Efficient spiking neural network model of pattern motion selectivity in visual cortex

*Michael Beyeler[1], Micah Richert[2,3], Nikil D. Dutt[1], and Jeffrey L. Krichmar[1,2]
[1]Department of Computer Science, University of California Irvine, Irvine, CA, USA
[2]Department of Cognitive Sciences, University of California Irvine, Irvine, CA, USA
[3]Brain Corporation, San Diego, CA, USA

Keywords:
pattern motion selectivity, spiking neural network, MT, GPU, real-time, CARLsim

Correspondence:
Michael Beyeler
University of California, Irvine
Department of Computer Science
Irvine, CA 92697
mbeyeler@uci.edu
Office of Jeff Krichmar: (949) 824-5888

Abstract

Simulating large-scale models of biological motion perception is challenging, due to the required memory to store the network structure and the computational power needed to quickly solve the neuronal dynamics. A low-cost yet high-performance approach to simulating large-scale neural network models in real-time is to leverage the parallel processing capability of graphics processing units (GPUs). Based on this approach, we present a two-stage model of visual area MT that we believe to be the first large-scale spiking network to demonstrate pattern direction selectivity. In this model, component-direction-selective (CDS) cells in MT linearly combine inputs from V1 cells that have spatiotemporal receptive fields according to the motion energy model of Simoncelli and Heeger. Pattern-direction-selective (PDS) cells in MT are constructed by pooling over MT CDS cells with a wide range of preferred directions. Responses of our model neurons are comparable to electrophysiological results for grating and plaid stimuli as well as speed tuning. The behavioral response of the network in a motion discrimination task is in agreement with psychophysical data. Moreover, our implementation outperforms a previous implementation of the motion energy model by orders of magnitude in terms of computational speed and memory usage. The full network, which comprises 153,216 neurons and approximately 40 million synapses, processes 20 frames per second of a 40×40 input video in real-time using a single off-the-shelf GPU. To promote the use of this algorithm among neuroscientists and computer vision researchers, the source code for the simulator, the network, and analysis scripts are publicly available.

1. Introduction

Visual motion perception is a challenging problem that is critical for navigating through the environment and tracking objects. Several software packages are available to the public that deal with the neurobiologically plausible modeling of motion perception in the mammalian brain, such as spatiotemporal-energy models like the motion energy model of Simoncelli and Heeger (1998), or gradient-based models like ViSTARS (Browning et al. 2009a, 2009b). However, in order for these frameworks to become practical in, for example, neuromorphic or robotics applications, they must be capable of running large-scale networks in real-time. Moreover, to take advantage of state-of-the-art neuromorphic hardware, the elements of the algorithms need to be spiking neurons (Indiveri et al. 2006; Merolla et al. 2007; Vogelstein et al. 2007; Khan et al. 2008; Srinivasa and Cruz-Albrecht 2012). Developing such a simulation environment is challenging, due to the required memory to store the network structure and the computational power needed to quickly solve the equations describing the neuronal dynamics. A low-cost yet high-performance approach to simulating large-scale spiking neural networks (SNNs) in real-time is to leverage the parallel processing capability of graphics processing units (GPUs) (Nageswaran et al. 2009; Fidjeland and Shanahan 2010; Yudanov et al. 2010; Richert et al. 2011).

Based on this approach, we present a two-stage model of visual area MT that we believe to be the first large-scale spiking network to demonstrate pattern direction selectivity. The model combines and extends two previous incarnations of the motion energy model (Simoncelli and Heeger 1998; Rust et al. 2006). Broadly speaking, our model integrates the V1 stage of Simoncelli and Heeger (1998) with the MT stage of Rust et al. (2006) in the spiking domain. More precisely, our model uses a bank of spatiotemporal filters (Adelson and Bergen 1985; Simoncelli and Heeger 1998) to model the receptive fields of directionally selective neurons in V1, which then project to component-direction-selective (CDS) cells in area MT. However, the local motion estimates coded by the spike patterns of these neurons often vary drastically from the global pattern motion of a visual stimulus, because the local motion of a contour is intrinsically ambiguous (“aperture problem”). Therefore, in order to construct pattern-direction-selective (PDS) cells in MT that signal the global pattern motion, we implemented three design principles introduced by Rust et al. (2006): 1) spatial pooling over V1 or MT CDS cells with a wide range of preferred directions, 2) strong motion opponent suppression, and 3) a tuned normalization that may reflect center-surround interactions in MT. Whereas the implementation by Rust et al. (2006) was restricted to inputs that are mixtures of sinusoidal gratings of a fixed spatial and temporal frequency, our model can operate on any spatiotemporal image intensity.

