Grid Cells as a Plausible Velocity Detection Mechanism of Visual Stimuli

Wells Crosby[1], Ella Kazazic[1]

[1]Computation and Cognition Department, MIT

Abstract

Grid Cells are known to form 2D spatial coordinates for mammals navigating environments, and further applications in memory have recently been shown [4]. We investigate the feasibility of grid cells as also forming a backbone of the visual system. We show that grid cells form a plausible mechanism for detecting the velocity of an input stimulus across a sheet of grid cells. In a simulated grid cell sheet under certain constraints, we show that the sum activation of the sheet is greatly increased when the global velocity of the sheet matches the velocity of an input stimulus traveling over that sheet. This forms a feedback mechanism which can be used to tune the global velocity towards the input velocity, effectively detecting velocity. Additionally we show an extension of the velocity input method shown in Burak et al. 2009 [1] with greater biological plausibility. Finally, we publicly release a GPU accelerated convolutional grid cell implementation with full interactivity to aid further grid cell research.

Introduction

For years, how place cells (cells which fire selectively in different physical locations) operated accurately was a mystery, with many studies looking into different brain regions for the source of their inputs. With Hafting et al. in 2005 [2] the structure of grid cells in the entorhinal cortex was finally discovered. The firing patterns of collections of these grid cells sheets were shown to form a coordinate system while navigating through physical spaces. The firing patterns of grid cells form a triangular lattice with stable spacing and orientation, but with a varying phase while navigating an environment. By considering together the phase information from a number of grid cell networks (each with varying spacing and orientation) a mammal can accurately locate itself along large distance scales.

For the phase of the grid cell networks to change while navigating, a global input velocity must be applied to the networks. This input velocity perturbs the systems so that the stable configuration of the grid cell network translates appropriately [3]. A biologically plausible mechanism for the simulation of grid cell networks, and their global velocity input was described in Burak et al. 2009 [1]. Their method for simulating the grid cell network relies on the creation of a continuous attractor. A center surround kernel is applied to each cell in the sheet, where activations a short distance from the cell are excitatory, activations a medium distance away from the cell are inhibitory, and activations a long distance from the cell have no effect (this can also be formed with purely inhibitory connections and a global excitation). This formulation pushes the network to form the triangular lattice firing pattern of grid cells seen in the entorhinal cortex. A global velocity input is handled by having the center of the kernel be displaced in one of the four cardinal directions in a regular pattern across the sheet of neurons. When all of the neurons are able to act normally, the effect of the offset kernels cancel eachother out, leaving a positionally stable network. If the set of neurons in one of the cardinal directions is inhibited, this causes the pattern to drift in the opposite direction of the offset. With an inhibitory connection to each of the four cardinal directions, a global velocity can be inputted in any direction.

Artificial grid cell networks paired with an artificial hippocampus have been shown to form an effective tool for memory stabilization and retrieval in, especially in regards to episodic memory, suggesting their plausible use for memory in the brain [4]. This work used the global velocity mechanism of a grid cell network to effectively encode high dimensional transitions between different visual stimuli in a low dimensional grid cell space, aiding in memory tasks. Additionally, artificial grid cell networks have been shown to be effective in encoding eye saccades while visually assessing new visual stimuli [5]. Using the encoded saccade information in the grid cell network, object recognition tasks are improved. These two recent works show great promise for the application of grid cells in visual processing, however the use of grid cell networks in object tracking/motion processing remains unexplored.

Background

Recurrent Neural Networks are a class of neural networks where the outputs from one iteration of the

network are used to get a resultant value from the network as well as feed back as inputs into the next iteration of the network. Alongside the previous output of the network, an external input is also fed into the network at each iteration. This structure allows for an evolution of the initial state towards a new, and perhaps more useful state.

Many structures in the brain share properties with Recurrent Neural Networks, with graphs of neurons often forming recurrent connections, which lead to a useful output. Grid cells form a pretty clear relationship to Recurrent Neural Networks. The sheet of neurons formed by grid cells are effectively using the previous state of the neurons to determine the next state of the neurons (through the excitation and inhibition of the center surround kernel) over and over again. We can input a global velocity or excite certain neurons of the network to influence the evolution of the system, while reading out certain properties from the system. In fact structures like grid cells have been shown to emerge from training an artificial recurrent neural network to do a localization task [6].

