Using Grid Cells for Navigation

*Daniel Bush, *Caswell Barry, Daniel Manson, Neil Burgess

Neuron (in press)

Contact: drdanielbush@gmail.com

This readme file accompanies Matlab code (available at http://modeldb.yale.edu/182685) that can be used to run the simulations described in the manuscript, and replicate figures S2-S5.

Background

Grid cells in the rodent medial entorhinal cortex (mEC) exhibit multiple spatial firing fields distributed in a regular hexagonal array that covers all environments visited by the animal. Grid cells are organised into functional modules within mEC: cells that are proximate in the brain tend to have firing patterns that share the same scale and orientation but a fixed spatial offset relative to one another (i.e. exhibit a different spatial 'phase'). Importantly, the relative spatial phase of any two simultaneously recorded grid cells from the same module appears to be conserved across all environments visited by the animal, and a small number of grid firing patterns can completely cover the environment. Grid scale increases between modules in discontinuous steps along the dorso-ventral axis of mEC, with the smallest being around 25cm and the largest so far recorded exceeding 300cm and probably representing the fourth or fifth of up to ten discrete scales. The orientations of grid firing patterns in different modules are also clustered.

The regular periodic firing patterns of grid cells potentially provide a compact code for location that resembles a residue number system, encoding positions over a very large range that approaches the lowest common multiple of the spatial scales of all grid modules. By providing a context-independent spatial metric, grid cells – unlike place cells - have the potential to solve the problem of vector navigation: to compute a direct translation vector between arbitrary locations, even when those locations are much farther apart than the largest grid scale. Here, we present four neural network models of vector navigation that make use of the grid cell representation of space.

Each model takes input from M=10 grid cell modules whose spatial scales are arranged in a geometric progression (i.e. $s_i = s_M \alpha^{M-i}$) with a minimum grid scale of s_{10}=25cm and a common factor of α=1.4 (giving a maximum grid scale of ~5m). Each module consists of N_{GC}=400 grid cells distributed evenly among m_i=20 equally distributed spatial phases p_i. We note that the exact distribution of grid scales has no effect on the simulations presented here, aside from determining the capacity of the grid cell system to encode unique locations and the ability of the grid cell system to deal with neural noise. For each model, translation vectors are decoded for N=1000 randomly assigned current and goal locations in a 2D arena. These locations are constructed from randomly assigned displacements along two 500m principal axes of that arena. For simplicity, these axes are oriented at $\vec{x} = 0°$ and $\vec{y} = 60°$ to match the principal axes of the grid pattern.

The Distance Cell Model

In distance cell simulations, distance cells encode current or goal locations in allocentric space along the principal axes $\vec{x}$ and $\vec{y}$. Two distance cell arrays (one for each of the principal axes $\vec{x}$ and $\vec{y}$) receive input from a population of grid cells whose firing rates encode the current location, and two separate distance cell arrays (one for each of the principal axes $\vec{x}$ and $\vec{y}$) receive input from a population of grid cells whose firing rates encode the goal location. Grid cells project to distance cells in each array with synaptic weights that are proportional to their mean firing rate at the allocentric location a encoded by that distance cell on the directional axis $\vec{x}$ or $\vec{y}$, i.e. $w_{DC,GC} \propto r_{j,i}(a)$, analogous to models of the grid cell to place cell transformation. The number of spikes fired by each grid cell is dictated by a Poisson process with a cosine tuned rate function and a time window of 100ms.

Distance cells have a spatial resolution of 0.04m to give N_{DC}=12500 distance cells in each of the four arrays (corresponding to current and goal locations on the two axes $\vec{x}$ and $\vec{y}$), or a total of 50000 distance cells overall. Distance cells within each array are subject to winner-take-all dynamics implemented with an E%-max algorithm, such that cells only fire if their input is within *k*=1% of the maximum feed-forward excitation. Activity within each distance cell array is then normalised, such that the integrated activity across all distance cells within each 100ms time window is consistent across simulations.

