Author's accepted manuscript. The published paper is available at https://doi.org/10.1109/IJCNN.2017.7966406

Long-Range Navigation by Path Integration and Decoding of Grid Cells in a Neural Network

Vegard Edvardsen
Department of Computer Science
NTNU – Norwegian University of Science and Technology
Trondheim, Norway
Email: vegarded@ntnu.no

***Abstract*—Neural modelers in the domain of robot navigation, e.g. within the fields of neurorobotics and neuromorphic engineering, can benefit from a wealth of inspiration from neuroscientific research in the hippocampal formation—cell types such as place cells and grid cells provide a window into the inner workings of high-level cognitive processing, and have spawned many interesting computational models. Grid cells are thought to participate in path integration and to implement a general coordinate system, both of which are useful features in a neural navigation model. Continuous attractor networks are a computational model that can embody both aspects of grid cells, and in previous work we showed that a neural network can successfully decode the outputs of such networks in order to implement vector navigation. That work assumes that the grid cell system represents long distances by employing a geometric progression in its spatial scaling of successive submodules, in such a way that "nested" grid cell decoding can be performed. For long-range navigation this requires that the continuous attractor networks can implement sufficiently long geometric progressions of grid scales, but this turns out to trigger the issue of "pinning". In this paper we demonstrate conditions under which pinning occurs as well as its consequences for the grid cell-based navigation model. We propose and assess several candidate solutions to the problem, in particular based on differential adjustment of neurons' update rates in the model. We finally demonstrate that the system is able to perform long-range navigation using our chosen solution.**

I. Introduction

Neural networks have in recent years had a renaissance as a tool for building artificially intelligent computer systems [1]. Improved hardware, datasets and techniques for construction and training of deep neural networks have yielded systems that have advanced the state-of-the-art in areas of such wide variety as image recognition [2] and beating human players in the board game Go [3]. Neural networks for such tasks are often feed-forward in architecture, especially in sensory processing applications [4]. However, the need for a short-term memory capacity has also been emphasized, particularly for problems demanding higher-level cognitive processing [5].

One domain of cognitive tasks that usually requires a short-term memory is that of navigation—keeping track of where you come from and where you are going; planning how to get to a goal location and thereafter back home. In earlier work [6] we showed how an artificial neural network inspired by the brain's grid cell system can keep track of an agent's current 2D coordinates and use this information to guide the agent to a goal location. In this paper we will build upon that work to improve our grid cell-based neural navigation system to work over longer distances.

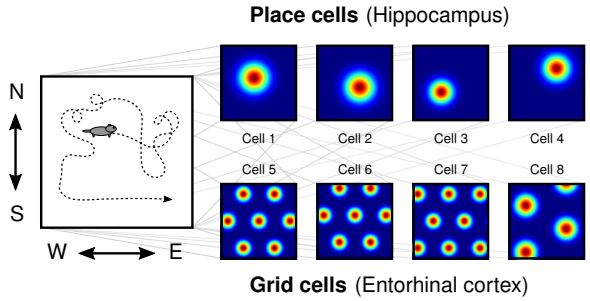

Fig. 1. Illustration of two types of spatial neurons found in the hippocampal formation of rodents such as rats—place cells and grid cells. Left: Neurons are recorded as the animal explores an enclosure. The animal's position in the horizontal plane is recorded simultaneously. Right: By plotting heatmaps for each neuron—showing the neuron's average activation in each visited position bin—the characteristic spatial responses of place cells and grid cells are revealed. Place cells respond typically in one or a few areas of the enclosure, whereas grid cells respond at the vertices of an infinite hexagonal grid pattern.

Section II presents pertinent background material on grid cells and a computational model for them. In Section III we demonstrate the range of our navigation system when using a single module of grid cells. Section IV discusses ways in which the range of the grid cell system can be extended by introducing multiple modules. The navigation system is then tested with increasing numbers of modules and is shown to experience a shortfall in range improvement after a certain number of modules have been added. Section V demonstrates the issue of "pinning" that is responsible for this shortfall. In Section VI we propose and assess several candidate solutions to the pinning problem, with our chosen solution evaluated in Section VII. Section VIII concludes the paper.

II. Background

A. Spatial neurons in the brain

Neuroscientific studies of navigation are often concentrated on the mammalian brain region around the hippocampus, primarily in rats and other rodents. This brain region is believed to implement a *cognitive map*—a neural implementation of high-level cognitive information about the spatial environment. Cell types in this region, such as place cells [7], head-direction cells [8], border cells [9], and grid cells [10], provide a wealth

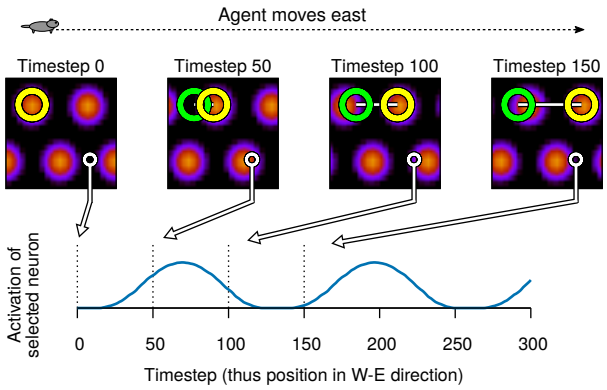

Fig. 2. Illustration of key concepts behind continuous attractor networks, which constitute a major class of computational models for grid cells. Each neuron is assigned a row and a column in a "neural sheet". Neurons are fully connected recurrently, with connection strengths pre-wired in such a way that bumps of activity will spontaneously emerge in the neural sheet and distribute in a hexagonal pattern. This hexagonal pattern becomes visible in the readouts of individual neurons across space because the pattern in the neural sheet is made to shift in proportion to the agent's movements in 2D space. The square heatmaps show instantaneous snapshots of the activation levels of the neurons in the neural sheet of a CAN-based grid module, each pixel indicating the activation level of one neuron. These snapshots are shown at four different timesteps, all while the agent is moving eastward. In response to the velocity input, the pattern shifts rightward in the neural sheet (indicated by the yellow circle, which tracks the motion of one of the bumps in the pattern). By reading out the activation level of a single neuron and plotting it over time, the grid pattern is revealed and the neuron therefore acts as a grid cell.

