\gamma^2)$ where $\sigma_\gamma = 0.07$. Then we assume a list of three scales: $\sigma_\gamma = [0.07, 0.14, 0.28]$ and learn the corresponding list of $u_\gamma(x')$. The pool of allowed displacements for a single step is defined as: dr can be 1 or 2 grids, while θ can be chosen from 200 discretized angles over $[0, 2\pi]$. Fig. 11 demonstrates several examples of path planning in the open field environment, where the agent is able to plan straight path to the target. When $x^{(t)}$ is far from the target, kernel with large σ_γ is chosen, and as $x^{(t)}$ approaches the target, the chosen kernel gradually switches to the one with small σ_γ. A planning episode is treated as a success if the distance between $x^{(t)}$ and target is smaller than 0.5 grid within 40 time steps. The agent achieves a success rate of 100% (tested for 10,000 episodes).

For a field with obstacles or rewards, we can learn the deformed $A_\gamma(x, x')$ and $(v(x), u_\gamma(x'))$ by temporal difference learning with a random walk exploration policy as suggested in [12]. After learning $A_\gamma(x, x')$ and $(v(x), u_\gamma(x'))$, we can continue to use Eq. (23) for path planning. We shall further study it in future work.

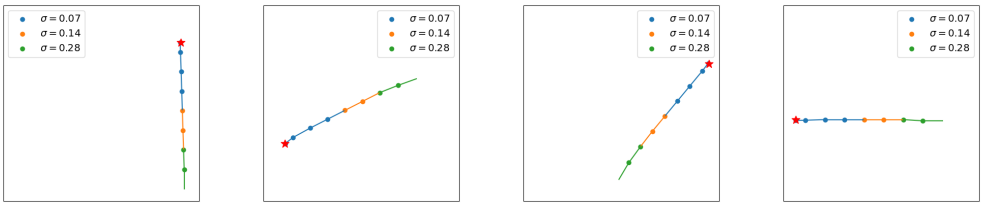


Figure 11: Examples of path planning results in an open field environment. The target is shown as a red star.

2.6 Integrating egocentric vision

When the agent moves in darkness, it can infer its self-position by integrating self-motion, as illustrated by our experiments on path integration. If there is visual input, the agent can infer its self-position (as well as head direction) from the visual image alone. We extend our grid cells model to study this problem of egocentric vision, which is important in computer vision.

Specifically, suppose the agent navigates in a 3D scene such as a room, and the height of the eye (or camera) remains fixed. Suppose at 2D self-position $\boldsymbol{x}$ and with head direction θ, the agent sees an image $\boldsymbol{I}$, which is called a posed image. We use the vector representation $\boldsymbol{v}(\boldsymbol{x})$ in our original grid cells model to represent the 2D self-position $\boldsymbol{x}$, and use another vector representation $\boldsymbol{h}(\theta)$ to represent the head direction θ. If the agent changes its head direction from θ to $\theta + \Delta\theta$, $\boldsymbol{h}(\theta)$ is transformed to

$$\boldsymbol{h}(\theta + \Delta\theta) = \exp(\boldsymbol{C}\Delta\theta)\boldsymbol{h}(\theta). \tag{24}$$

We assume that there are K modules or blocks in $\boldsymbol{h}(\theta)$ and $\boldsymbol{C}$ is skew-symmetric. This is similar to the transformation of $\boldsymbol{v}(\boldsymbol{x})$ in our grid cells model.

$(\boldsymbol{x}, \theta)$ is called the pose of the camera (or eye), and we call $(\boldsymbol{v}(\boldsymbol{x}), \boldsymbol{h}(\theta))$ the pose embedding.

To associate the pose embedding $(\boldsymbol{v}(\boldsymbol{x}), \boldsymbol{h}(\theta))$ with the posed image $\boldsymbol{I}$, we use a vector representation or scene embedding $\boldsymbol{s}$ to represent the 3D scene which is shared across different posed images of the same scene, and we learn a generator network G_β that maps the embeddings $\boldsymbol{s}$ and $(\boldsymbol{v}(\boldsymbol{x}), \boldsymbol{h}(\theta))$ to the posed image $\boldsymbol{I}$:

$$\boldsymbol{I} = G_\beta(\boldsymbol{s}, \boldsymbol{v}(\boldsymbol{x}), \boldsymbol{h}(\theta)) + \varepsilon, \tag{25}$$

where the generator G_β is parametrized by a multi-layer deconvolutional neural network with parameters β, and ε is the residual error.