Synaptic weights between each pair of (current and goal location) distance cell arrays and two read-out neurons for each principal axis $\vec{x}$ or $\vec{y}$ increase linearly in opposing directions along topographically ordered distance cells. Hence, the relative firing rate of these two read-out neurons encodes the displacement between current and goal locations on that axis. The magnitude of translation vectors on each principal axis is decoded by fitting a line of regression to a plot of the difference in the firing rate of read-out cells against the true displacement vector along that axis across all *N*=1000 iterations.

The distance cell simulations are run using the 'DistanceCellModel' function.

Inputs to this function are 'GC_cpm', the number of grid cells used to encode each phase offset in each module (GC_cpm=20 in all simulations presented in the manuscript); and 'DrawFig', a toggle used to dictate whether the output is plotted to replicate Figure S2 in the manuscript.

Outputs of the function are 'Starts', an array of randomly assigned 2D start locations, in metres; 'Goals', an array of randomly assigned 2D goal locations, in metres; 'Decoded', an array of decoded 2D translation vectors, in metres (i.e. Starts + Decoded = Goals); 'Vec_l', an array of true vector lengths, in metres; 'RateDiff', an array of the difference in read-out firing rates on each axis, in Hz; and 'Error', an array of distance errors in the decoded translation vectors, in metres.

Hence, the following syntax is used to replicate Figure S2 of the manuscript (shown below):

```
[Starts Goals Decoded Vec_l RateDiff Error] = DistanceCellModel(20, 1);
```