of inspiration for modelers of artificial neural navigation systems. In the context of systems that benefit from a short-term memory capacity, grid cells are particularly interesting. Like place cells and border cells, grid cells are neurons that activate depending on the animal's location in space. However, whereas place cells usually activate at only a particular place in an environment and border cells activate only along particular borders, a given grid cell is active whenever the animal is located at the vertices of an imaginary hexagonal grid extending throughout the 2D plane (Fig. 1). This relationship between the cell's activity and the animal's location in 2D space persists even in complete darkness, indicating that grid cells reflect an internally maintained neural activity and that this activity can be generically updated by self-motion inputs. This suggests that grid cells participate in a path integration process in the brain, continuously adding the animal's current velocity vector to an internal variable representing a vector of the total displacement from a point of reference (Fig. 3).

B. Continuous Attractor Networks (CANs)

"Continuous Attractor Networks" (CANs) are recurrent neural networks that are wired in a particular way so that the energy landscape of the network contains a continuum of stable network states of a particular dimensionality [11]. One-dimensional CANs are for example used as a model for head-direction cells—the stable states of the network each represent a particular head-direction, i.e. the stable states fall along a 1D line, and inputs signaling head-turns cause the network to shift to new network states that reflect the updated head-direction.

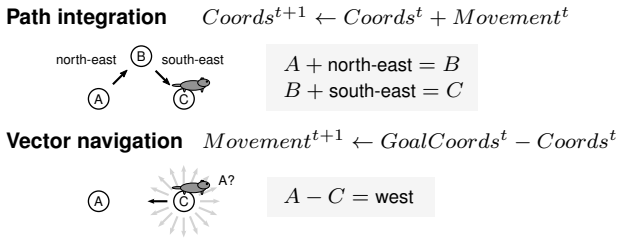

Fig. 3. Path integration maintains an internal estimate of the agent's current coordinates based on its movement history. Vector navigation calculates the direction to a goal location based on the current coordinates and the goal coordinates.

These CANs thus perform path integration on the 1D head-direction variable—by extending a CAN to two dimensions, it will be able to use inputs representing velocity in the 2D plane to update a network state representing the current 2D total displacement (Fig. 2). It is possible to create 2D CANs whose individual neurons produce the repeating, hexagonal grid patterns across space that are characteristic of grid cells, and CANs have indeed emerged as one of the major classes of computational models for grid cells [12].

C. Vector navigation with grid cells

The robot navigation system RatSLAM [13] is an example of the potency of looking to neuroscientific findings in the hippocampus for neural principles to use in an artificial navigation system. The core of this algorithm is a CAN that performs path integration and participates in place recognition. A CAN such as used by RatSLAM will, within a certain range, generate unique neural activity patterns for each distinct location, and reactivate these patterns whenever a "loop is closed" during revisits to old locations. The outputs of the CAN can therefore be used to generate novel "labels" for new locations and then reactivate those labels when an old place is visited anew [14], therefore helping the system downstream of the CAN to perform place learning and place recognition.

However, using the neural representation of grid cells, another function also presents itself. Not only is there a systematic way in which grid cell activity can be generated by a path integration process, but the reverse can also be said: Grid cell activity from location A and location B can be compared in order to extract the direction of movement between them. The pieces are therefore in place for grid cells to be used as a 2D coordinate system in artificial neural navigation systems; coordinates can be updated by adding new velocity vectors to them, and movement directions can be recovered by comparing/"subtracting" coordinates (Fig. 3). Bush et al. [15] provide several possible neural networks for decoding grid cells to movement vectors, and in earlier work [6] we showed a neural system where both path integration and grid cell decoding is integrated in the same agent controller. There are yet, however, many possibilities and questions left to explore in this area, one of which—the possibility of using grid cells for navigation over long distances—is the topic of this paper.

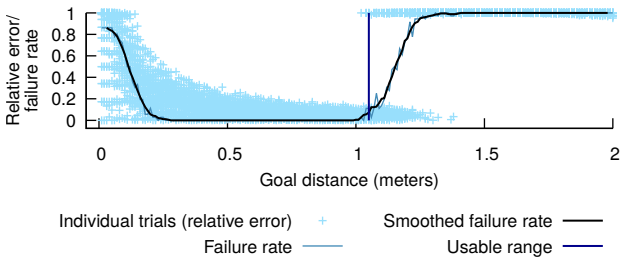

Fig. 4. Navigating with a single grid module. The agent was tested for "goal distances" in 1 cm increments from 1 cm to 200 cm, conducting 36 trials in 10 degree increments for each distance. A trial consisted of the agent driving in the specified direction, halting once the distance to the origin was within 10^{-6} m of the "goal distance", and then being allowed twice the number of outbound timesteps to try to navigate back to the origin. The closest approach to the origin was divided by the initial halting distance to obtain a "relative error" for the trial. Trials with relative errors of 0.5 or more were considered failures, and the failure rate was calculated for each distance bin. This rate was then smoothed by averaging it in a sliding window of 5 bins. The "usable range" was determined as the last distance bin before the smoothed failure rate reached a threshold of 0.1 (skipping the initial region of failed trials).