Given the above assumptions, we introduce two extra loss terms in addition to the loss function described in Section 5 of the main text.

$$L_3 = \sum_{k=1}^{K} \mathbb{E}_{\theta, \Delta\theta} \|\boldsymbol{h}_k(\theta + \Delta\theta) - \exp(\boldsymbol{C}_k \Delta\theta)\boldsymbol{h}_k(\theta)\|^2, \tag{26}$$

$$L_4 = \mathbb{E}\|\boldsymbol{I} - G_\beta(\boldsymbol{s}, \boldsymbol{v}(\boldsymbol{x}), \boldsymbol{h}(\theta))\|^2. \tag{27}$$

L_3 is to model the head rotation, and L_4 is to model the generation of the posed image.

During training, we alternatively update $(G_\beta, \boldsymbol{s})$ and $(\boldsymbol{v}(\boldsymbol{x}), \boldsymbol{B}(\theta), \boldsymbol{u}(\boldsymbol{x}'), \boldsymbol{h}(\theta), \boldsymbol{C})$ by gradient descent on the overall loss function that is a linear combination of L_0, L_1 and L_2 in the main text, as well as L_3 and L_4 introduced above.

The learned model enables two useful applications:

(a) **Novel view synthesis**. Given an unseen pose $(\boldsymbol{x}, \theta)$, the model can predict the corresponding posed image by $G_\beta(\boldsymbol{s}, \boldsymbol{v}(\boldsymbol{x}), \boldsymbol{h}(\theta))$.

(b) **Inference of pose**, i.e., self-position $\boldsymbol{x}$ and head direction θ, from posed image $\boldsymbol{I}$ alone. Specifically, after training the model, we can learn an additional inference network F_ξ that maps an observed posed image $\boldsymbol{I}$ to its pose embedding $\boldsymbol{v}(\boldsymbol{x})$ and $\boldsymbol{h}(\theta)$. The inference network is learned by minimizing the ℓ_2 distance between the predicted and true pose embeddings: $\mathbb{E}\|(\boldsymbol{v}(\boldsymbol{x}), \boldsymbol{h}(\theta)) - F_\xi(\boldsymbol{I})\|^2$. Then

Table 1: Average error of pose inference.

	x_1	x_2	θ
Error	.0225m	.0230m	1.37°

given an unseen posed image $\boldsymbol{I}$, we can infer the pose by $\arg\min_{\boldsymbol{x},\theta} \|(\boldsymbol{v}(\boldsymbol{x}), \boldsymbol{h}(\theta)) - F_\xi(\boldsymbol{I})\|^2$. In this task, $F_\xi(\boldsymbol{I})$ is the estimate of $(\boldsymbol{v}(\boldsymbol{x}), \boldsymbol{h}(\theta))$, and it is likely that this estimate contains error. This error will translate to the error in the estimated $(\boldsymbol{x}, \theta)$. Thus our theoretical analysis of error translation in the main text is highly relevant, and the isotropic scaling condition is motivated by the analysis of error translation.

We conduct experiments on a dataset generated by the Gibson Environment [14], which provides tools for rendering images of different poses in 3D rooms. Specifically, we select 20 areas of size 2m $\times$ 2m from different rooms and render about 28k 64 $\times$ 64 RGB posed images for each area. The camera height is fixed and the camera can only rotate horizontally. The scene embedding vector $\boldsymbol{s}$ is of 512 dimensions. Both $\boldsymbol{v}(\boldsymbol{x})$ and $\boldsymbol{h}(\theta)$ are of 192 dimensions, partitioned into $K = 16$ modules.

Hexagon patterns still emerge in the learned $\boldsymbol{v}(\boldsymbol{x})$ (average gridness score 0.71). For novel view synthesis, we evaluate the performance on 374k testing posed images. The resulting peak signal-to-noise ratio (PSNR) between synthesized images and ground truth images is 25.17, indicating that the model can generate reasonable unseen posed images. Fig. 12 demonstrates several examples of the novel view synthesis results.