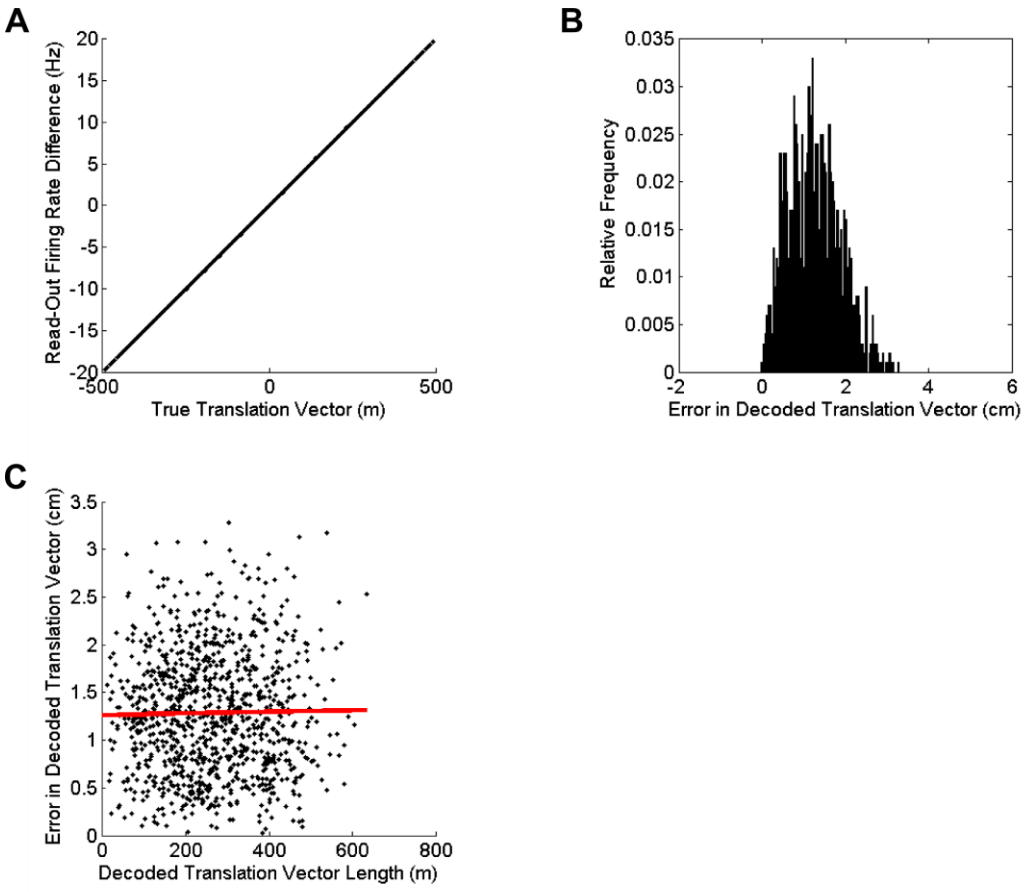

Figure S2: Simulations of the Distance Cell Model. We postulate two arrays of N_{DC}=12500 distance cells that encode start and goal locations, respectively, on each of two axes $\vec{x}$ and $\vec{y}$. Grid cells in each module project to distance cells with synaptic weights that are proportional to their mean firing rate at the corresponding location on that axis. The number of spikes fired by N_{GC}=400 grid cells, representing m_i=20 equally distributed spatial phases p_i along each axis, is given by a Poisson process with a rate function that is cosine tuned for location along that axis and a time window of 100ms. Distance cells in each array are subject to winner-take-all dynamics, and all distance cells in each array project to a pair of read-out cells with synaptic weights that increase topographically in different directions along each axis. Hence, the relative firing rate of those read-out cells encodes the displacement between start and goal locations along that axis. **(A)** The difference in read-out cell firing rates (collapsed across both principal axes) plotted against the true translation vector for *N*=1000 simulations with randomly assigned start and goal locations in a 500m sided 2D arena. This illustrates that translation vectors can be accurately decoded from the relative firing rate of read-out cells. **(B)** The distribution of errors in 2D translation vector lengths decoded from the relative firing rate of read-out cells across the same *N*=1000 simulations. Note that distance cells were given a resolution of 4cm in these simulations. **(C)** Total decoded 2D translation vector lengths plotted against decoding error. This demonstrates that there is no correlation (r=0.017, p=0.60) between the length of translation vectors and the decoding error, as the spatial resolution of distance cells is even across the entire 2D arena.

The Rate-coded Vector Cell Model

In vector cell simulations, separate populations of vector cells decode the displacement between arbitrary start and goal locations in each direction along the principal axes $\vec{x}$ and $\vec{y}$. The spatial resolution of vector cells is pseudo-exponentially distributed in the 0 : 500m range to give a total of N_{VC}=1250 vector cells in each of four arrays (corresponding to each direction of movement along the two principal axes $\vec{x}$ and $\vec{y}$), or a total of 5000 vector cells overall.

In these simulations, for each pair of *N*=1000 arbitrary start and goal locations, the current location is updated by 80% of the decoded vector length along each axis in each step, and then translation vectors are iteratively re-calculated, until the current location is within 1m of the true goal. In each step, vector cells within each pair of arrays encoding displacements along a single axis are subject to winner-take-all dynamics implemented with an E%-max algorithm, such that cells only fire if their input is within *k*=1% of the maximum feed-forward excitation. The overall translation vector is decoded by combining the average of all active vector cells along each axis $\vec{x}$ or $\vec{y}$, weighted by their firing rate.

Each pair of vector cell arrays that encode displacement in either direction along the principal axis $\vec{x}$ or $\vec{y}$ receive input from a population of grid cells that encodes start and goal locations on that axis. Vector cells receive input from grid cell pairs within each module whose unwrapped phase difference corresponds to that absolute vector on that directional axis through multiplicative synapses, such that activity only reaches the vector cell if both grid cells in a start-goal pair are active. The number of spikes fired by each grid cell in each step is dictated by a Poisson process with a cosine tuned rate function and a time window of 100ms.

The rate-coded vector cell simulations are run using the 'VectorCellModel' function.