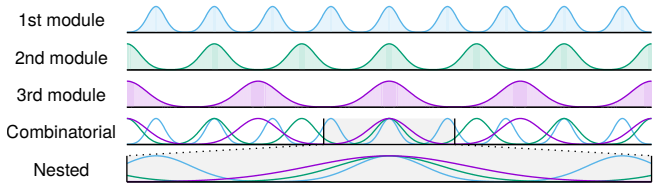

Fig. 5. Conceptual comparison of two different views on how the ambiguity from a single grid module can be resolved—and the usable range thus be increased—by utilizing multiple grid modules of increasing scale. Depending on the particular grid scales used, the grid modules can continue to generate unique activity combinations over distances exceeding the range of the largest-scaled module in a "combinatorial" fashion. A different view is to assume that there is a hierarchy of grid modules such that the largest module is sufficiently large to cover the entire behavioral range of the animal and to guide the animal into the usable range of the smaller-scaled modules. The smaller-scaled modules are thus "nested" within the larger-scaled ones, their purpose being to increase precision beyond what the larger-scaled modules might provide. Grid scales observed in neuroscientific experiments are known to follow a geometric progression, so both of these views remain viable options.

III. Navigating with a single grid module

One of the defining characteristics of a grid cell is that the neuron's activity pattern repeats across space, so it is not possible to unambiguously determine the animal's location by making a read-out of one grid cell's current activation level. The simultaneous activation levels of other grid cells must therefore also be considered. Biological grid cells are known to cluster into *grid modules*, where all neurons in a grid module share the same hexagonal grid pattern across space (identical scaling and orientation of the pattern), except for a shift/offset in the pattern between neurons. Cells 5–7 in Fig. 1 might thus belong to the same grid module. The read-out of an entire grid module will have the same ambiguity due to repeating grid patterns as an individual grid cell—after traveling a distance sufficient to make one grid cell start to repeat itself, then all the other grid cells in the module will also have started repeating. Grid cells organized in a module make it possible to determine the animal's location within the boundaries of the module's "unit tile", but it does not reveal any information about where this unit tile might be located in global space.

We sought to demonstrate the limits of navigating with only one grid module in our current grid cell-based navigation system. The system performs path integration using a configurable number of CAN-based grid modules as the simulated agent drives away from the origin location, and then later tries to find its way back to the origin by decoding the information residing in the recurrent short-term memory of the grid modules. The version of our navigation model used in this paper is in most respects similar to how it was described in [6].

Fig. 4 shows the results from an experiment where the agent was tested at various goal distances between 1 cm and 200 cm. Using a criterion described in detail in the figure caption, we calculated the "usable range" to be 1.05 meters.

A navigational range on the order of one meter will clearly not be sufficient for a number of applications, and neither is this range behaviorally sufficient for animals such as rats and bats that forage across distances up to kilometers away from the nest [16]. However, the fact that there are multiple grid modules in the brain might provide a solution.

IV. Navigating with multiple grid modules

There are multiple grid modules in the brain, and there appears to be a constant scaling factor between the spatial grid scales of successive modules, so that the grid scales of a sequence of grid modules form a geometric progression [17]. This enables two distinct views of how the grid cell system might unambiguously represent 2D coordinates over longer distances (Fig. 5), which we will refer to respectively as the "combinatorial" and the "nested" approaches. The "combinatorial" approach emphasizes that the collective activity of a set of grid modules can remain unique over a total range far exceeding that of the largest grid module. This range can theoretically be as much as the least common multiple of all of the modules' grid scales [18], which increases quickly when adding just a few grid modules together—however, it has been argued that utilization of the full range requires precise readouts from each module [19], and the possibility has been raised that an error correction mechanism might be required during ongoing path integration operation [20]. The "nested" approach embraces the fact that the grid scales of successive modules appear to follow a geometric progression. As the number of grid modules increases, the grid scale of the largest module grows exponentially. In this view, one therefore assumes that there is a sufficient number of grid modules so that the largest grid scale is larger than the behavioral range of the animal, i.e., that there is a grid module large enough to by itself unambiguously indicate the animal's location in 2D space. While the resolution of a grid module might grow coarser as the grid scale increases—so that a very large-scaled grid module might only give a rough estimate of the animal's location—smaller-scaled grid modules could provide refinements to the initial estimate from the largest module [21].

It is the latter view we will consider in this paper; we showed in [6] that there exists a simple neural grid cell decod-

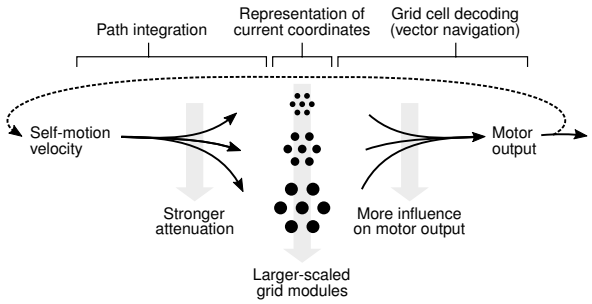

Fig. 6. Conceptual overview of the full navigation model considered in this paper. Self-motion velocity is used to update multiple CAN-based grid modules of increasing grid scales that follow a geometric progression. Larger grid scales are achieved by attenuating the input to the grid module, so that the module's produced grid pattern appears stretched across space. The collective activity of all grid modules represents a set of "coordinates" in the navigation model. Larger-scaled grid modules are given priority in the decoding process, in accordance with the view that smaller-scaled grid modules are "nested" within the larger ones [21].

ing mechanism for such setups. The decoder considers the grid cell signal in each grid module individually and then at the output end of the system gives priority to the largest-scale grid module outputs, reminiscent of the nested refinement described by [21]. Following this principle, in order to increase the range of the navigation system from ∼1 meter to e.g. 150 meters, we should repeatedly add extra grid modules to the system in a geometric progression until the projected range of the largest module is sufficiently large. Using a scale ratio of 1.5, as suggested by theoretical studies and within the range of scale ratios reported from neuroscientific experiments, we would need 14 grid modules in total ($1.05\ \mathrm{m} \cdot 1.5^{13-1} \approx 136\ \mathrm{m}$, $1.05\ \mathrm{m} \cdot 1.5^{14-1} \approx 204\ \mathrm{m}$).

