Visual Motion Perception using Critical Branching Neural Computation

Janelle K. Szary (jszary@ucmerced.edu)
Christopher T. Kello (ckello@ucmerced.edu)
Cognitive and Information Science, 5200 North Lake Rd., Merced, CA 95343 USA

Abstract

Motion processing in visual systems is supported by various subcortical and cortical microcircuits. However, all motion processing requires a basic capacity to integrate and combine information over time, as may be true for all microcircuits that support perceptual and cognitive functions. In the present study, a generic microcircuit model is presented that self-tunes its recurrent spiking dynamics to its critical branching point. The model is shown to have generic memory capacity that can be tapped for the purpose of visual motion processing. These results suggest that critical branching neural networks may provide general bases for spiking models of motion processing and other perceptual and cognitive functions.

Keywords: Critical branching; reservoir computing; leaky integrate-and-fire neural networks; motion perception.

Introduction

Communication in neural networks largely occurs via thresholded spiking signals between neurons, which are connected by characteristically recurrent loops varying in spatial and temporal scale (Buzsáki, 2006). This connectivity structure produces patterns of network activity that are continually in flux, and in this sense network dynamics cannot be characterized by simple point or limit cycle attractors. This limits the extent to which attractor-based neural network models, common in cognitive science (e.g. connectionist recurrent networks), can relate the brain's spiking dynamics with cognitive functions.

In order to investigate how computational functions might be based in non-attractor network dynamics, Maass et al. (2002b) and Jaeger (2001) developed the concept of *reservoir computing*. The basic idea is that recurrent networks can produce dynamics with a generic capacity for computation, and so-called "read-out" networks can learn to interpret reservoir dynamics for task-specific purposes.

The present work is focused on Maass et al.'s (2002b) *liquid state machine* (LSM) framework, which consists of a pool of leaky integrate-and-fire (LIF) units with random recurrent connectivity that is roughly constrained to be local and sparse. Time-varying input signals are fed to the network to drive and alter the "liquid" spiking dynamics. Due to recurrence, thresholding, and other non-linearities, the resulting spatial and temporal spike patterns are complex, unknown, and unlearned functions of their inputs.

Nonetheless, these spike patterns will contain information about past inputs to the extent that their dynamics are not overly convergent or overly divergent (i.e. with Lyapunov exponents near one; Bertschinger & Natschläger, 2004). Maass et al. (2002b) found that even when the input patterns are non-linearly separable (e.g. XOR and parity), simple linear classification can be used to categorize inputs based on the liquid spike patterns they produce. Thus, simple learning algorithms can be used to interpret complex spiking dynamics for non-trivial functions and tasks. Moreover, the authors showed that functions of *past* inputs can be learned from *current* spike patterns because memory is an inherent property of recurrent dynamics that are not overly convergent or divergent.

The efficacy and simplicity of reservoir computing makes it an attractive generic mechanism for computational perception tasks. In fact, Verstraeten et al. (2005) used LSMs to classify spoken words, and Maass et al. (2002a) used them to classify the shapes and velocities of visual objects in motion. Burgsteiner et al. (2006) used real-time video footage from the RoboCup Soccer challenge to show that an LSM could be used to predict the future location of a soccer ball in motion.

Visual Motion Perception

Biological motion perception starts with motion detectors in the retina, where ganglion cells just two or three synapses away from rods and cones detect motion in preferred directions (Vaney et al., 2000). Behaviorally, sensitivity to the direction of motion is demonstrated in motion coherence studies (reviewed in Braddick, 1997), where even minute asymmetries in the percentage of motion in one or another direction can be detected. Information about the direction of motion is used for navigation, such as when coherent motion across the visual field constitutes optic flow, and for the detection of objects in motion, such as when local motion differs from global motion (Srinivasan, 2000).

Tracking objects is another important function of motion perception, i.e. the ability to predict future locations of objects in motion. This kind of prediction is hypothesized to be necessary for accurate smooth-pursuit eye movements (Carpenter, 1988), and it enables organisms to either avoid collisions with moving objects, or coincide with them as in catching or hitting a baseball. For instance, McBeath (1990) showed that the motor action of aiming and swinging a bat must be implemented when the ball is just over halfway to the plate, using a prediction of where the ball *will be*. Similarly, studies of cricket batters' eye movements show predictive eye fixations on where the ball will be when it crosses the mound, rather than where it currently is (Land & McLeod, 2000).