Inputs to this function are 'GC_cpm', the number of grid cells used to encode each phase offset in each module (GC_cpm=20 in all simulations presented in the manuscript); and 'DrawFig', a toggle used to dictate whether the output is plotted to replicate Figure S3 in the manuscript.

Outputs of the function are 'Starts', an array of randomly assigned 2D start locations, in metres; 'Goals', an array of randomly assigned 2D goal locations, in metres; 'Decoded', an array of decoded 2D translation vectors, in metres (i.e. Starts + Decoded = Goals); 'Vec_l', an array of true vector lengths, in metres; 'FirstError', an array of distance errors in the translation vector decoded in the first iterative step, in metres; 'LastError', an array of distance errors in the translation vector decoded in the final iterative step, in metres; and 'Steps', an array of the number of iterative steps needed to decode the final translation vector.

Hence, the following syntax is used to replicate Figure S3 of the manuscript (shown below):

```
[Starts Goals Decoded Vec_l FirstError LastError Steps] = VectorCellModel(20, 1);
```

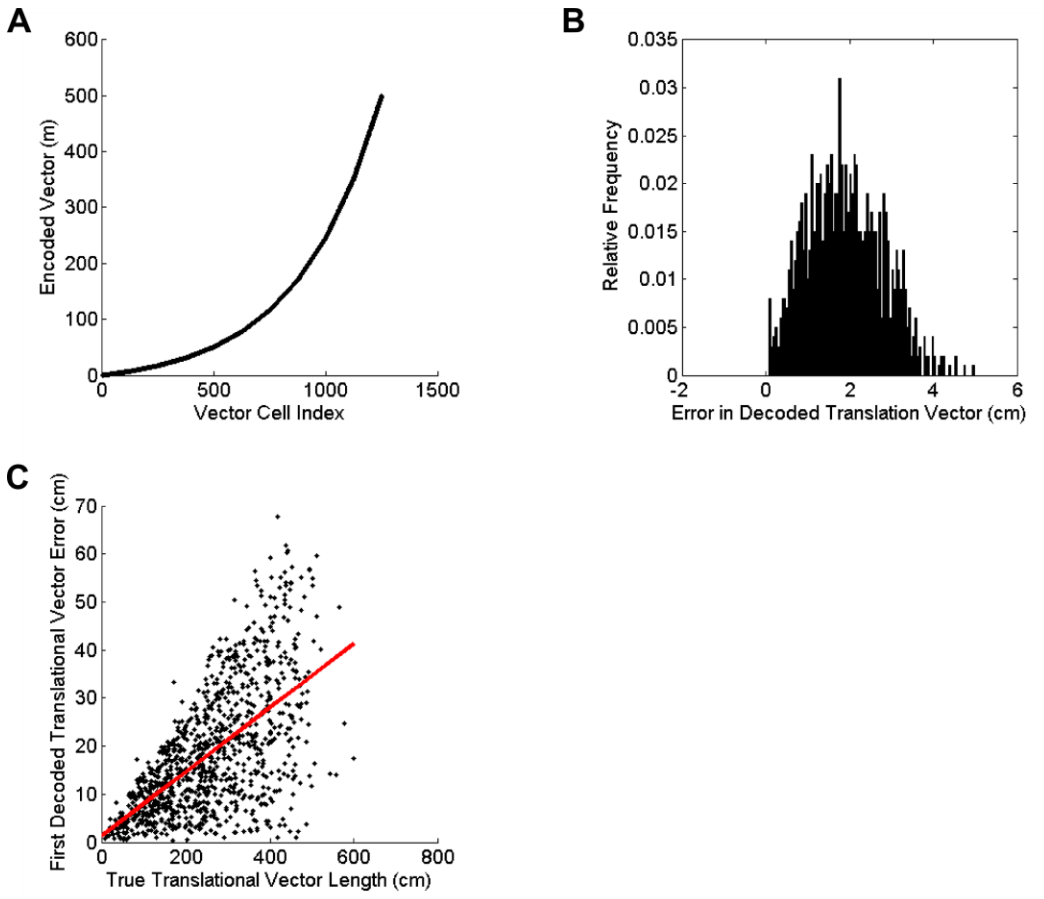

Figure S3: Simulations of the Rate-coded Vector Cell Model. We postulate two arrays of N_{VC}=1250 vector cells that encode the displacement between start and goal locations in either direction along each of two axes $\vec{x}$ and $\vec{y}$. Vector cells receive input from grid cell pairs within each module whose unwrapped phase difference corresponds to that absolute vector on that axis. The number of spikes fired by N_{GC}=400 grid cells, representing m_i=20 equally distributed spatial phases p_i along each axis, is given by a Poisson process with a rate function that is cosine tuned for location along that axis and a time window of 100ms. Vector cells in each array are subject to winner-take-all dynamics, and translation vectors are directly decoded from the weighted mean of vector cell activity. Translation vectors are calculated iteratively, with the start location updated by 80% of the total decoded vector length along each axis prior to each new calculation, until the start location is within 1m of the true goal. **(A)** Pseudo-exponential distribution of translation vectors encoded by vector cells. **(B)** The distribution of translation vector errors decoded in the final iterative step from the activity of vector cells across *N*=1000 simulations with randomly assigned start and goal locations in a 500m sided 2D arena. **(C)** Total decoded 2D translation vector lengths plotted against decoding error in the first iterative step. This demonstrates that there is a significant correlation (r=0.61, p<0.001) between the length of translation vectors and initial decoding error, as the spatial resolution of vector cells decreases with increasing encoded displacement.