Figure 12: Examples of synthesizing novel views. *Left*: Ground truth unseen posed images. *Right*: synthesized unseen posed images.

For inference of pose (self-position $\boldsymbol{x} = (x_1, x_2)$ and head direction θ), we evaluate the performance on the same 374k testing posed images and report the average inference error in Table 1. The estimates are reasonably accurate.

2.7 Ablation studies

Isotropic scaling condition is necessary for hexagon grid patterns. A natural question is whether the isotropic scaling condition (condition 2) is important for learning hexagon grid patterns. To verify this, we learn the model by removing the loss term L_2 (Eq. (19) in the main text) from the loss function, which constrains the model to meet condition 2. As shown in Fig. 13, more strip-like patterns emerge without L_2, indicating that condition 2 is important for hexagon grid patterns to emerge.

Assumption of $\boldsymbol{u}(\boldsymbol{x}') \geq 0$ is not necessary for hexagon grid patterns. During training, we make an assumption of $\boldsymbol{u}(\boldsymbol{x}') \geq 0$ to make sure the connections from grid cells to place cells are excitatory [16, 10]. However, we want to emphasize this is not a key assumption in our model. Fig. 14 demonstrates the learned neurons in the network without assuming $\boldsymbol{u}(\boldsymbol{x}') \geq 0$, where hexagonal grid firing patterns also emerge. The average gridness score is 0.82 and the percentage of grid cells is 87.50%. However, the grid activations can be either positive/excitatory (in red color) or negative/inhibitory (in blue color).

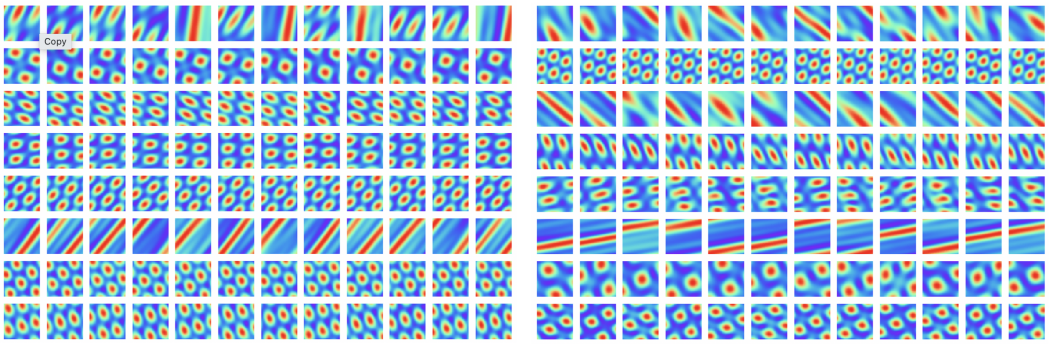


Figure 13: Learned neurons without loss term L_2, which is the constraint on isotropic scaling condition. More strip-like firing patterns emerge.

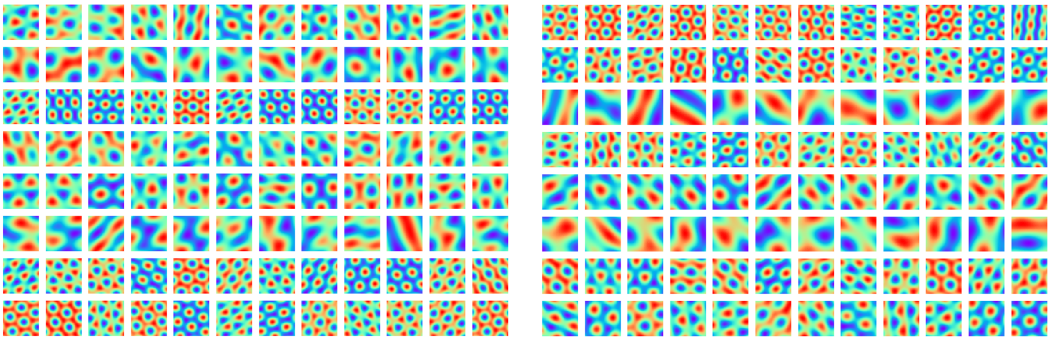


Figure 14: Learned neurons without the assumption of $\boldsymbol{u}(\boldsymbol{x}') \geq 0$. Hexagonal grid firing patterns also emerge, with the grid activations being either positive/excitatory (in red color) or negative/inhibitory (in blue color).