Related Works

There has been a recent growth of work using grid cells with visual stimuli processing. Chandra et al. 2023 [4] showed that grid cells can be used to enhance visual memory. When processing a series of visual stimuli, the high dimensional representation of each visual stimuli in the hippocampus is transformed into a low dimensional representation living in grid cell space. There exists a relation between the grid cell representation and the hippocampal representation so that given a grid cell representation of the stimulus, it is easier to retrieve the hippocampal representation. When you are trying to remember a sequence of visual stimuli, instead of recalling the high dimensional transformation from one stimuli to another, all the brain has to recall is the low dimensional transition in grid cell space between the stimuli, making episodic memory much more feasible. While this paper showed the viability of grid cells in the later processing stages of visual input, the early stages of visual processing are not accounted for.

Bicanski et al. 2019 [5] showed that grid cells can be used to enhance object recognition. When identifying an object, you are quickly moving your eye from place to place over the object (known as saccading), capturing the local information at that point. To build up a representation of the object as a whole the local information at each point must be spatially integrated together. Here grid cells are shown to encode the positional variation of the saccades. Through integration of grid cells, an artificial neural network was shown to improve on its visual object recognition ability. This paper shows the usefulness of grid cells in the earlier stages of visual processing, however it only accounts for static object recognition.

In our paper we show that grid cells can be further used in visual processing, by forming a mechanism to identify the velocity of moving visual stimuli.

Methods

In order to begin testing the efficacy of grid cells in visual processing, we first needed an implementation of the grid cells. We decided to construct our own implementation roughly inspired by methods described in Burak et al. 2009 [1]. For our implementation we decided to prioritize computation speed to allow for real-time interaction with the grid cell network. To achieve high computation speed we decided to use a convolutional kernel, which allowed each neuron to only receive input from the neurons within a relevant area, instead of from the whole neural sheet. Additionally our implementation is executed entirely using PyTorch, which lets the program massively parallelize on the GPU.

The first item to note in the creation of the grid cell network is the construction of the kernel. The kernel is radially symmetric, so the key factor in creating the kernel is to define a function of the distance from the center. In the Burak et al. 2009 paper the kernel is positive in the center, smoothly descends to a negative value, and then smoothly ascends back to 0. We found this type of kernel to be ineffective for generating the grid cell pattern with our setup. Instead we found our own kernel generation scheme to work well.

$$\text{Gaussian}(x, \mu, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right) \quad (1)$$

$$\text{val}(d, r) = \cos\left(\frac{d}{r} \cdot \frac{3\pi}{2}\right) \cdot \text{Gaussian}\left(\frac{d}{r} \cdot \frac{3\pi}{2}, 0, 2.9\right) \quad (2)$$

For each pixel within the kernel of radius r, we first need to set the value of the pixel to the result of Equation 2 with parameters (distance from center pixel, r). We then find the mean value of all the pixels we set and subtract that mean from each pixel. Since PyTorch expects a square kernel, every pixel with distance from the center pixel greater than r will be set to 0. We now have a kernel generation scheme that results in the kernel in Figure 1. A cross section cutting through the center of the kernel can be seen in Figure 2. Note: to create a center surround