The Phase-coded Vector Cell Model

In phase-coded vector cell simulations, pairs of vector cell arrays that encode displacement in either direction along the principal axis $\vec{x}$ or $\vec{y}$ receive input from separate populations of grid cells that exhibit phase precession aligned with each axis. Only the grid cells within each module GC_b that exhibit the maximum firing rate at the goal location *b* fire within a single t_{theta}=100ms theta cycle. Their theta firing phase is drawn from a circular normal distribution with a mean $\bar{\emptyset}_\iota$ that is linearly correlated with the circular distance between the peak of that grid cell's firing field *b* and the current location *a* along that axis, and a circular standard deviation of $\mu = {}^{\pi}/_{6}$ rad.

In these simulations, vector cells must be sensitive to specific spike timing input patterns, analogous to models of polychronous computation. For simplicity, this is achieved by grid cells in each module projecting to vector cells through delay lines with transmission delays t_i that are proportional to the difference between the grid scale s_i and the encoded vector *d*. The activity of each vector cell is then taken as the resultant vector length of input spike phases from grid cells encoding the goal location in all modules.

The phase-coded vector cell simulations are run using the 'PhaseModel' function.

Inputs to this function are 'PNoise', the standard deviation μ of phase noise applied to the theta firing phase of each grid cell (see above); 'GC_cpm', the number of grid cells used to encode each phase offset in each module (GC_cpm=20 in all simulations presented in the manuscript); and 'DrawFig', a toggle used to dictate whether the output is plotted to replicate Figure S4 in the manuscript.

Outputs of the function are 'Starts', an array of randomly assigned 2D start locations, in metres; 'Goals', an array of randomly assigned 2D goal locations, in metres; 'Decoded', an array of decoded 2D translation vectors, in metres (i.e. Starts + Decoded = Goals); 'Vec_l', an array of true vector lengths, in metres; 'FirstError', an array of distance errors in the translation vector decoded in the first iterative step, in metres; 'LastError', an array of distance errors in the translation vector decoded in the final iterative step, in metres; and 'Steps', an array of the number of iterative steps needed to decode the final translation vector.