Skew-symmetric assumption of $\boldsymbol{B}(\theta)$ is not important for hexagon grid patterns. To make the linear transformation a rotation, we have assumed that $\boldsymbol{B}(\theta)$ is skew-symmetric, i.e., $\boldsymbol{B}(\theta) = -\boldsymbol{B}(\theta)^\top$. Nonetheless, this assumption is not important for the emergence of hexagon grid patterns. Fig. 15 demonstrates the learned neurons without assuming that $\boldsymbol{B}(\theta)$ is skew-symmetric. Hexagon grid firing patterns emerge in most of the neurons, with only one block of square grid firing patterns.

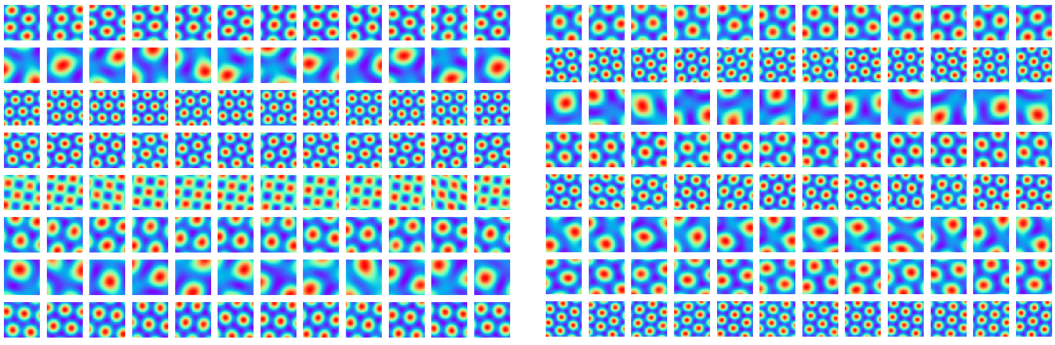


Figure 15: Learned neurons without skew-symmetric assumption of $\boldsymbol{B}(\theta)$. Hexagonal grid firing patterns emerge in most of the neurons, with a block of square grid firing patterns.

Number and sizes of blocks do not matter. It is worthwhile to mention that the emergence of hexagonal grid firing patterns in the learned neurons are not due to specific design of the block size or the number of blocks. Fig. 16 visualizes the learned neurons by fixing the total number of neurons at 192 and altering the block size and number of blocks. Hexagon patterns emerge in all the settings.

Multiple blocks or modules are necessary for learning grid patterns of multiple scales. We further try to fully remove the assumption of blocks or modules; i.e., we learn a single block of $\boldsymbol{B}(\boldsymbol{\theta})$. Fig. 17 shows the learned neurons and the corresponding autocorrelograms. All the learned neurons share similar large scales, which indicates that the high frequency part of $A(\boldsymbol{x}, \boldsymbol{x}')$ may not be fitted very well.

References

[1] Haggai Agmon and Yoram Burak. A theory of joint attractor dynamics in the hippocampus and the entorhinal cortex accounts for artificial remapping and grid cell field-to-field variability. *eLife*, 9:e56894, 2020.

[2] Daniel J Amit and Daniel J Amit. *Modeling brain function: The world of attractor neural networks*. Cambridge university press, 1992.

[3] Andrea Banino, Caswell Barry, Benigno Uria, Charles Blundell, Timothy Lillicrap, Piotr Mirowski, Alexander Pritzel, Martin J Chadwick, Thomas Degris, Joseph Modayil, et al. Vector-based navigation using grid-like representations in artificial agents. *Nature*, 557(7705):429, 2018.