The motion energy model of Simoncelli and Heeger (1998), henceforth referred to as the S&H model, is conceptually equivalent to an elaborated Reichardt detector at the end of the V1 stage (van Santen and Sperling 1985), and is a specific implementation of the intersection-of-constraints (IOC) principle at the end of the MT PDS stage (Bradley and Goyal 2008). The IOC principle in turn is one possible solution to the aperture problem; that is, a velocity-space construction that finds the global pattern motion as the point in velocity-space where the constraint lines of all local velocity samples intersect. Adelson and Movshon (1982) differentiated among three methods to estimate the global pattern motion; 1) IOC principle, 2) vector average (VA), and 3) blob or feature tracking, which may be equally valid approaches to solving the aperture problem (for a recent review on the topic see Bradley and Goyal (2008)). Although the S&H model is not complete, in the sense that it does not specify the exact pattern or object velocity, the model in particular and the IOC principle in general are consistent with various experimental data.

In the present paper, we introduce a large-scale spiking neuron model of cortical areas critical for motion processing, which is efficient enough to run in real-time on available processors. We show that the responses of neurons in the network are comparable to electrophysiological results for grating and plaid stimuli, as well as speed tuning. The behavioral response of the network in a two-alternative forced choice (2AFC) motion discrimination task (that is, a random dot motion coherence task) is in agreement with psychophysical data. Moreover, our implementation outperforms a previous rate-based C/Matlab implementation of the S&H model by up to a factor of 12 in terms of computational speed and by orders of magnitude in terms of memory usage. The full network, which comprises 153,216 neurons and approximately 40 million synapses, processes 20 frames per second of a 40×40 input video in real-time using a single off-the-shelf GPU.

The network was constructed using an open-source SNN simulator (Richert et al. 2011) that provides a PyNN-like programming interface; its neuron model, synapse model, and address-event representation (AER) are compatible with recent neuromorphic hardware (Srinivasa and Cruz-Albrecht 2012). To promote the use of this algorithm among the neuroscientist and computer vision research communities, the source code for the simulator, the network, and analysis scripts are publicly available at http://www.socsci.uci.edu/~jkrichma/CARLsim/.

2. Methods

2.1 The simulator

The present model was developed on a simulator that was previously published in Nageswaran et al. (2009) and Richert et al. (2011). The first study demonstrated real-time performance for a simulation of 100,000 neurons on a single NVIDIA C1060 GPU. The latter added a wide range of functionalities, such as equations for synaptic conductances, spike-timing-dependent plasticity (STDP), and short-term plasticity (STP). The present release builds on this mainly by: 1) providing the complete source code for a detailed large-scale model of visual motion processing in V1 and MT, 2) improving the original model to demonstrate PDS responses and speed tuning, and 3) introducing source code-level optimizations that improve GPU memory management and ensure code stability. Whereas the optimizations should be applicable to a wide range of GPU architectures, they are not directly relevant to this paper and will thus not be discussed (for more information please refer to the release notes).

The main code to run the experiments described in this paper can be found in the file "examples/v1MTLIP/main_v1MTLIP.cpp", which is part of the CARLsim 2.1 software package. The "examples" directory also contains a number of other experiments that were part of a previous code release—for more information refer to Richert et al. (2011). Matlab scripts to analyze the network output and create the figures can be found in the directory "scripts/v1MTLIP/". Please note that Matlab is not necessary to use the simulator, as the scripts are provided mainly for analysis purposes.