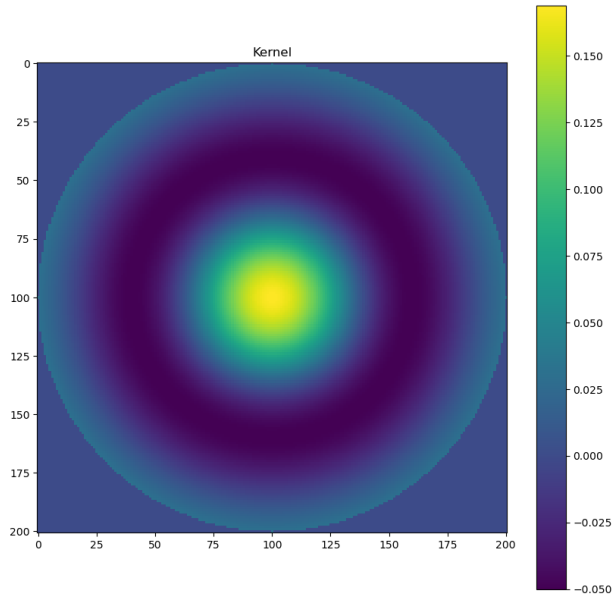

Figure 1: *Center Surround Kernel*

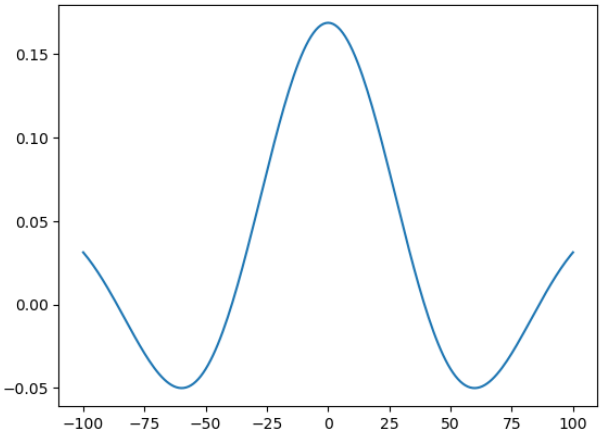

Figure 2: *Kernel Cross Section*

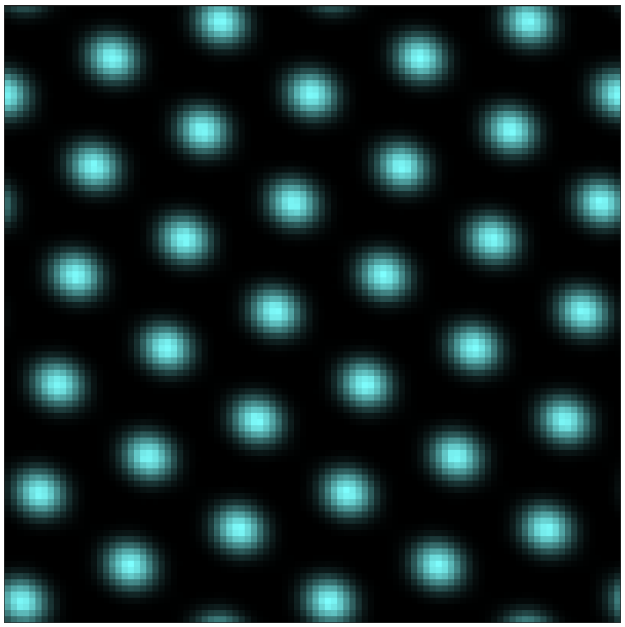

Figure 3: *Static Triangular Grid*

pattern offset from the center of the kernel is simple, instead of feeding the distance from the center pixel into Equation 2, you instead feed the distance from the pixel you want the center surround pattern to lie upon.

There are a few notable differences to the kernel used in the Burak et al. 2009 paper. Firstly, the mean value of our kernel is 0. We found that having a mean of 0 aided in the stability of the results. Secondly, after the values dip into the negatives, they rise back up to the positive values. By having the values around the center of the radius be positive, the propensity for generating a triangular lattice is increased. You can think of this as the value of any pixel prefers not only to have similar values around it, but also similar values one radius away from it. This property is not strictly necessary however it added to the stability of the grid cell pattern. Finally, our use of a cos function along side a Gaussian in Equation 2 is a bit unique. Typically the kernels for grid cell simulations rely on exponent based functions solely, however we found the cos function to produce more stable results.

It is worth mentioning that this kernel generation scheme was created mostly ad hoc, with little theory to back up why the things that worked ended up working. In further works, optimizing the kernel generation to provide even better stability and biological plausibility would be worthy of study.

Now that we have a kernel generating scheme, we can move on to actually running the grid cell simulation.