The present work focuses on both of these functions of motion processing: direction classification and future location prediction. Reservoir computing techniques are particularly well-suited for motion perception tasks, as both

are inherently dynamical in nature. That is, microcircuits involved in motion processing must support a wide range of visual functions in the face of constantly changing inputs, and must be sensitive to the time course of these inputs.

Critical Branching Applied to Motion Processing

In the present study, motion processing tasks are used to investigate the efficacy of a generic circuit implemented using the reservoir computing framework. The model builds on previous work by Maass et al. (2002a) and Burgsteiner et al. (2006) who developed LSMs for predicting future locations of objects in motion. Here we replicate and extend their work by 1) developing a new model for tuning reservoir dynamics of LIF units, and 2) applying the model to a wider range of tests that more thoroughly examine its capacity for motion processing.

The present model is based on Kello and Mayberry's (2010) work on *critical branching* neural computation (see also Kello & Kerster, 2011; Kello et al., 2011). The basic idea is that the computational capacity of reservoir spiking dynamics should be enhanced when near their critical branching point. Any spiking neural network can be viewed as a branching process whereby a spike occurring at time t may subsequently "branch" into some number of spikes at time $t+\Delta t$ over the neurons connected via its axonal synapses. Let us call the former an "ancestor" presynaptic spike, and the latter "descendant" postsynaptic spikes. The number of descendants divided by ancestors is the *branching ratio* of a spiking network: $\sigma=N_{post}/N_{pre}$. If $\sigma<1$, spikes diminish over time and information transmission through the network is inhibited by dampened propagation of spiking signals. If $\sigma>1$, spikes increase over time and eventually saturate the network, which also inhibits information transmission. $\sigma=1$ is the critical branching point at which spikes are conserved over time, and so propagate without dying out or running rampant. A related concept of "homeostatic synaptic scaling" is being investigated in cortical circuits, and refers to the idea that these circuits also adjust themselves to achieve a balanced level of overall activity in terms of mean firing rate (for a review, see Turrigiano, 2008).

Computational models verify the information processing advantage of a network with $\sigma=1$. For example, an analogous critical point between convergent and divergent dynamics (i.e Lyapunov exponents near one) has been shown to maximize memory capacity in a recurrent network of threshold gating units (Bertschinger & Natschläger, 2004). Here we formulate a reservoir computing model using LIF spiking neurons that are more biologically realistic compared with threshold gate neurons. Our model includes a simple algorithm that probabilistically potentiates and de-potentiates synapses so that reservoir dynamics approach their critical branching point.

We presented inputs to the model that come from a simple, abstracted visual field. The visual field consisted of nothing other than a diamond-shaped moving object on a 12x12 grid with periodic boundary conditions (i.e. left/right and top/bottom edges wrap around). Inputs created and perturbed ongoing spiking dynamics in a pool of LIF reservoir neurons. A group of perceptron units was used to "read out" spiking dynamics for two different functions of motion processing: Direction classification and future location prediction. These functions are similar, except that direction classification requires generalization across position (and has a higher chance rate of performance), whereas future location prediction is position-specific (and has a lower chance rate of performance).

In Simulation 1, we test whether critical branching spiking dynamics have the kind of memory capacity capable of simultaneously supporting these two motion processing functions, using simple straight-line motions that require only modest memory capacity for accurate performance. In Simulations 2 and 3, we employ more complex zig-zag and spiral motions that require greater memory capacity.

Reservoir Computing Model

Our model was based on the LSM framework, which consists of three layers: input units, reservoir units (tuned to critical branching, hereafter referred to as the "CB layer"), and readout units (Figure 1). The input layer was a 12x12 grid, and each unit projected a synaptic connection onto each of the reservoir units with probability 0.5. There were 400 LIF units in the CB layer, and each one projected recurrently onto each other with probability 0.5. There were four direction classification readout units, and twenty-four location prediction readout units (representing two 12-unit axis coordinates). Each unit in the CB layer projected onto every readout unit. Input patterns were diamonds with sides approximately 7-8 units in length.