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

[5] Jonathan J Couey, Aree Witoelar, Sheng-Jia Zhang, Kang Zheng, Jing Ye, Benjamin Dunn, Rafal Czajkowski, May-Britt Moser, Edvard I Moser, Yasser Roudi, et al. Recurrent inhibitory circuitry as a mechanism for grid formation. *Nature neuroscience*, 16(3):318–324, 2013.

[6] Yedidyah Dordek, Daniel Soudry, Ron Meir, and Dori Derdikman. Extracting grid cell characteristics from place cell inputs using non-negative principal component analysis. *Elife*, 5:e10094, 2016.

[7] William Gerard Dwyer and CW Wilkerson. The elementary geometric structure of compact lie groups. *Bulletin of the London Mathematical Society*, 30(4):337–364, 1998.

[8] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[9] Hugh Pastoll, Lukas Solanka, Mark CW van Rossum, and Matthew F Nolan. Feedback inhibition enables theta-nested gamma oscillations and grid firing fields. *Neuron*, 77(1):141–154, 2013.

[10] David C Rowland, Horst A Obenhaus, Emilie R Skytøen, Qiangwei Zhang, Cliff G Kentros, Edvard I Moser, and May-Britt Moser. Functional properties of stellate cells in medial entorhinal cortex layer ii. *Elife*, 7:e36664, 2018.

[11] Ben Sorscher, Gabriel Mel, Surya Ganguli, and Samuel A Ocko. A unified theory for the origin of grid cells through the lens of pattern formation. 2019.

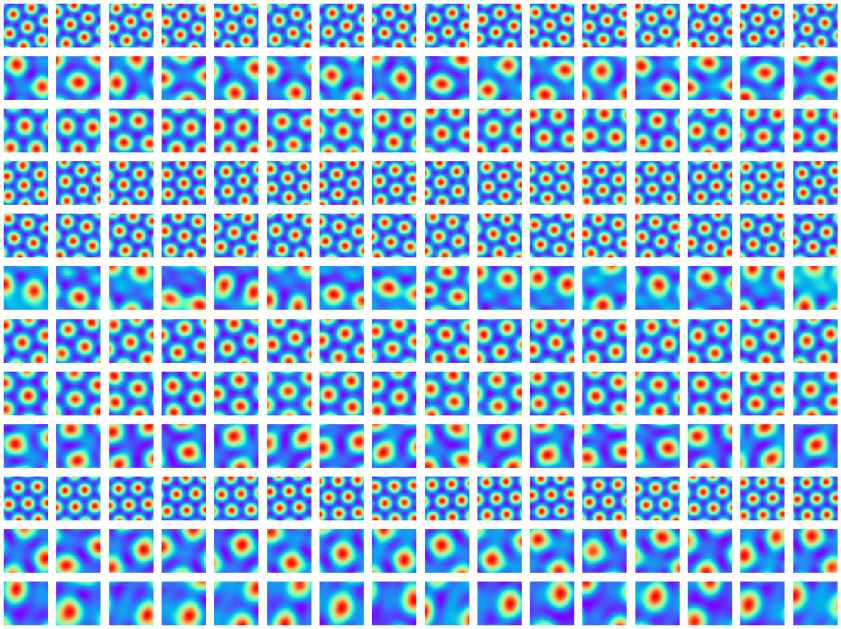
[12] Kimberly L Stachenfeld, Matthew M Botvinick, and Samuel J Gershman. The hippocampus as a predictive map. *Nature neuroscience*, 20(11):1643, 2017.

[13] Michael Taylor. Lectures on lie groups. *Lecture Notes, available at http://www. unc. edu/math/Faculty/met/lieg. html*, 2002.