2.1.1 Setting up a simulation

Step-by-step instructions on how to set up, interact with, and run a simulation can be found in the tutorial on our website and in our previous code release (Richert et al. 2011). For the reader’s convenience, we include here a representative example to illustrate the ease of setting up and running a simulation. Listing 1 randomly connects ten Poisson spike generators (gIn) firing at 50 Hz mean rate to a population of 100 excitatory Izhikevich neurons (gEx), records and stores the spike times in a binary file "spkEx.dat", and runs the network for a second of simulation time:

```
#include "snn.h"
CpuSNN sim("My network");

// set up network
int gIn=sim.createSpikeGeneratorGroup("input", 10, EXCITATORY_NEURON);
int gEx=sim.createGroup("excitatory", 100, EXCITATORY_NEURON);
sim.setNeuronParameters(gEx, 0.02f, 0.2f, -65.0f, 8.0f); // RS neurons sim.connect(gIn, gEx,
"random", 1.0, 1.0, 0.10f, 1, 20, SYN_FIXED);

// write spike times to file
sim.setSpikeMonitor(gEx, "spkEx.dat");

// set spike rates and run network
PoissonRate inSpikes(100);
for (int i=0; i<100; i++)
  inSpikes.rates[i] = 50.0f; // 50 Hz
sim.setSpikeRate(gIn, &inSpikes);
sim.runNetwork(1,0); // run for 1 sec and 0 msec
```

Listing 1

In this example, connectivity (achieved through CpuSNN:connect(...)) is random with an initial weight of 1.0, a maximum weight of 1.0, a 10 % (0.10) connection probability, a synaptic delay uniformly distributed between 1 ms and 20 ms, and static synapses (SYN_FIXED). Note that any type of connectivity profile is possible by using a callback mechanism. For a description of the Izhikevich neuron model please refer to Section 2.1.3.

2.1.2 CPU vs. GPU simulation mode

A major advantage of our simulator is the possibility to run a simulation either on standard x86 central processing units (CPUs) or off-the-shelf NVIDIA GPUs, simply by passing a constant with value CPU_MODE or GPU_MODE as an additional function argument to CpuSNN::runNetwork(...). A new feature is the option to pass a “device index” to the same method, which can be used in multi-GPU systems to specify on which CUDA device to establish a context. For example, Listing 2 would run a built network for one second on the second GPU (if such a device exists):

```
CpuSNN sim(“My network”);
... // build network
int run_sec = 1; int run_msec = 0; // run for 1 s and 0 ms
bool onGPU = true; // run on GPU
int ithGPU = 1; // run on 2nd device (0-indexed)
sim.runNetwork(run_sec, run_msec, onGPU?GPU_MODE:CPU_MODE, ithGPU);
```

Listing 2

The two simulation modes allow the user to exploit the advantages of both architectures. Whereas the CPU is more efficient for relatively small networks, the GPU is most advantageous for network sizes of 1,000 neurons and up (Nageswaran et al. 2009; Richert et al. 2011). It has been demonstrated that a GPU implementation (on NVIDIA GTX-280 with 1 GB of memory) for a simulation of 100,000 neurons and 50 million synaptic connections can run up to 26 times faster than a CPU version (Core2 4600 @ 2.13 GHz with 4 GB of memory) of the same network (Nageswaran et al. 2009). On the other hand, the CPU mode allows for execution of extremely large networks that would not fit within the GPU’s memory.

It is worth noting that a simulation can be run in CPU mode even if the code is compiled in the presence of CUDA source files. An example of this hybrid mode is the network explained in the present work, which contains a V1 stage purely written in CUDA. In this case the network would be allocated on the CPU’s memory, but the generation of motion energy responses would be delegated to the GPU.

2.1.3 Neuron model

The simulator currently supports four parameter Izhikevich point-neurons (Izhikevich 2003). Other neuron models will follow in future releases. The Izhikevich model aims to reduce Hodgkin-Huxley-type neuronal models to a two-dimensional system of ordinary differential equations,

$$\frac{dv(t)}{dt}=0.04v^2(t)+5v(t)+140-u(t)+i_{syn}(t) \tag{1}$$

$$\frac{du(t)}{dt}=a(bv(t)-u(t)). \tag{2}$$

Here (1) describes the membrane potential v for a given external current i_{syn}, whereas (2) describes a recovery variable$u$; the parameter a is the rate constant of the recovery variable, and the parameter *b* describes the sensitivity of the recovery variable to the subthreshold fluctuations of the membrane potential. All parameters in (1) and (2) are dimensionless; however, the right-hand side of (1) is in a form such that the membrane potential *v* has mV scale and the time *t* has ms scale (Izhikevich 2003). The Izhikevich model is well-suited for large-scale simulations, because it is computationally inexpensive yet capable of spiking, bursting, and being either an integrator or a resonator (Izhikevich 2004, 2007).