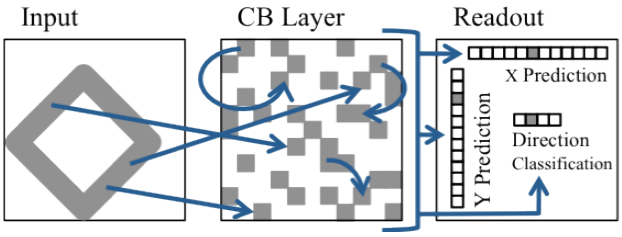

Figure 1: Model architecture. Gray squares show example spike patterns, and blue lines represent synaptic connectivity

Model Variables and Update Equations. LIF units generally have the following variables (Roman letters) and parameters (Greek letters): A membrane potential V_i for each neuron i, a membrane threshold θ_i and membrane leak λ_i, and a level of potentiation w_j for each axonal synapse j, where $w_j>=0$ for excitatory neurons and $w_j<=0$ for inhibitory neurons. Models may also include variable synaptic delays τ_j, as well as parameters governing the time course of action potentials and postsynaptic potentials (e.g. membrane resistance).

Our model included all of the above, except that action potentials and postsynaptic potentials were instantaneous for the sake of simplicity. We sought to enhance the

biologically plausibility of the model with 1) variable updates that were local in time and local with respect to immediately connected synapses and neurons (numerical values were not transmitted over connections among neurons, as they are in e.g. backpropagation), and 2) synaptic and neuronal updates were asynchronous and event-based. The latter criterion helped further the plausibility of our critical branching tuning algorithm.

Each update event in the model begins when a given neuron receives as input a postsynaptic potential I_j at time t, which may either come from another neuron within the CB layer, or from the input layer:

$$V_i \leftarrow V_i e^{-\lambda_i (t-t')} + I_j, \quad [1]$$

where $\leftarrow$ denotes the instantaneous update of a variable, and t' is the previous time that V_i was updated. Thus, the model included continuous exponential leak, applied each time a given neuron received an input. Immediately after each V_i update, if $V_i > \theta_i$, then $V_i \leftarrow 0$, and a postsynaptic potential I_j was generated for each axonal synapse of i. Each $I_j = w_j$, and was applied at time $t+\tau_j$.

In a typical connectionist model, w_j can be any real-valued number, possibly bounded by some minima and maxima. However, neurophysiological evidence indicates that synapses may only have a limited, discrete number of levels of potentiation, possibly just two (Petersen et al., 1998). Therefore we used binary-valued synapses in order to limit the number of potentiated synapses ($w_j \neq 0$), and to enable a stochastic tuning algorithm. In particular, each synapse had two possible strengths i.e. levels of potentiation, 0 or φ_j. Each LIF model neuron has two free parameters, λ_i and θ_i, and each synapse has two free parameters, τ_j and φ_j. Values for all four free parameters were sampled randomly from uniform distributions whose ranges were set to reasonable default values. In particular, values were real numbers in the ranges $1 < \theta_i < 2$, $0.5 < \lambda_i < 1$, $1 < \tau_j < 1.5$, $1 < \varphi_j < 2$ for excitatory units, and $0.1 < \varphi_j < 1$ for inhibitory units. The decision to set φ_j higher for excitatory units was driven by performance considerations, rather than neurophysiological data.

The set of membrane potentials V and postsynaptic potentials I comprise the dynamics of neurons in our LIF model. These variables are governed by event-based updates (Eq 1, plus threshold dynamics) that may occur asynchronously across neurons (at any point in continuous time, simulated with arbitrary precision). The set of synaptic weights w comprise the dynamics of synapses, and are governed by the critical branching algorithm described next.

Self-Tuning Algorithm. The objective of the self-tuning algorithm is to potentiate and de-potentiate synapses so that each ancestor spike is followed by one descendant spike on average. A local estimate for σ is computed over the interspike interval (ISI) for each model neuron i. This means that only $N_{post,i}$ need be estimated, because $N_{pre,i}=1$ by definition, with respect to a given neuron's ISI. Thus, to achieve critical branching, $N_{post,i}$ should sum to one.

When a given neuron spikes, its local estimate of σ is reset, $N_{post,i} \leftarrow 0$. For each axonal synapse's *first* spike occurring at time t, $N_{post,i}$ was incremented by $s_i = e^{-\lambda_i (t-t')}$. For each increment, each descendant spike was weighted as a decaying function of the time interval between pre- and postsynaptic spikes, with maximal weighting when the former was immediately followed by the latter.