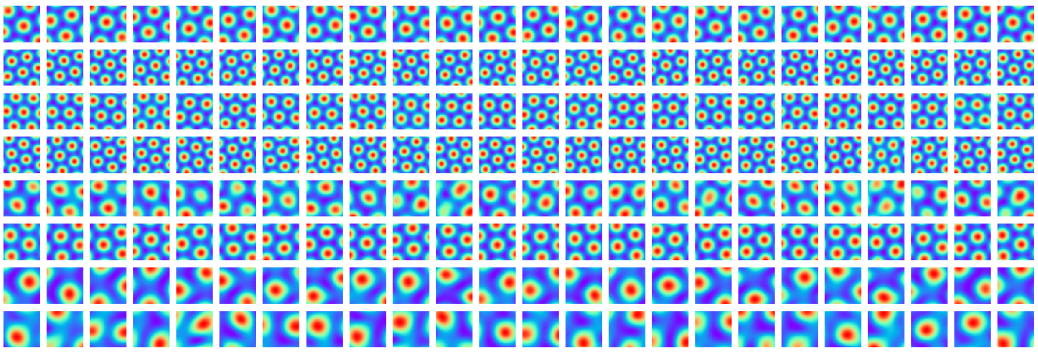
[14] Fei Xia, Amir R Zamir, Zhiyang He, Alexander Sax, Jitendra Malik, and Silvio Savarese. Gibson env: Real-world perception for embodied agents. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9068–9079, 2018.

[15] Anthony Zee. *Group theory in a nutshell for physicists*. Princeton University Press, 2016.

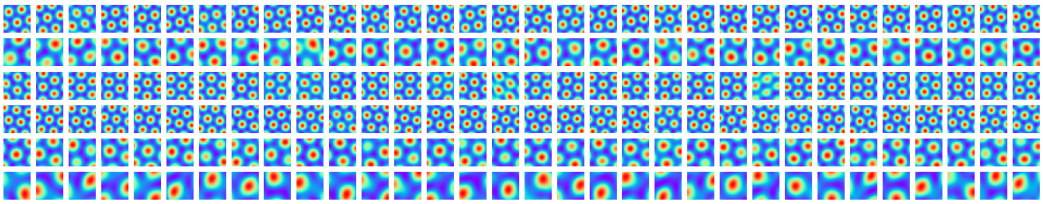
[16] Sheng-Jia Zhang, Jing Ye, Chenglin Miao, Albert Tsao, Ignas Cerniauskas, Debora Ledergerber, May-Britt Moser, and Edvard I Moser. Optogenetic dissection of entorhinal-hippocampal functional connectivity. *Science*, 340(6128), 2013.



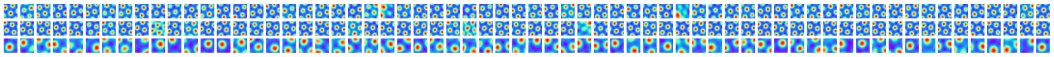
(a) Block size = 16



(b) Block size = 24



(c) Block size = 32



(d) Block size = 64

Figure 16: Learned patterns of $\boldsymbol{v}(\boldsymbol{x})$ with different block sizes. The total number of units is fixed at 192. Every row shows the learned patterns within the same block.

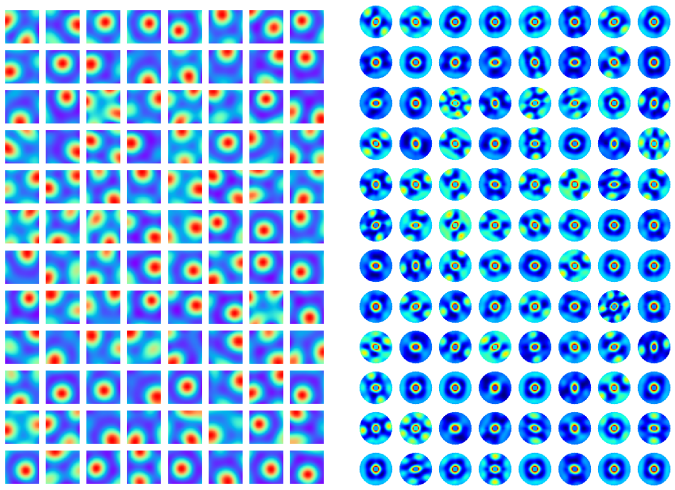


Figure 17: *Left*: learned neurons with a single block of $\boldsymbol{B}(\theta)$. The firing patterns has a single large scale, meaning that the high frequency part of $A(\boldsymbol{x}, \boldsymbol{x}')$ is not fitted very well. *Right*: autocorrelograms of the learned neurons. Some exhibit clear hexagon grid patterns, while the other do not, probably because the scale of those grid patterns are beyond the scope of the whole area.