Hence, the following syntax is used to replicate Figure S4 of the manuscript (shown below):

```
[Starts Goals Decoded Vec_l FirstError LastError Steps] = PhaseModel(pi/6, 20, 1);
```

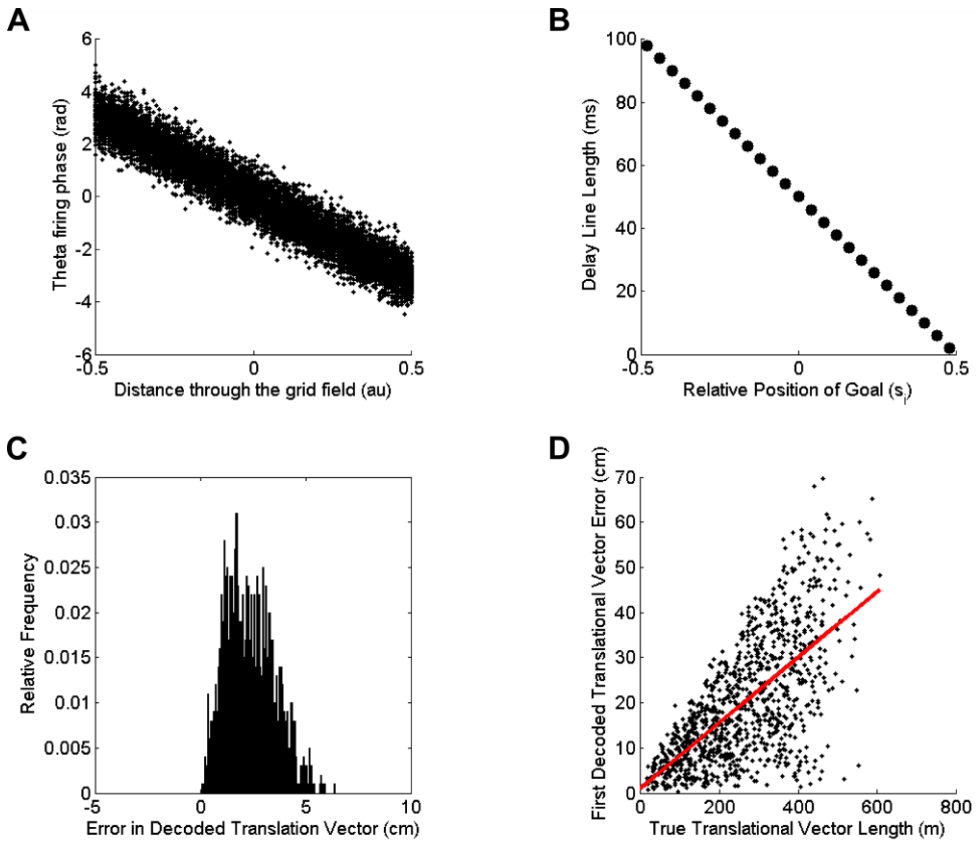

Figure S4: Simulations of the Phase-coded Vector Cell Model. We postulate two populations of grid cells that exhibit phase precession aligned with specific non-collinear axes; and that only the grid cells within each module which exhibited the maximum rate at the goal location are reactivated, firing one spike within a single t_{theta}=100ms at a theta phase consistent with the current location. **(A)** Theta firing phases are drawn from a circular normal distribution with a mean that is linearly correlated with the distance (modulo grid scale) between the peak of that grid cell's firing field and the current location along the axis of phase precession and a circular standard deviation of $\mu = {}^{\pi}/_{6}$ rad. **(B)** Grid cells project to vector cells through delay lines with transmission delays which ensure that spikes from grid cells encoding the goal location across modules arrive simultaneously at the corresponding vector cell. The activity of vector cells is determined by the temporal proximity of incoming spikes, i.e. vector cells perform temporal coincidence detection on the inputs that arrive from grid cells through delay lines. Vector cells in each array are subject to winner-take-all dynamics, and translation vectors can be directly decoded from the weighted mean of vector cell activity. Translation vectors are calculated iteratively, with the start location updated by 80% of the total decoded vector length along each axis prior to each new calculation, until the start location is within 1m of the true goal. **(C)** The distribution of translation vector errors decoded in the final iterative step from the activity of vector cells across *N*=1000 simulations with randomly assigned start and goal locations in a 500m sided 2D arena. **(D)** Total decoded 2D vector lengths plotted against decoding error in the first iterative step. This demonstrates that there is a significant correlation ($r=0.66$, $p<0.001$) between the length of translation vectors and initial decoding error, as the spatial resolution of vector cells decreases with increasing encoded displacement.