The sum of time-weighted descendants is used (before it is reset to zero) each time the neuron spikes to update weights on its axonal synapses. In particular, if $N_{post,i}<1$, then the update $w_j \leftarrow \varphi_j$ is performed for each synapse j with probability

$$\eta f(s_i) \left| N_{post,i} - 1 \right| / U, \quad [2]$$

where η is a global tuning rate parameter (fixed at 0.1), and U is the number of synapses available for potentiation. $f(s_i) = 1 - e^{-\lambda_i (t-t')}$ if neuron i was excitatory, and $f(s_i) = e^{-\lambda_i (t-t')}$ if inhibitory. If $N_{post,i}>1$, then perform the update $w_j \leftarrow 0$ with probability set according to Eq 2, except U is the number of synapses available for de-potentiation, and the assignment of $f(s_i)$ is switched for excitatory versus inhibitory neurons.

In essence, the critical branching algorithm potentiates synapses when too few descendant spikes occur, and de-potentiates when too many occur. Spikes are time-weighted because effects of ancestor spikes on descendant neurons diminish according to their leak rates. Critical branching weight updates increase in likelihood as local branching ratio estimates diverge from one, and depend on spike timing. With regard to spike timing, excitatory synapses are more likely to be (de)potentiated when postsynaptic neurons have (not) fired recently, which helps to spread spikes across neurons. The same principle leads to the opposite rule for inhibitory neurons.

Readout Layer. Readout units were not spiking units. Instead, the normalized exponential function (i.e. softmax) was used to compute their outputs from their summed inputs. The readout layer consisted of three normalized groups of units: 4 direction classification units (up, down, left, and right), 12 X-coordinate position units, and 12 Y-coordinate position units. Unlike synapses projecting into reservoir units, connections into readout units were real-valued and initialized in the range [-0.1, 0.1].

For each unit time interval of simulation, reservoir spikes resulted in normalized output activations over readout units. During training, readout units received 1-of-N targets for each of the three normalized readout groups. Targets were compared with outputs using the divergence error function, and the resulting error signal was used to update connection weights with the delta learning rule (using momentum = 0.5, learning rate = 0.00001). At testing, the maximally active readout unit in each group was compared with the targeted output to assess model accuracy (both X- and Y-coordinate units had to match their targets for location prediction to be considered accurate). Location prediction was always one unit time interval into the future.

Only weights on connections into readout units were trained. Levels of synaptic potentiation on connections into reservoir units were set by the critical branching tuning (given the diamond input patterns), and were not effected by readout units' error signals. Thus, task-specific performance of readout units was based on generic, task-neutral spiking dynamics tuned near their critical branching point.

Simulations

Each simulation consisted of a set of tuning trials, and a set of trials used for both training and testing. Each trial began with the diamond-shaped input pattern initialized at a random location on the input grid (with periodic boundary conditions, as shown in Figure 2). Each trial proceeded for 20 time intervals over which the diamond was moved for 20 increments, each one corresponding to approximately one unit on the input grid. At each increment, input units corresponding to the position of the diamond were induced to spike, and other input units did not spike for that time interval.

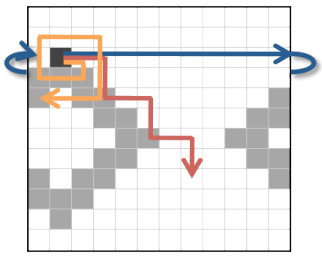

Figure 2: Diamond-shaped input pattern with periodic boundary conditions shown in gray. Arrows demonstrate straight-line (Simulation 1; blue), zig-zag (Simulation 2; red), and spiral (Simulation 3; orange) motion patterns

Reservoir dynamics were *not* reset after each trial; the model ran continuously as spikes were input to the network trial after trial. There were a total of 1,000 tuning trials, followed by 1,000 training and testing trials. The critical branching tuning algorithm was engaged only during tuning trials, and readout units were engaged only during training and testing trials. Each simulation was run five times (with parameters initialized anew each time), and mean performance over the five runs is presented. Reservoir units successfully approached their critical branching point by the end of tuning (mean estimated branching ratio of .88; for related results on critical branching, see Kello et al., 2011).

Simulation 1: Straight-Line Motion

For Simulation 1, movement of the diamond-shaped input pattern on each trial was in a straight line in one of four randomly chosen directions (up, down, left, right).

Results. Readout performance as a function of trial time interval is shown in Figure 3. At the beginning of each trial, results from both direction and location tasks are near chance performance because input spikes have not yet begun to perturb ongoing spiking activity. Performance on both tasks then gradually ramps up to asymptotic performance. This ramp-up shows that spiking dynamics, while not attractor-based, increasingly encode motion information as it consistently accumulates over successive inputs. The ramp-up in both results shows that the same, generic reservoir dynamics can be simultaneously tapped to compute different functions of motion.