In contrast to other simple models such as the leaky integrate-and-fire (LIF) neuron, the Izhikevich neuron is able to generate the upstroke of the spike itself. Thus the voltage reset occurs not at the threshold, but at the peak (v_{cutoff}=+30), of the spike. The action potential downstroke is modeled using an instantaneous reset of the membrane potential whenever *v* reaches the spike cutoff, plus a stepping of the recovery variable:

$$v(v>30)=c \quad \text{and} \quad u(v>30)=u-d. \tag{3}$$

The inclusion of *u* in the model allows for the simulation of typical spike patterns observed in biological neurons. The four parameters*a*, *b*, *c*, and *d* can be set to simulate different types of neurons. Unless otherwise specified, excitatory neurons in all our simulations were modeled as regular spiking (RS) neurons (class 1 excitable, $a=0.02, b=0.2, c=-65, d=8$), and all inhibitory neurons were modeled as fast spiking (FS) neurons (class 2 excitable, $a=0.1, b=0.2, c=-65, d=2$) (Izhikevich 2003, 2004).

2.1.4 Synapse model

A simulation can be run with either a current-based or a conductance-based neuron model (sometimes referred to as CUBA and COBA, respectively). All experiments in the present study were run in COBA mode.

In a conductance-based model, each ionic current that contributes to the total current i_{syn} (see (1)) is associated with a conductance. The simulator supports four of the most prominent synaptic conductances found in the cortex: AMPA (fast decay), NMDA (slow decay and voltage-dependent), GABAa (fast decay), and GABAb (slow decay), which are modeled as dynamic synaptic channels with zero rise time and exponential decay according to

$$\frac{d\,g_r(t)}{dt}=\frac{-1}{\tau_r}g_r(t)+w\sum_i \delta(t-t_i), \qquad (4)$$

where δ is the Dirac delta, the sum is over all presynaptic spikes arriving at times t_i, w is the weight of that synapse, τ_r is its decay time constant, and the subscript r denotes the receptor type; that is, AMPA, NMDA, GABAa, or GABAb. Unless otherwise specified, a spike arriving at a synapse that is post-synaptically connected to an excitatory (inhibitory) neuron increases both g_{AMPA} and g_{NMDA} (g_{GABA_a} and g_{GABA_b}¿. In our simulations we set the time constants to $\tau_{AMPA}=5$ ms, $\tau_{NMDA}=$ 150 ms, $\tau_{GABA_a}=6$ ms, and $\tau_{GABA_b}=150$ ms (Dayan and Abbott 2001; Izhikevich et al. 2004). The rise time of these conductances was modeled as instantaneous, which is a reasonable assumption in the case of AMPA, NMDA, and GABAa (Dayan and Abbott 2001), but a simplification in the case of GABAb, which has a rise time on the order of 10 ms (Koch 1999).

Then the total synaptic current i_{syn} in (1) for each neuron is given by:

$$i_{syn}=-g_{AMPA}(v-0)-g_{NMDA}\frac{\left(\frac{v+80}{60}\right)^2 ¿}{1+\left(\frac{v+80}{60}\right)^2}(v-0)¿ \\ -g_{GABAa}(v+70)-g_{GABA_b}(v+90), \qquad (5)$$

where v is the membrane potential of the neuron, and the subscript indicates the receptor type. This equation is equivalent to the one described in Izhikevich et al. (2004).

2.2 The network

The network architecture is shown in Fig. 1. Grayscale videos are fed frame-by-frame through a model of the primary visual cortex (V1), the middle temporal area (MT), and the lateral intraparietal cortex (LIP). Bold black arrows indicate synaptic projections. Note that inhibitory populations and projections are not shown for the sake of clarity. Numbers in parentheses next to an element are the equations that describe the corresponding neuronal response or synaptic projections, as will be explained in the subsections below.