To achieve a simple static triangular lattice is now relatively simple. We first initialize the 2D grid cell sheet with random noise between 0 and 1, any sort of noise will do here so long as activations are not constant. We now are going to apply the kernel we created to the whole grid using torch.nn.Conv2d to run the process on the GPU. Importantly we need to use circular padding, otherwise the grid cell pattern is unable to form. Circular padding effectively maps the 2D sheet onto a torus, so the neurons on the left side are adjacent to the right side, and the neurons at the top are adjacent to the neurons on the bottom. Having performed one iteration of the convolution, we now want to clamp the activations to be between 0 and 1. To do this we subtract the minimum value from the whole sheet, and then divide each value by the new maximum of the sheet. This is done to prevent the network from diverging to positive or negative infinity. We suspect that a more careful creation of the kernel would render this step unnecessary. Now that we have the constrained convolution update, we will now take each of the activations and raise them to a power. To achieve the

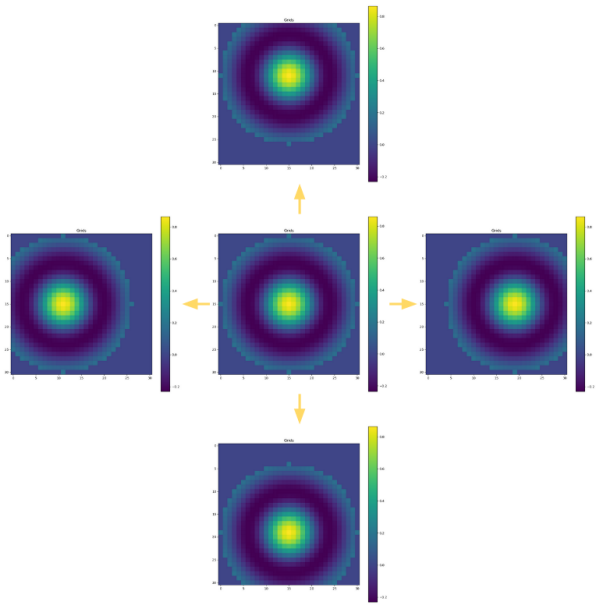

Figure 4: *Cardinal Direction Kernels*

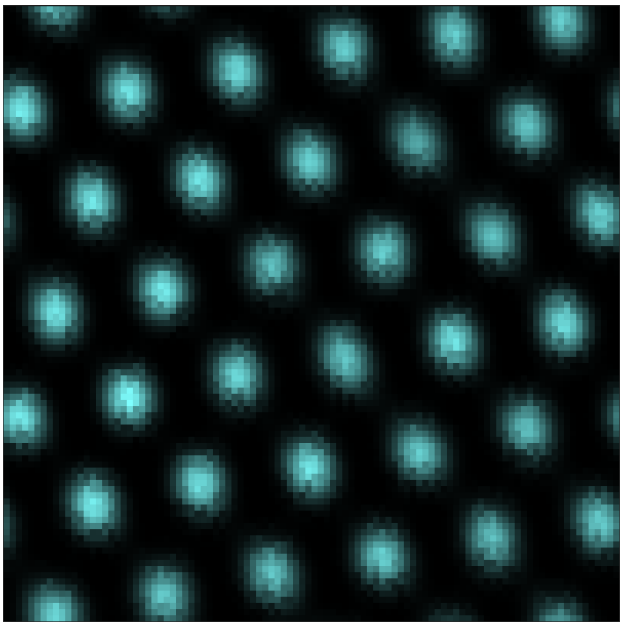

Figure 5: *Grid Cells With Velocity Perturbation*

triangular lattice we are after, setting this value (with or setup) to around 1.3 works well. This is perhaps the most unique step in our simulation procedure, and messing with the value of the power leads to intriguing results we will explore later.

We now have a stable grid cell network as you can see in Figure 3! This is all well and good, however to do most anything interesting with the grid cells we need a method for inputting a global velocity.

The easiest way of making this happen in this scheme is to simply regenerate a new kernel for each iteration which is offset in the direction you want the pattern to move. While this is easy, and produces the cleanest results possible, it is not very biologically plausible. We want a way of inputting a global velocity that does not incur any weight updates in the network, so modifying the kernel as we run the simulation is off of the table.

The method we used to make this happen takes inspiration from the Burak et al. 2009 paper. Instead of changing the offset of all kernels as we run the simulation, we instead randomly distribute locations in the grid cell network where the kernel is offset by a fixed amount in a random cardinal direction as shown in Figure 4!. By balancing the inhibition of the sets of cells offset in each cardinal direction we are able to input a global velocity into the system. One caveat with our implementation is that we are having four cells (one for each cardinal direction) located at each randomly chosen location, with each taking input from its surroundings, but feeding into an accumulation cell which takes the weighted (by the global velocity) sum of the four directions. Now we have a biologically plausible global velocity input into our system! You can see the velocity cells modifying the activations in Figure 5.