Fig. 6 illustrates an overall schematic of the full grid cell-based navigation system with all of these pieces in place. A number of CAN-based grid modules, following a geometric progression of grid scales, receive self-motion velocity in order to perform path integration. The output from these modules is decoded in a nested fashion to perform vector navigation. The illustration also alludes to the specific mechanism by which the different grid scales are attained in the otherwise identical CAN modules—by increasingly *attenuating* the velocity input to each module. We will return to this topic in later sections.

Fig. 7 shows the results of an experiment that follows the paradigm in Fig. 4, but where much longer distances are tested and where separate sets of trials were performed with the model configured to use 2, 4, 6, 8, 10, 12 and 14 grid modules. The set of trial distances was selected to be equally spaced on a logarithmic scale, because we are interested in observing behaviors of the system that are expected to follow an exponential development. Usable ranges were determined as in Fig. 4, and aggregated results are shown in Fig. 8.

The system behaves as expected for low numbers of modules—as extra modules are added, the usable range increases exponentially. However, at 12 and 14 modules there are clear deviations from this trend. The usable range appears to level off around the level attained with 10 modules. The

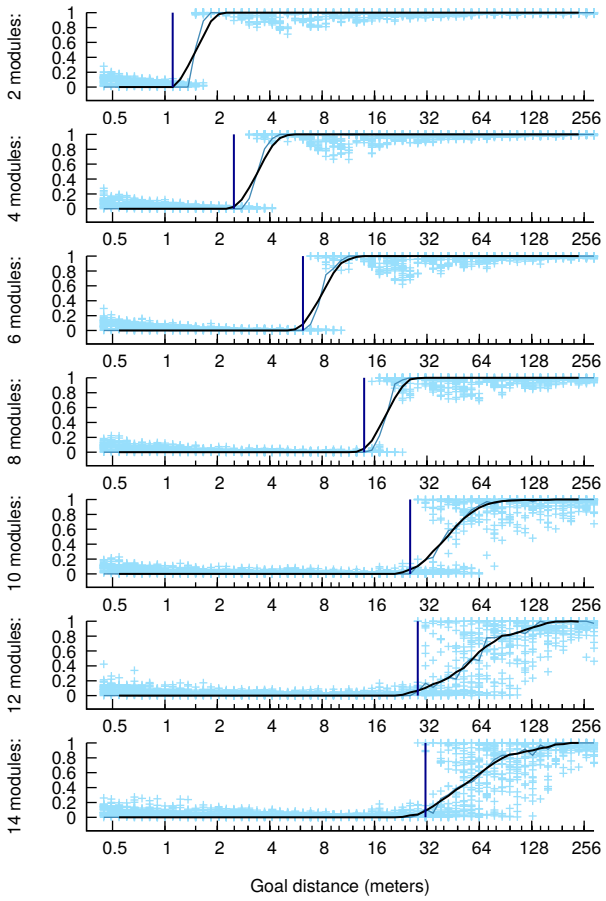

Fig. 7. Navigating with multiple grid modules. The trials and analyses were conducted as in Section III, but with either 2, 4, 6, 8, 10, 12 or 14 modules, and with 65 tested distances selected to be equally spaced on a logarithmic scale from $1.5^{-2} \approx 0.444$ m to $1.5^{14} \approx 292$ m. Legend as in Fig. 4.

extra grid modules added from 10 to 12 and 14 do not seem to have contributed appreciably to the usable range of the system. In the remainder of this paper we will look into the cause of this problem and discuss possible solutions.

V. "Pinned" pattern flows in CANs

The source of the shortcoming is demonstrated in Fig. 9. In this experiment, the flow of the activity pattern in a CAN is measured for different strengths of net input velocity. The relevance to the grid scaling problem is that this is currently how the CANs' grid scales are increased from the baseline grid scale—by increasingly attenuating the velocity inputs to the modules that should have larger grid scales.

The average flow reported for the unattenuated case in Fig. 9 was ∼981 neurons per 100 meters. In order to produce a grid module with e.g. twice the grid scale, the velocity input to that module should be half of the original signal—the module would then use twice the amount of time/distance to produce the same output as the unattenuated module, in effect producing a grid pattern stretched to twice the spatial scale. In general, to scale the grid pattern by a factor s, we attenuate the velocity input by $a = s$ by dividing it by a.

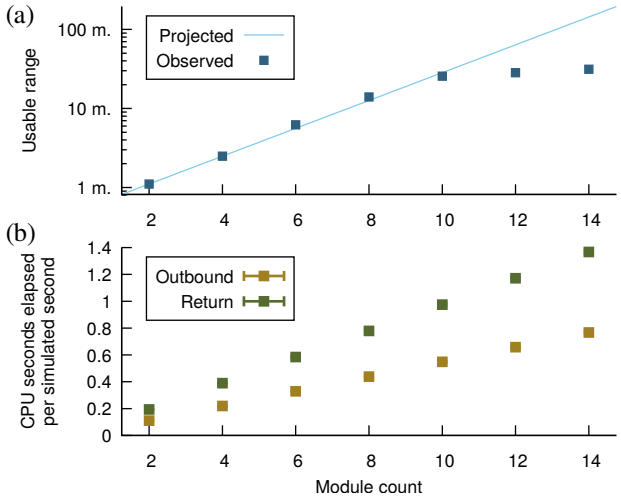

Fig. 8. Various results from the trials shown in Fig. 7, aggregated by the number of modules. (a) Usable range for each number of modules, as determined in Fig. 7. Also shows the expected range as a function of number of modules, projected from the range value found for 2 modules and using a factor of 1.5 per module. (b) Mean $\pm$ std.dev. across trials, of CPU time spent per simulated time. 1 simulated second corresponds to travelling 0.2 meters. "Outbound" is the initial excursion away from the starting location, and "return" is the following navigation phase where a return back to the starting location is attempted.