The Linear Look-ahead Model

In the linear look ahead model, four linear look ahead events are required to compute translation vectors between arbitrary start and goal locations in 2D space, corresponding to the sequential exploration of each direction along the two principal axes $\vec{x}$ and $\vec{y}$. During each linear look ahead event, activity is initiated by setting grid cell firing rates in each module to match those at the start location and then updating grid cell firing in each subsequent time step according to simulated movement away from the current location at a constant speed along that directional axis. The distance between start and goal locations in either direction along each principal axis is decoded from the time elapsed between the initiation of linear look ahead activity and firing activity in the place cell encoding the goal location.

In these simulations, we set a time step *dt*=5ms that corresponds to the integration time constant of a post-synaptic neuron – that is, grid cells across modules that fire within a 5ms time window encode a single location that can be decoded by an output place cell. We then set a constant, virtual speed for linear look ahead events of v_{sweep}=8ms^{-1}, which subsequently determines the distance increment between each integration time step and therefore the spatial resolution of output place cells (0.04m in this case). Synaptic weights between grid cells and place cells are proportional to the grid cell firing rate at the peak of the place cell firing field along the axis $\vec{x}$ or $\vec{y}$, i.e. $w_{DC,GC} \propto r_{j,i}(a)$. The number of spikes fired by each grid cell in each time step is dictated by a Poisson process with a cosine tuned rate function and the time window of *dt*=5ms. Activity within each place cell array is subject to winner-take-all dynamics implemented with an E%-max algorithm such that cells only fire if their input is within *k*=1% of the maximum feed-forward excitation.

The linear look-ahead cell simulations are run using the 'LinearLookAheadModel' function.

Inputs to this function are 'GC_cpm', the number of grid cells used to encode each phase offset in each module (GC_cpm=20 in all simulations presented in the manuscript); and 'DrawFig', a toggle used to dictate whether the output is plotted to replicate Figure S5 in the manuscript.

Outputs of the function are 'Starts', an array of randomly assigned 2D start locations, in metres; 'Goals', an array of randomly assigned 2D goal locations, in metres; 'Decoded', an array of decoded 2D translation vectors, in metres (i.e. Starts + Decoded = Goals); 'Vec_l', an array of true vector lengths, in metres; 'Error', an array of distance errors in the decoded translation vectors, in metres; and 'LLA_Time', an array of times taken to complete each linear look-ahead event, in seconds.

Hence, the following syntax is used to replicate Figure S5 of the manuscript (shown below):

```
[Starts Goals Decoded Vec_l Error LLA_Time] = LinearLookAheadModel(20, 1);
```