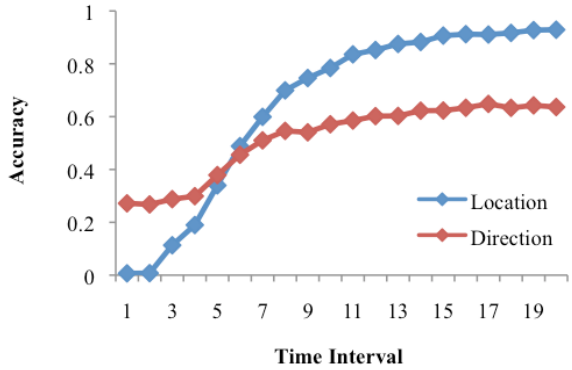

Figure 3: Accuracy on direction and location prediction tasks in Simulation 1

Figure 3 also shows that direction classification was more difficult than location prediction. Given that both functions essentially coded the same information (i.e. 1-of-4 different directions and 1-of-4 future locations given the current location), we can conclude that this difference arises in part because the direction task requires generalization over position. Additionally, fewer weights encoded the direction function (4 versus 24 readout units). Further work is needed to pull these two factors apart.

Simulation 2: Zig-Zag Motion

The straight-line motion task in Simulation 1 required minimal integration of past information to achieve accurate performance. That is, it was sufficient to "remember" where the diamond was for any two or more of the previous time intervals. However, motion processing can be more difficult, in that information about the sequence of several past inputs may be required for accurate performance. In Simulation 2, a zig-zag motion pattern was used as a more stringent test of the memory capacity of reservoir dynamics.

Each zig-zag trial began with the diamond-shaped input pattern initialized in a random start location. Motion again began in one of four randomly chosen directions (up, down, left, right), but here the direction of motion alternated by right angles at regular intervals to create a zig-zag pattern. The direction of the initial 90˚ turn was chosen randomly at the onset of each trial, and motion alternated between this and the original direction for the duration of the trial. The simulation was performed with inter-turn intervals (zig-zag lengths) ranging from 2 to 5.

In order for readout units to achieve accurate performance, reservoir dynamics must now encode more specific information about inputs in the past. In particular,

reservoir dynamics must encode the time and direction of the previous zig-zag change, which may have occurred several time intervals in the past.

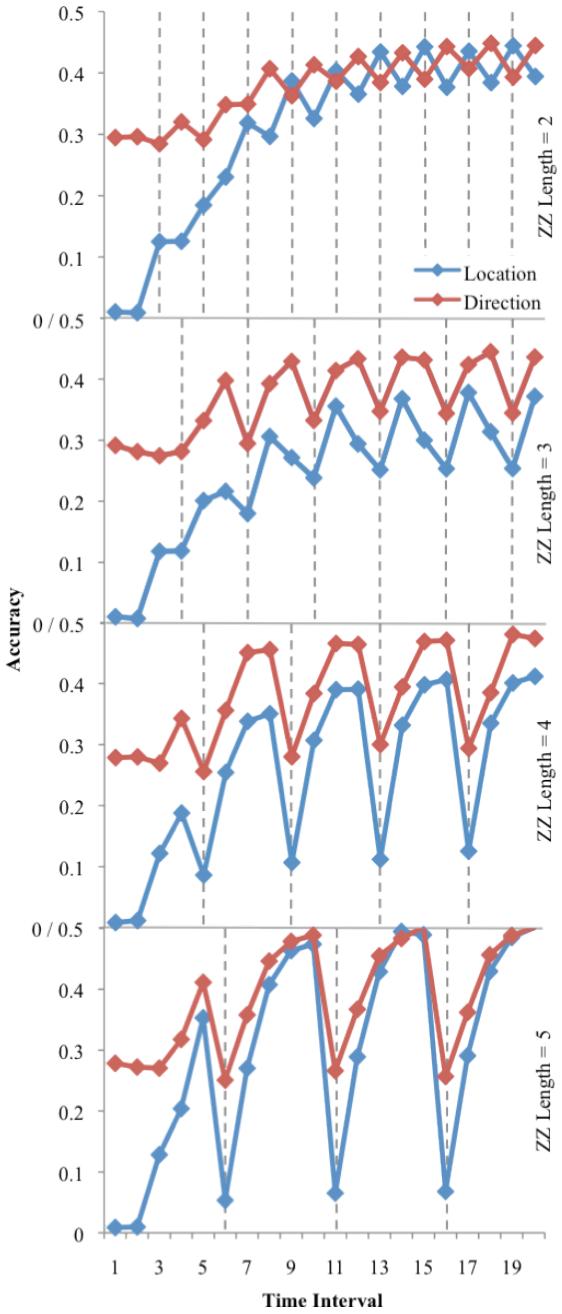

Figure 4. Accuracy for direction and location prediction in Simulation 2 (dashed lines show changes in direction)