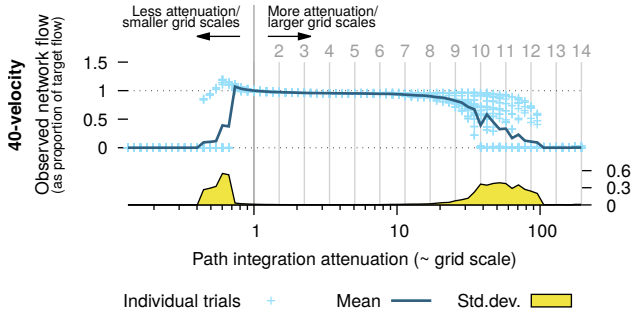

Fig. 9. Tracking the flow of the network pattern in a CAN-based grid module, across a range of different attenuation levels for the velocity inputs to the CAN. The attenuation is 1 for the smallest-scaled, i.e. the first, grid module. To create larger grid scales, the velocity inputs are attenuated increasingly—the attenuation levels corresponding to the 14 grid modules used in Fig. 7 are marked by vertical lines. For each of 73 different attenuation levels, equally spaced on a logarithmic scale from 1.5^{-5} to 1.5^{13}, 36 trials were conducted in 10 degree increments. Each trial consisted of the agent making an outbound excursion for 600 meters in the specified direction, with the velocity inputs to the single grid module attenuated at the specified level, followed by a rest phase. The flow of the pattern in the neural sheet of the CAN was tracked by following the motion of one of the activity bumps in the sheet, and reported as the number of neurons the pattern shifted in the x and the y directions. The Euclidean distance of this pattern shift was considered the "observed network flow" of the trial. The "target flow", i.e. the amount of flow we would expect for a given attenuation level if the module behaved completely linearly, was calculated as the mean observed flow at $x = 1$ divided by the attenuation. The outcome of each trial was plotted as the observed flow divided by the target flow, and the goal is for these points to fall along $y = 1$. The mean across the 36 trials for each particular attenuation level is shown as a dark line, and the corresponding std.dev. is shown on a separate y axis for clarity.

For this strategy to be successful, the network response should always remain proportional to the net velocity input. Thus, with an attenuation of e.g. 100—in order to produce a grid scale of 100 times the normal scale—we want the network pattern to flow $981/100 = 9.81$ neurons per 100 meters, etc. Fig. 9 explores whether this is the case, by measuring the network flow at various attenuation levels between 1.5^{-5} and 1.5^{13}, equally spaced on a logarithmic axis.

The specific attenuation levels used for the 14 different modules in Section IV are indicated in the figure. At the attenuation levels observed by modules 1–8, the network behavior is indeed inversely proportional to the attenuation, so that the reported flow values are around 1 as a proportion of the "target flow". However, for stronger attenuations the behavior starts to break down. Between the attenuation levels corresponding to the 8th and the 13th modules, there is an increased spread in the observed flow values, and consequently the grid module is less reliable as a path integrator. At yet higher attenuation levels, the CAN barely seems affected by the input signal and is unable to perform any path integration at all. This phenomenon can account for the ineffectiveness seen in Figs. 7 and 8 of adding any extra modules beyond 10. The phenomenon corresponds to what Burak & Fiete refer to as "pinning" of the network pattern at low velocities [11].

VI. Avoiding pinning

With this limited range of attenuation levels that the CAN will accept, we will not be able to extend the navigational range of the system no matter how many extra modules we include. All of the large-scaled modules will experience pinning and therefore not provide any useful navigational signal for their respective scales. In order to extend our grid cell-based navigation model to longer ranges, we thus need to solve the problem of implementing CAN modules for large grid scales while avoiding pinning. The issue at hand is that the input signals to the network simply get too weak for the network dynamics to be able to respond, i.e. for the network pattern to reliably shift to adjacent states. As an example, the attenuated velocity signal that reaches the 14th module is only $1.5^{-13} \approx 0.005$, i.e. half a percent, in strength compared to what the first module receives. However, we have implemented all of the CAN modules using identical network dynamics. Implicitly we therefore require our CAN modules to respond proportionally for inputs across several orders of magnitude, which is quite a tall order. In this section we will consider a few options for solving this problem.

One class of candidate solutions is to expand the CAN to make it better equipped to respond to input levels from a wider range. In this paper we will consider the effects of increasing the size of the CAN's neural sheet, i.e. increasing the number of neurons that participate in the grid module. The idea is that the CAN might get a better response range the more neurons it contains—the weak input signal reaches a larger number of neurons, thus the chances might be higher that the collective response of the network will be able to overcome the pinning phenomenon. We will test the effects of increasing the neural sheet size from 40^2 neurons to 48^2, 56^2 and 64^2 neurons.