A note on the PyTorch implementation of this velocity input is that we found the most efficient way to get the convolution for each cardinal direction was to simply to run the convolution with each offset kernel over the whole network, and then use a mask to set the velocity input cells to the proper values.

This recipe is an effective way of creating a stable, biologically plausible grid cell network which is able to run extremely quickly by leveraging the parallelization power of the GPU. The evaluation of the performance of our network was primarily visual, we wanted to see the triangular lattice appear, and for a global velocity to properly translate the pattern. This evaluation mechanism was very effective for getting the simulation to perform as we hoped.

Results

With the grid cell implementation completed we are able to run some interesting experiments. The first notable finding lies in how the power we raise the activations to during an iteration of the simulation changes the behavior of the network.

With a power of 1, the lattice structure formed is not the triangular lattice we see in real grid cells, but instead a square lattice. You can see this result in Figure 6. We suspect that the reason for a square lattice cropping up from the power of 1 lies in how the spaces between the areas of high activation are able to meld between the high points. This effect is relatively weaker in a triangular lattice, so without raising to a power greater than 1, and therefore

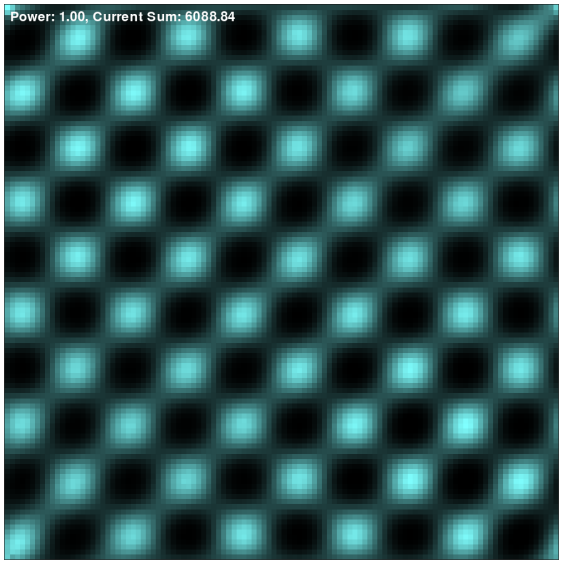

Figure 6: *Square Lattice Formation*

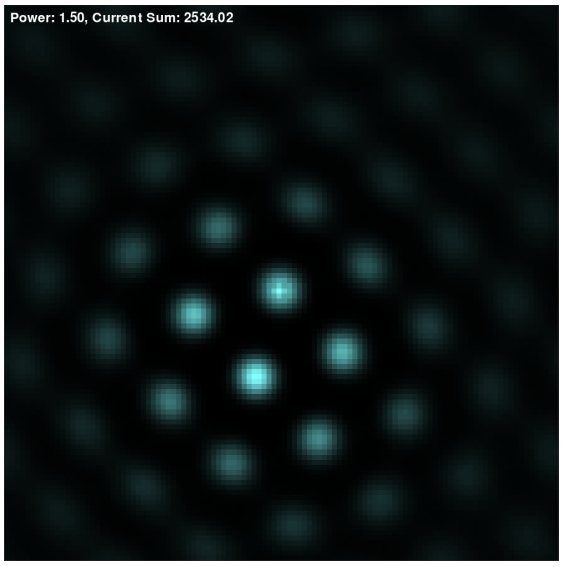

Figure 7: *Stationary Pattern*

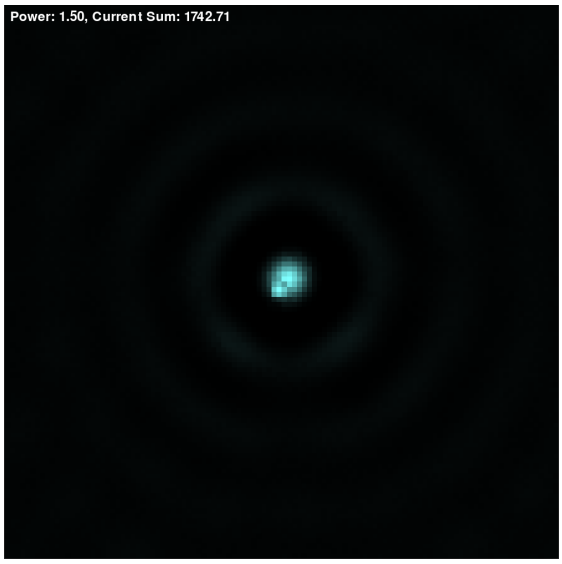

Figure 8: *Partially Collapsed Pattern*

pushing smaller activations down, the square lattice is preferred.

Raising to a power above 1 eventually pushes the structure towards a triangular lattice, however as stated before at around 1.3, we quickly and stably see this behavior.

If we continue to raise the power to around 1.7, the pattern completely collapses down to just one high point, producing a largely uninteresting result.