Results. Results are shown in Figure 4. Performance in Simulation 2 was worse overall than in Simulation 1. This difference reflects the need for weights into readout units to extract more conjunctive information about past inputs. Notably, performance followed a zig-zag pattern aligned with the zig-zag of motion (direction changes shown by dotted lines). This pattern indicates that, as expected, predicting changes in direction was difficult because it required greater memory capacity. As motion continued linearly between each turn, performance gradually rose until it fell again at the next turn.

The most important result was the overall increase in performance, which is best seen in zig-zag lengths 2 and 3. This trend indicates that reservoir dynamics accumulated information about the zig-zag pattern itself as inputs came in over time, and were more capable of anticipating the turns. This trend is rather small for length 4, and mostly absent from length 5. These results show limitations in the reservoir's memory capacity as the length of time increased between turns that must be remembered. This generic memory capacity increases with the size of the reservoir, so these limitations are not absolute. It is an open question whether there may be other means of increasing memory capacity, either generically or for task-specific purposes. Finally, one can also see from Figure 4 that the relative performance of the two tasks differs from that observed in Simulation 1, especially with shorter zig-zag lengths. This result requires further investigation, but it presumably has to do with the position-general nature of direction coding, versus position-specific nature of location prediction.

Simulation 3: Spiral Motion

The zig-zag motions in Simulation 2 minimally required memory about only the last turn. In Simulation 3, an expanding spiral motion pattern was used to test whether reservoir dynamics can encode memory about all prior turns. On each trial, motion again began in one of four randomly chosen directions, and turns were either 90 degrees clockwise or counter-clockwise, also chosen randomly. The sides of the spiral expanded over time in the sequence 1, 1, 2, 2, 3, 3, 4, and 4 (for a sum of 20 intervals; see Figure 2 for motion depiction). Simulation methods were otherwise the same.

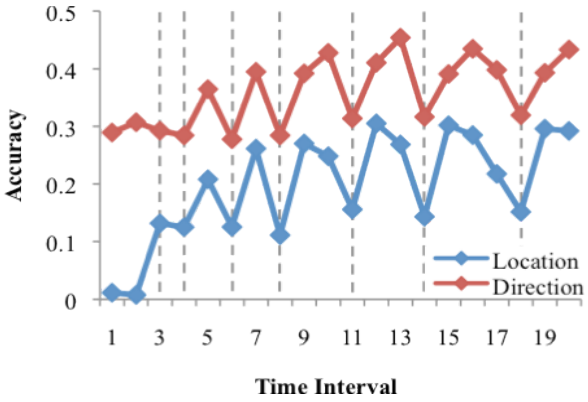

Figure 5. Accuracy on direction and location prediction in Simulation 3 (dashed lines show changes in direction)

Results. Results are shown in Figure 5. As expected, overall performance on the spiral motion simulation was even worse than on the zig-zag motion, and performance dipped at each turn. However, performance once again showed a

slight upward trend over the entire trial interval, indicating that reservoir dynamics contained information about more than just one previous change in direction.

Conclusion

The present results are a proof-of-concept that generic, recurrent spiking dynamics have memory for visual inputs that can be tapped for motion processing tasks. Spike dynamics were tuned to their critical branching point, which has been shown to enhance the memory and computational capacity of reservoir spiking networks (Kello & Mayberry, 2010). Furthermore, recordings from cortical slice preparations and EEG have provided evidence of critical branching neural activity (Poil et al., 2008). Thus, critical branching may be a basic principle utilized by motion processing microcircuits, as well as neural networks in general. Model results did not reach the level of accuracy necessary for simulating human motion processing capabilities, but the pattern of results showed that critical branching spiking dynamics are capable of the kinds of motion processing demonstrated in human behavior.