The other class of solutions we will consider is to adjust the

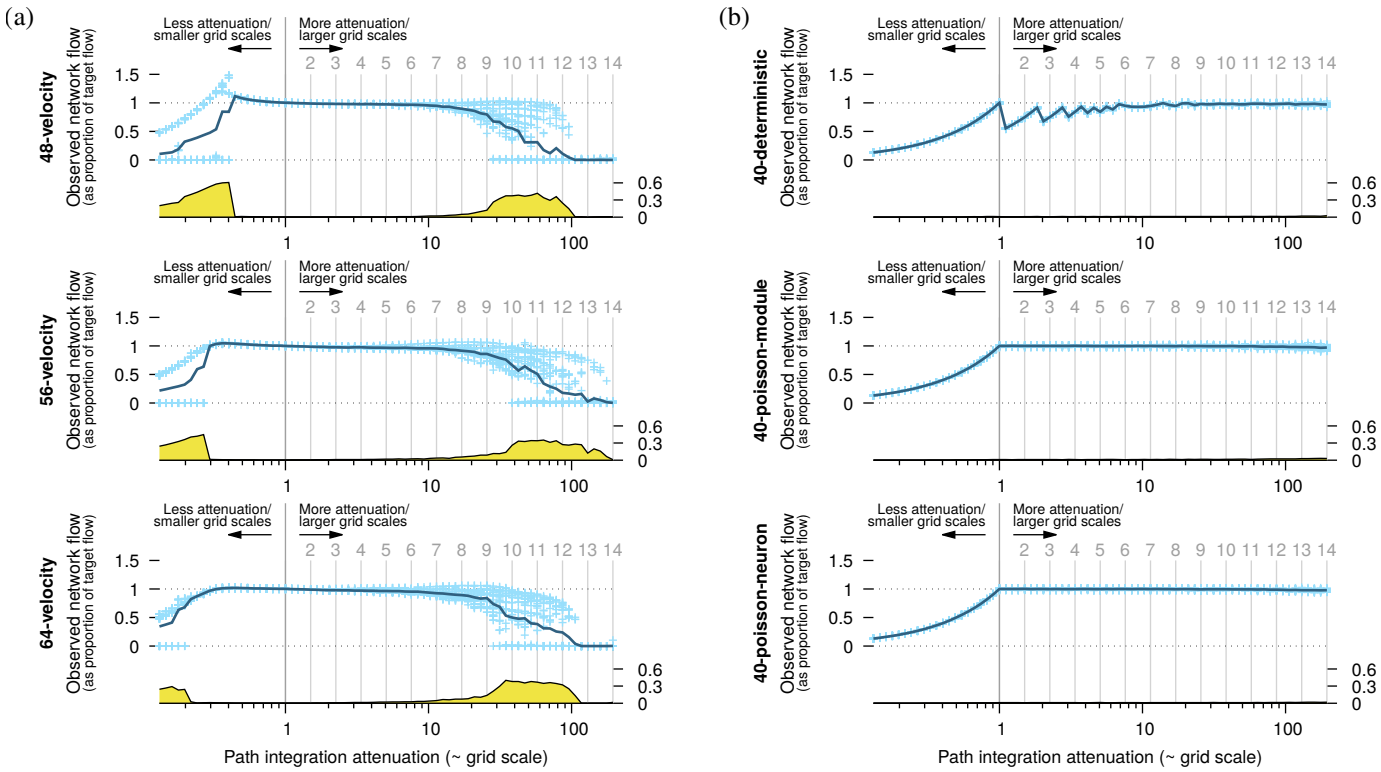

Fig. 10. Tracking the network flow under our various proposed solution schemes. Experiments were conducted as in Fig. 9, but with the model configured as indicated in bold to the left of each figure. "Target flow" is based on values in Fig. 9. (a) Trials with the number of neurons in the CAN module increased from 40^2 to 48^2, 56^2 or 64^2. (b) Trials with attenuation implemented through other means than by scaling the input velocity. Legend as in Fig. 9.

update rates of the grid cells as a function of the attenuation level—essentially to "subsample" the sequence of velocity values provided to the grid cells. Take our previous example of a module that shifts its network pattern 981 neurons per 100 meters, whose grid pattern we now want to scale up by a factor of 10; our current baseline approach is to achieve this grid scaling by dividing the input speed by 10. However, assuming that the network response correctly reflects the time integral of the input signal, an equivalent way of obtaining the same effect should be to use the original strength for the input signal, but to *only update the network a tenth as often*. From the perspective of the CAN module, the input history will reflect moving at full speed for 10 meters, rather than moving at 1/10th speed for 100 meters. However, because the agent will in fact have moved 100 meters spatially, the module's output should appear 10 times stretched across space, as intended.

There are several ways to go about implementing this basic idea. One is to update the entire module on fixed, repeating timestep intervals, and to skip updating the module at all other timesteps. We refer to this as the "deterministic" update mode. Two aspects of the deterministic update mode motivate a further development. First, this update mode is not able to generate all possible grid scales—there is e.g. no way to achieve an attenuation of 1.5 with this scheme (the skip interval necessarily has to be an integer, so the first possible skip amount beyond 0 is 1, skipping every other network update and resulting in an attenuation of 2). Second, with a deterministic update mode the system will be susceptible to failure in cases e.g. where the agent moves in periodic patterns that match the module's update rate. For example, if the agent follows a movement pattern that cycles every 10 timesteps, a module that updates precisely every 10th timestep will not be able to adequately sample the full trajectory of the agent.

To alleviate this, we introduce stochasticity into our update rule. Instead of calculating a discrete, deterministic update interval from the desired attenuation level, we calculate an update *probability* instead. At every timestep of the model, the entire module is updated by chance according to that update probability. Because this is a memoryless criterion—not relying on the model's timestep counter, as the deterministic mode does—we call this the "poisson-module" update mode. This scheme should solve our two reservations about the deterministic update mode. An attenuation of 1.5 would be achieved by updating the module with a probability of $1/1.5 \approx 0.67$ on every timestep, etc. Due to the probabilistic nature of the update rule, it should not be vulnerable to periodic fluctuations in the animal's velocity.

In both of these update modes, for strong attenuations, the module might go a large number of timesteps between each update and thus miss out on a substantial amount of velocity information. As our final proposed update scheme, we suggest to apply the probabilistic update rule *individually to each neuron* on every timestep, rather than to the module as a whole. Each neuron should over time experience the same update rate

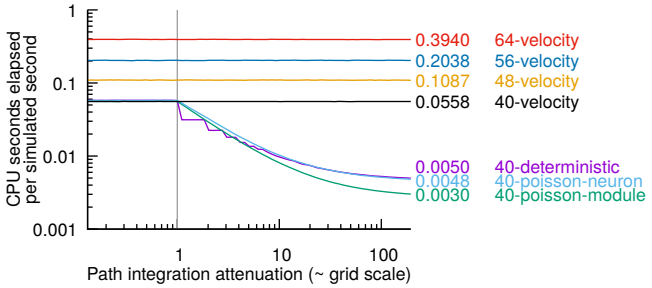

Fig. 11. CPU demand for the various schemes presented in Figs. 9 and 10. Mean CPU seconds per simulated seconds across trials for each attenuation level and model configuration. The labels to the right show the mean values at the rightmost data points.

as with the module-wide probabilistic update rule, but because it is applied individually to each neuron every timestep, there might potentially be an active subset of neurons in any given timestep. We term this the "poisson-neuron" update mode.