The V1 model consisted of a bank of spatiotemporal filters (rate-based) according to the S&H model (Simoncelli and Heeger 1998), which will be described in detail in Section 2.2.1. At each point in time, a 32×32 input video frame was processed by V1 cells at three different spatiotemporal resolutions (labeled “3 scales” in Fig. 1). Simulated V1 simple cells computed an inner product of the image contrast with one of 28 space-time oriented receptive fields (third derivatives of a Gaussian), which was then half-wave rectified, squared, and normalized within a large Gaussian envelope. V1 complex cell responses were computed as a weighted sum of simple cell afferents that had the same space-time orientation, but were distributed over a local spatial region. We interpreted these filter responses as mean firing rates of Poisson spike trains (labeled “Hz” in the figure) as explained in Section 2.2.1, which were first scaled to match the contrast sensitivity function of V1 simple cells, and then used to drive Izhikevich spiking neurons representing cells in area MT.

Area MT consisted of two distinct populations of spiking neurons (explained in Section 2.2.2), the first one being selective to all local component motions of a stimulus (CDS cells), and the other one responding to the global pattern motion (PDS cells). MT CDS cells responded to three different speeds (1.5 pixels/frame, 0.125 pixels/frame, and 9 pixels/frame) illustrated as three distinct populations in the MT CDS layer of Fig. 1. Divisive normalization between these populations enabled the generation of speed tuning curves that are in agreement with neurophysiological experiments (Rodman and Albright 1987). The three MT CDS populations consisted of eight subpopulations, each of which was not only selective to a particular speed but also to one of eight directions of motion, in 45 degree increments. PDS cells were constructed by 1) pooling over MT CDS cells with a wide range of preferred directions, 2) using strong motion opponent suppression, and 3) employing a tuned normalization that may reflect center-surround interactions in MT (Rust et al. 2006). PDS cells were selective to the same speed as their CDS afferents. For the purpose of this paper we only implemented PDS cells selective to a speed of 1.5 pixels/frame (see MT PDS layer in Fig. 1) to be used in a motion discrimination task. However, it is straightforward to implement PDS cells that are selective to another speed.

A layer of decision neurons (see Section 2.2.3) was responsible for integrating over time the direction-specific sensory information that is encoded by the responses of MT PDS cells. Analogous to the MT layer, the decision layer consisted of eight subpopulations, each of which received projections from a subpopulation of MT PDS cells selective to one of eight directions of motion. This information was then used to make a perceptual decision about the presented visual stimulus, such as determining the global drift direction of a field of random moving dots in a motion discrimination task (presented in Section 3.3). Fig. 1 exemplifies this situation by showing a snapshot of the network's response to a random dot kinematogram (RDK) where dots drift to the right at a speed of 1.5 pixels/frame. The subpopulation of decision neurons that is coding for rightward motion is activated the strongest. The temporal integration of sensory information might be performed in one of several parietal and frontal cortical regions in the macaque, such as LIP, where neurons have been found whose firing rate are predictive of the behavioral reaction time (RT) in a RDK task (Shadlen and Newsome 2001; Roitman and Shadlen 2002).

The following subsections will explain the model in detail.

2.2.1 Spatiotemporal-energy model of V1

The first (V1) stage of the S&H model was implemented and tested in a Compute Unified Device Architecture (CUDA) environment (Richert et al. 2011). This part of the model is equivalent to Eqs. 1–4 in Simoncelli and

Heeger (1998) and their subsequently released C/Matlab code, which can be obtained from: http://www.cns.nyu.edu/~lcv/MTmodel/. Unless otherwise stated, we used the same scaling factors and parameter values as in the S&H model.

A visual stimulus is represented as a light intensity distribution $I(x, y, t)$, that is, a function of two spatial dimensions (x, y) and time t. The stimulus was processed at three different spatiotemporal resolutions (or scales), r (labeled "3 scales" in Fig. 1). The first scale, $r=0$, was equivalent to processing at the original image (and time) resolution. The other two scales were achieved by successively blurring the image with a Gaussian kernel. The three stimuli $I_r(x, y, t)$ can thus be expressed as:

$$\begin{aligned} I_0(x, y, t) &= I(x, y, t) \\ I_1(x, y, t) &= \exp\left(\frac{-\left(x^2+y^2+t^2\right)}{2}\right) * I_0(x, y, t) \\ I_2(x, y, t) &= \exp\left(\frac{-\left(x^2+y^2+t^2\right)}{2}\right) * I_1(x, y, t), \end{aligned} \tag{6}$$

where ¿ denotes convolution. In order to circumvent the non-causality of these convolutions (the response depends both on past and future stimulus intensities), a time delay of four frames was introduced (see (Simoncelli and Heeger 1998)).