Behavioral research has shown that an object's motion is an important cue for object recognition (Newell et al., 2004), and researchers are currently investigating the idea that the temporal contiguity of an object's image on the retina contributes to the development of invariant object recognition (Li & DiCarlo, 2008). Because the recurrent dynamics of the model described here are shown to be capable of integrating information from multiple time steps, future work will investigate whether this memory can be tapped to form invariant object representations.

Acknowledgements

This work was funded by a DARPA contract to IBM, and a University of California, Merced, Faculty Mentor Program Fellowship award. The authors would like to thank reviewers for helpful comments and suggestions.

References

Bertschinger, N., & Natschläger, T. (2004). Real-time computation at the edge of chaos in recurrent neural networks. *Neural Computation, 16(7),* 1413-1436.

Braddick, O. (1997). Local and global representations of velocity: Transparency, opponency, and global direction perception. *Perception, 26,* 995-1010.

Burgsteiner, H., Kröll, M., Leopold, A., & Steinbauer, G. (2006). Movement prediction from real-world images using a liquid state machine. *Applied Intelligence, 26(2),* 99-109.

Buzsáki, G. (2006). *Rhythms of the Brain.* New York: Oxford University Press.

Carpenter, R. H. S. (1988). *Movements of the eyes*, 2nd ed. London: Pion Press.

Jaeger, H. (2001). *The "echo state" approach to analyzing and training recurrent neural networks.* GMD Report 148, GMD – German National Research Institute for Computer Science.

Kello, C. T., & Kerster, B. (2011). Power laws, memory capacity, and self-tuned critical branching in an LIF model with binary synapses. *Proceedings of Computational and Systems Neuroscience 2011.* Salt Lake City, UT.

Kello, C. T., Kerster, B., & Johnson, E. (2011). Critical branching neural computation, neural avalanches, and 1/f scaling. To appear in *Proceedings of the 33rd Annual Cognitive Science Society Conference.* Boston, MA.

Kello, C. T., & Mayberry, M. (2010). Critical branching neural computation. *IEEE World Congress on Computational Intelligence* (pp. 1475-1481). Barcelona, Spain.

Land, M. F., & McLeod, P. (2000). From eye movements to actions: How batsmen hit the ball. *Nature Neuroscience, 3,* 1340-1345.

Li, N., & DiCarlo, J. (2008). Unsupervised natural experience rapidly alters invariant object representations in visual cortex. *Science, 321,* 1502-1507.

Maass, W., Legenstein, R., & Markram, H. (2002a). A new approach towards vision suggested by biologically realistic neural microcircuit models. *Lecture Notes in Computer Science, 2525/2002*, 282-293.

Maass, W., Natschläger, T., & Markram, H. (2002b). Real-time computing without stable states: A new framework for neural computation based on perturbations. *Neural Computation, 14(11),* 2531-2560.

McBeath, M. K. (1990). The rising fastball: Baseball's impossible pitch. *Perception, 19,* 545-552.

Newell, F. N., Wallraven, C., & Huber, S. (2004). The role of characteristic motion in object categorization. *Journal of Vision, 4(2),* 118-129.

Petersen, C. C. H., Malenka, R. C., Nicoll, R. A., & Hopfield, J. J. (1998). All-or-none potentiation at CA3-CA1 synapses. *Proceedgins of the National Academy of Sciences of the United States of America, 95,* 4732-4737.

Poil, S.-S., van Ooyen, A., & Linkenkaer-Hansen, K. (2008). Avalanche dynamics of human brain oscillations: Relation to critical branching processes and temporal correlations. *Human Brain Mapping, 29*, 770-777.

Srinivasan, M. V. (2000). Visual navigation: The eyes know where their owner is going. In J. M. Zanker & J. Zeil (Eds.), *Motion vision: Computational, neural, and ecological constraints.* Berlin: Springer.

Turrigiano, G. G. (2008). The self-tuning neuron: Synaptic Scaling of Excitatory Synapses. *Cell, 135,* 422-435.

Vaney, D. I., He, S., Taylor, W. R., Levick, W. R. (2000). Direction-selective ganglion cells in the retina. In J. M. Zanker & J. Zeil (Eds.), *Motion vision: Computational, neural, and ecological constraints.* Berlin: Springer.

Verstraeten, D., Schrauwen, B., Stroobandt, D., & Van Campenhout J. (2005). Isolated word recognition with the *Liquid State Machine*: A case study. *Information Processing Letters, 95,* 521-528.