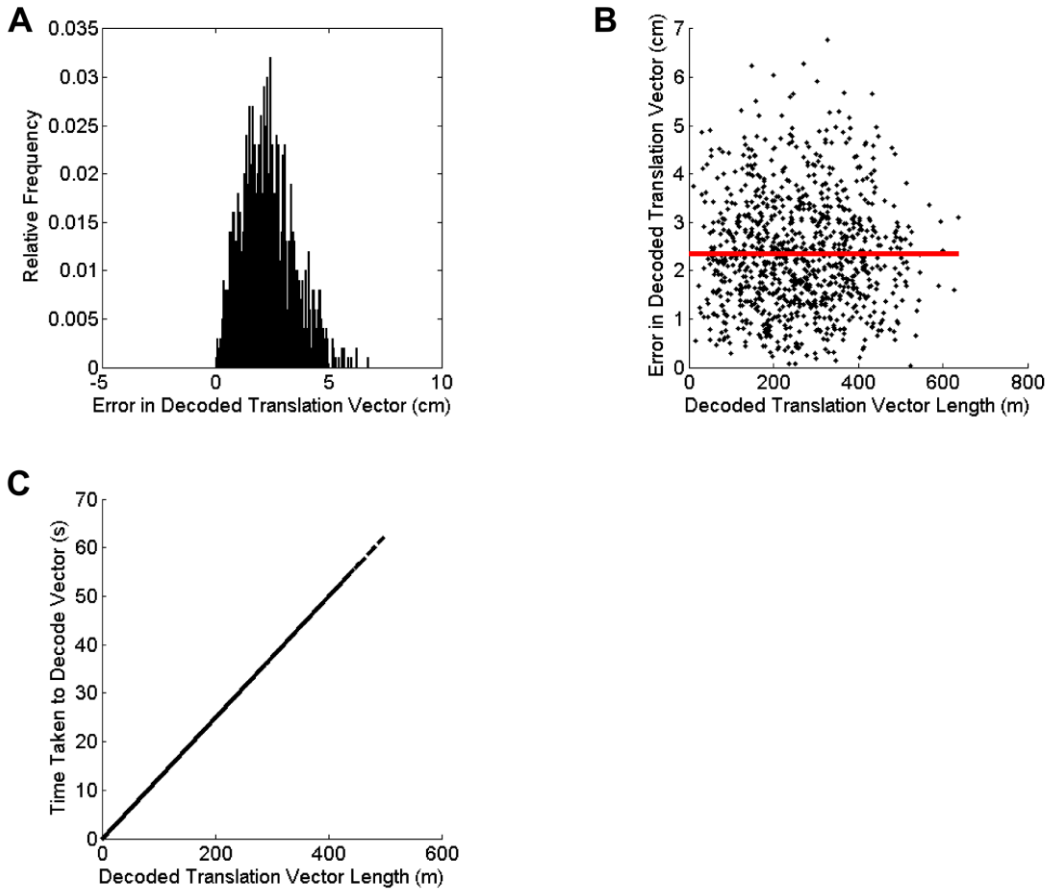

Figure S5: Simulations of the Linear Look-ahead Model. Linear look ahead activity is initiated by setting grid cell firing rates in each module to match those at the start location and then updating those firing rates in each subsequent 5ms time step according to simulated movement away from the current location at a constant speed of $8ms^{-1}$ along a specific directional axis. Grid cells in each module project to an array of place cells that evenly cover the 2D arena with synaptic weights that are proportional to their mean firing rate at the centre of the place field, and place cells are subject to winner-take-all dynamics. The time taken for grid cells encoding the goal location across modules to become simultaneously active, or the firing rate of a neuron that integrates total activity in the grid cell network during linear look ahead, subsequently encodes the displacement between start and goal locations along that axis. This is indicated by activity in the corresponding place cell. The number of spikes fired by N_{GC}=400 grid cells, representing m_i=20 equally distributed spatial phases p_i along each axis, is given by a Poisson process with a rate function that is cosine tuned for location along that axis. Overall translation vectors in 2D are constructed from linear look ahead in each direction along two non-collinear axes. **(A)** The distribution of decoding errors across N=1000 simulations with randomly assigned start and goal locations in a 500m sided 2D arena. **(B)** Total decoded 2D translation vector lengths plotted against decoding error. This demonstrates that there is no correlation (r=0.017, p=0.60) between the length of translation vectors and the decoding error, as place cell firing fields evenly cover the entire 2D arena. **(C)** The time taken to decode translation vectors along each axis (i.e. the length of each linear look ahead event) is determined by the displacement between start and goal locations along that axis.