To evaluate the effect of these proposed solution schemes, Fig. 10 shows the results of performing the same experiment as in Section V for each of the following six configurations:

48-velocity, **56-velocity**, **64-velocity**:
Velocity-based attenuation, i.e. the original approach used in earlier sections of this paper, but with the number of neurons in the CAN increased from 40^2 to respectively 48^2, 56^2 and 64^2.

40-deterministic, **40-poisson-module**, **40-poisson-neuron**:
Subsampling-based attenuation as described above, while leaving the CAN size unchanged at 40^2.

From Fig. 10a we can see that increasing the number of neurons in the CAN module does help to increase the range of viable attenuation levels, and thus the range of grid scales attainable. However, the improvement occurs at the wrong end of the scale, for attenuation levels less than 1. These attenuation levels would be used to produce smaller-scaled grid modules, but there is not any major improvement for the high attenuation levels, which is what we need in order to increase the overall usable range of the system. Fig. 10b shows the results from the update modes based on subsampling. The deterministic mode, as expected, shows artifacts from only being able to correctly represent integral attenuation levels. Both the per-module and per-neuron probabilistic update modes show promising results; for all attenuation levels from 1 and up, the observed network flows are close to the desired target flows needed to produce the intended grid scales. Neither of the subsampling-based update modes are built to adhere to attenuation levels less than 1, which would require updating the modules/neurons more often than once per timestep.

Fig. 11 shows timing results obtained during the trials in Figs. 9 and 10. For each trial the amount of CPU seconds elapsed per simulated outbound second was calculated, and these values were then averaged across trials for each attenuation value, for each tested model configuration. As expected, the CPU time for the velocity-based configurations is constant as the attenuation level changes, because the same amount

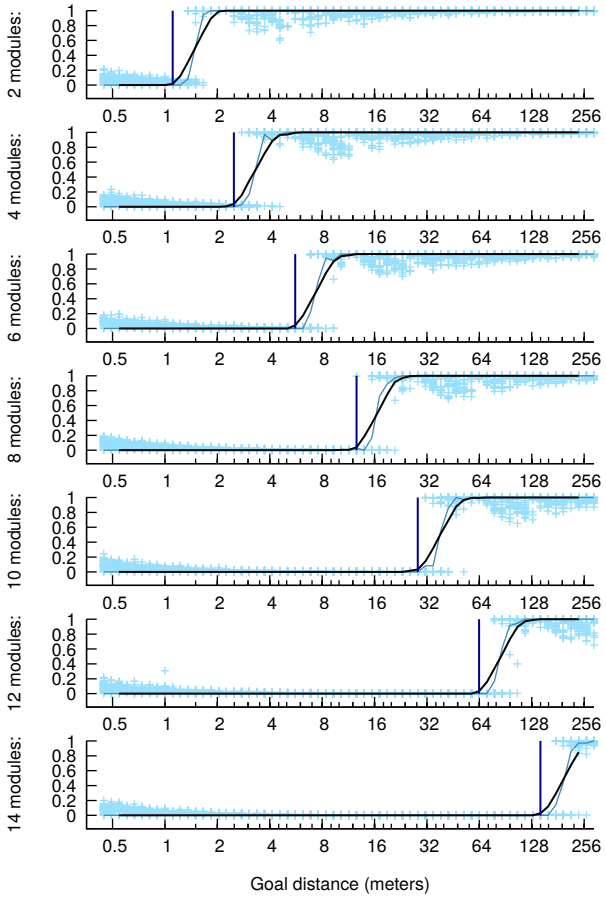

Fig. 12. Same experimental setup as in Fig. 7, but now using the "poisson-neuron" update mode. Legend as in Fig. 4.

of calculation takes place regardless of the attenuation. Note, however, that there is a substantial increase in CPU time going from 40^2 neurons to larger CAN modules. For all of the subsampling-based configurations, the CPU time decreases as the attenuation grows.

VII. Successful Long-Range Navigation

The results in the previous section lead us to favor the "poisson-module" and "poisson-neuron" update modes—as shown in Fig. 10b they are both able to represent large grid scales, and as shown in Fig. 11 they both require gradually less computational time as the grid scale increases. Future work should characterize whether there is any substantial difference between the two update modes e.g. in their ability to handle more fluctuating trajectories than the straight lines tested here, but for now we will proceed with the "poisson-neuron" update mode as our choice to evaluate in the full navigational system.

We re-ran the experiments from Section IV using the "poisson-neuron" update mode, with the results corresponding to Figs. 7 and 8 shown respectively in Figs. 12 and 13. Particularly comparing Figs. 8a and 13a, we can see that the shortfall previously experienced at 12 and 14 modules has been resolved—the viable range keeps growing exponentially as more modules are added beyond 10. Comparing

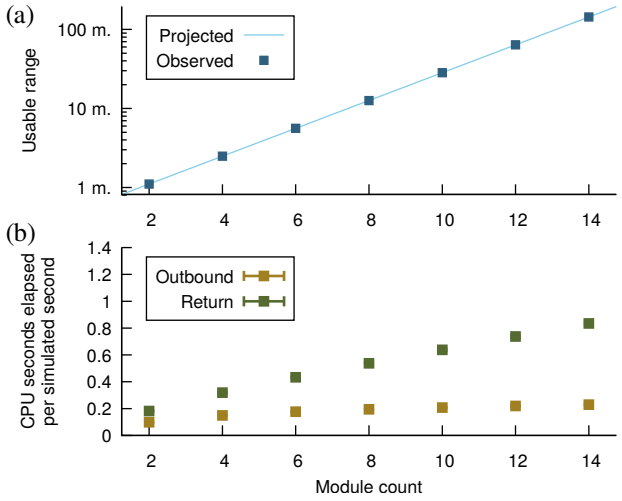

Fig. 13. Aggregate results as in Fig. 8, but now based on the experiment shown in Fig. 12, where the "poisson-neuron" update mode is used.