However if we find a sweet spot around a power of 1.5 and add an input stimulus (simply setting a small region of activations to 1), we start to see very compelling behavior. If we let the input stimulus remain static, the network forms a triangular lattice which is brightest around the input point and diminishes in brightness as you move away from the input. You can see this behavior in Figure 7. If we instead start moving the position of the input stimulus over time, or equivalently set a global velocity while the input stimulus remains static, the pattern partially collapses down into a single point around the input stimulus. You can see this behavior in Figure 8. If instead we set the global velocity to match the velocity of the input stimulus over the grid cell network, the the pattern seen in Figure 7 re-emerges, moving with the input stimulus.

This is extremely compelling behavior, as we have now constructed an effective feedback mechanism for detecting whether or not the global velocity of the network matches the velocity of an input stimulus. When the velocities match, the network forms the pattern in Figure 7 and when they disagree, they form the pattern in Figure Figure 8 (with closeness of match blending between these two extremes). To get the feedback on whether or not we have a velocity match, we simply need to sum up the activations of the network. When the velocities are matching, the sum of the activations is significantly higher than when the velocities are not matching.

We ran an experiment to quantify how matching the velocities effects the sum of activations. To perform this experiment we simply manually controlled the global velocity of the network and the velocity of the input stimulus separately. Running this experiment produced the results shown in Figure 9 (note that the blip at the start of the graph is a startup effect). You can clearly see that when the velocities are in agreement, the sum of activations quickly rises and remains stable. When the velocities disagree however, the sum activation quickly drops to a much lower value and remains there until the velocities agree once more.

With this method for detecting the level of agreement between the global velocity of a grid cell network and the velocity of an input stimulus, making