Figs. 8b and 13b we can see that the updated model requires less CPU time than before, and that the growth is at most linear in the number of grid modules added to the system.

VIII. Discussion

The basis for this project was a grid cell-based neural navigation system capable of vector navigation based on path integration processes in a geometric progression of grid modules [6]. We sought to determine whether such a system, based on CANs and nested decoding of grid cells, can support vector navigation over long distances. We found that the geometric progression of grid scales is interrupted due to "pinning" of network patterns in the CAN modules at the low input velocities used to implement large grid scales. Though this work specifically used a CAN grid cell model, the key issue of accommodating strong attenuation may also be relevant for the dynamics in other grid cell models. We assessed several candidate solutions and found that a probabilistic update rule in each grid module/grid cell can successfully implement large grid scales. Using this new approach, the system restored its exponential growth in range as the number of grid modules increases, enabling navigation beyond $\sim$100 meters as seen in these experiments and expectedly to much farther distances.

Bush et al. [15] demonstrated successful vector navigation with grid cells over distances of hundreds of meters, albeit in a setup where the grid cell signal was externally generated and thus would not suffer any issues arising from a neural path integration process. In any case, their largest grid scale is only $\sim$5 meters, the large final navigational range achieved by using a decoder capable of utilizing the "combinatorial" view of the grid cell code. In our present work we show that navigation at distances of $\sim$100 meters can also be accomplished when taking a "nested" view of the grid cell code and producing the grid cell signals in path integrating neural networks.

We saw an exponential growth in viable range as extra grid modules were added, with at most a linear growth in CPU time. We have thus demonstrated a grid cell-based neural navigation system where the CPU requirements are logarithmic in the desired navigational range. This is encouraging for the prospects for implementing this system to run in real time on a robot. For such an application it would also be natural to integrate additional spatial neurons, such as place cells.

Acknowledgement

Thanks to Keith Downing for helpful feedback on this work.

References

[1] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[2] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[3] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot *et al.*, "Mastering the game of go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, 2016.

[4] D. L. Yamins and J. J. DiCarlo, "Using goal-driven deep learning models to understand sensory cortex," *Nat. Neurosci.*, vol. 19, no. 3, pp. 356–365, 2016.

[5] A. Graves, G. Wayne, M. Reynolds, T. Harley, I. Danihelka, A. Grabska-Barwińska, S. G. Colmenarejo, E. Grefenstette, T. Ramalho, J. Agapiou *et al.*, "Hybrid computing using a neural network with dynamic external memory," *Nature*, vol. 538, no. 7626, pp. 471–476, 2016.

[6] V. Edvardsen, "Goal-directed navigation based on path integration and decoding of grid cells in an artificial neural network," *Natural Computing*, 2016.

[7] J. O'Keefe and J. Dostrovsky, "The hippocampus as a spatial map. Preliminary evidence from unit activity in the freely-moving rat," *Brain Res.*, vol. 34, no. 1, pp. 171–175, 1971.

[8] J. S. Taube, R. U. Muller, and J. B. Ranck, "Head-direction cells recorded from the postsubiculum in freely moving rats. i. description and quantitative analysis," *J. Neurosci.*, vol. 10, no. 2, pp. 420–435, 1990.

[9] T. Solstad, C. N. Boccara, E. Kropff, M.-B. Moser, and E. I. Moser, "Representation of geometric borders in the entorhinal cortex," *Science*, vol. 322, no. 5909, pp. 1865–1868, 2008.

[10] T. Hafting, M. Fyhn, S. Molden, M.-B. Moser, and E. I. Moser, "Microstructure of a spatial map in the entorhinal cortex," *Nature*, vol. 436, no. 7052, pp. 801–806, 2005.

[11] Y. Burak and I. R. Fiete, "Accurate path integration in continuous attractor network models of grid cells," *PLoS Comput. Biol.*, vol. 5, no. 2, 2009.

[12] L. M. Giocomo, M.-B. Moser, and E. I. Moser, "Computational models of grid cells," *Neuron*, vol. 71, no. 4, pp. 589–603, 2011.

[13] M. Milford and G. Wyeth, "Persistent navigation and mapping using a biologically inspired SLAM system," *Int. J. Robot. Res.*, vol. 29, no. 9, pp. 1131–1153, 2010.

[14] F. Carpenter and C. Barry, "Distorted grids as a spatial label and metric," *Trends Cogn. Sci.*, vol. 20, no. 3, pp. 164–167, 2016.

[15] D. Bush, C. Barry, D. Manson, and N. Burgess, "Using grid cells for navigation," *Neuron*, vol. 87, no. 3, pp. 507–520, 2015.

[16] M. Geva-Sagiv, L. Las, Y. Yovel, and N. Ulanovsky, "Spatial cognition in bats and rats: from sensory acquisition to multiscale maps and navigation," *Nat. Rev. Neurosci.*, vol. 16, no. 2, pp. 94–108, 2015.

[17] H. Stensola, T. Stensola, T. Solstad, K. Frøland, M.-B. Moser, and E. I. Moser, "The entorhinal grid map is discretized," *Nature*, vol. 492, no. 7427, pp. 72–78, 2012.

[18] I. R. Fiete, Y. Burak, and T. Brookings, "What grid cells convey about rat location," *J. Neurosci.*, vol. 28, no. 27, pp. 6858–6871, 2008.

[19] X.-X. Wei, J. Prentice, and V. Balasubramanian, "A principle of economy predicts the functional architecture of grid cells," *eLife*, vol. 4, 2015.

[20] S. Sreenivasan and I. Fiete, "Grid cells generate an analog error-correcting code for singularly precise neural computation," *Nat. Neurosci.*, vol. 14, no. 10, pp. 1330–1337, 2011.

[21] M. Stemmler, A. Mathis, and A. V. Herz, "Connecting multiple spatial scales to decode the population activity of grid cells," *Sci. Adv.*, vol. 1, no. 11, p. e1500816, 2015.