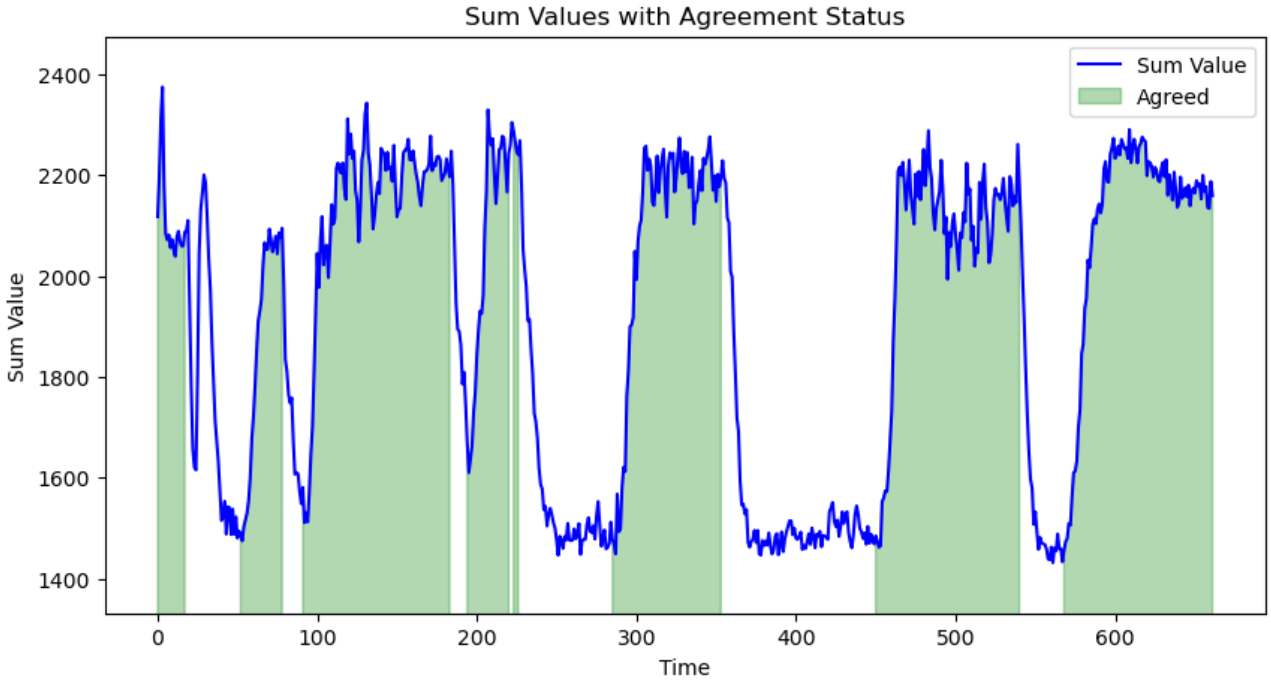

Figure 9: *Activation Sum With Agreement*

the global velocity match the input stimulus velocity is conceptually simple. We have a feedback mechanism, so something like a tuned PID loop should be able to keep the two velocities in sync.

Discussion

We have shown that grid cells are a plausible mechanism for velocity detection of a visual stimulus. When the global velocity of the network matches the velocity of an input stimulus, there exists a clear feedback mechanism. Our results show a new plausible capability for grid cells which opens up the door to the use of grid cells in further cognitive functions.

We have extended the work of Chandra et al. (2023) [4] and Bicanski et al. (2019) [5]. Their papers showed the use of grid cells in visual memory and visual object recognition, however did not address the dynamic nature of the visual world. We show that grid cells are plausibly useful building block towards understanding of dynamic visual stimuli.

While the primary context of this paper is centered around processing visual stimuli, we should not ignore the possible applications to other ares. Detecting the velocity of input stimuli along a grid cell network is a generally useful capability that could be applied to further sensory areas such as tactile sensation, and auditory perception. This capability could also build towards higher level cognitive tasks such as language processing and task planning.

In addition to the discovered plausible capability of grid cells, we also show a unique implementation for simulating grid cells. Our implementation has interesting tuning capabilities in changing the power parameter. Our implementation also shows the viability and advantage of running the simulation on the GPU. Without the ability to interact with the simulation in real time, it would have been much harder to discover unique behavior.

Conclusion

Our results demonstrate the feedback mechanism which could underlie a velocity detection neural circuit, however we did not build that circuit, doing so would be an interesting area of further research. Additionally we did not quantify the behavior of the feedback mechanism in a very wide variety of situations, which would also be a useful area of further study. While the implementation of our grid cell simulator is state of the art so far as we can tell, there is clear room for improvement in theoretical guarantees, biological plausibility, and performance optimization.

Appendix

[1] Burak, Yoram, and Ila R. Fiete. "Accurate path integration in continuous attractor network models of grid cells." PLoS computational biology 5.2 (2009): e1000291.

[2] Hafting, Torkel, et al. "Microstructure of a spatial map in the entorhinal cortex." Nature 436.7052 (2005): 801-806.

[3] Moser, Edvard I., May-Britt Moser, and Yasser Roudi. "Network mechanisms of grid cells." Philosophical Transactions of the Royal Society B: Biological Sciences 369.1635 (2014): 20120511.

[4] Chandra, Sarthak, et al. "High-capacity flexible hippocampal associative and episodic memory enabled by prestructured “spatial” representations." bioRxiv (2023): 2023-11.

[5] Bicanski, Andrej, and Neil Burgess. "A computational model of visual recognition memory via grid cells." Current Biology 29.6 (2019): 979-990.

[6] Cueva, Christopher J., and Xue-Xin Wei. "Emergence of grid-like representations by training recurrent neural networks to perform spatial localization." arXiv preprint arXiv:1803.07770 (2018).