V1 simple cells. A large body of research has found that neurons located in V1 that project to MT are directionally selective and may be regarded as local motion energy filters (Adelson and Bergen 1985; DeAngelis et al. 1993; Movshon and Newsome 1996). In our network, V1 simple cells are modeled as linear space-time-oriented filters whose receptive fields are third derivatives of a Gaussian (Simoncelli and Heeger 1998). These filters are very similar to a Gabor filter, but more computationally convenient as they allow for separable convolution computations.

The full set of V1 linear receptive fields consisted of 28 space-time orientations that are evenly distributed on the surface of a sphere in the spatiotemporal frequency domain. The kth space-time-oriented filter in the V1 population can be described by a unit vector $\hat{u}_k = (\hat{u}_{k,x}, \hat{u}_{k,y}, \hat{u}_{k,t})'$ that is parallel to the filter orientation, where $k = 1, 2, \ldots, 28$ and ' denotes vector transposition. For more information please refer to Simoncelli and Heeger (1998). An example of a spatiotemporal receptive field is illustrated in Fig. 2, where the colored ovals correspond to the orientation of the positive (green) and negative (red) lobes of the spatiotemporal filter. If a drifting dot traces out a path (dashed line) in space (x, for now ignoring y) and time (t) that is oriented in the same way as the lobes, then the filter could be activated by this motion (Fig. 2a). A dot moving in the orthogonal direction would not elicit a filter response because its path intersects both positive and negative lobes of the filter (as depicted in Fig. 2b).

First, input images were filtered with a 3D Gaussian corresponding to the receptive field size of a V1 simple cell:

$$f_r(x, y, t) = \exp\left(\frac{-(x^2 + y^2 + t^2)}{2\sigma^2_{v1simple}}\right) * I_r(x, y, t) \qquad (7)$$

where ¿ is the convolution operator, r denotes the scale, and $\sigma_{v1simple} = 1.25$ pixels.

Then the underlying linear response of a simple cell at spatial location (x, y) and scale r with space-time orientation k is equivalent to the third-order derivative in the direction of $\hat{u}_k$; that is,

$$L_{kr}(x, y, t) = \alpha_{v1lin} \sum_{T=0}^{3} \left[\sum_{Y=0}^{3-T} \left[\frac{3!}{X!Y!T!} (\hat{u}_{k,x})^X (\hat{u}_{k,y})^Y (\hat{u}_{k,t})^T \frac{\partial^3 f_r(x, y, t)}{\partial x^X \partial y^Y \partial t^T} \right] \right] \qquad (8)$$

where $!$ denotes the factorial, $X = 3 - Y - T$, and $\alpha_{v1lin} = 6.6084$ is a scaling factor. Note that the two sums combined yield exactly 28 summands. This operation is equivalent to Eq. 2 in the original paper, and can also be expressed using vector notation:

$$L_r = \alpha_{v1lin} M b_r, \qquad (9)$$

where L_r is the set of all V1 responses at scale r, each element of b_r is one of the separable derivatives in (8) at scale r, and each element of the 28×28 matrix M is a number $3!/(X!Y!T!)(\hat{u}_{k,x})^X(\hat{u}_{k,y})^Y(\hat{u}_{k,t})^T$. Each row of M has a different value for k, and each column of M has different values for X, Y, and T. We will make use of this notation in Section 2.2.2, where we will explain the construction of synaptic projections from V1 to MT.

At this stage of the model it is possible that filter responses L_{kr} at positions (x, y) close to the image border have become unreasonably large. We suppressed these edge effects by applying a scaling factor to L_{kr} whenever (x, y) was near an image border.

Simple cell responses were constructed by half-squaring and normalizing the linear responses L_{kr} from (8) within a large Gaussian envelope:

$$S_{kr}(x,y,t) = \frac{\alpha_{filt \to rate,r}\,\alpha_{v1rect}\lfloor L_{kr}(x,y,t)\rfloor^2}{\alpha_{v1norm}\exp\left(\frac{-(x^2+y^2)}{2\sigma_{v1norm}^2}\right) * \left(\frac{1}{28}\sum_{k=1}^{28}\lfloor L_{kr}(x,y,t)\rfloor^2\right) + \alpha_{v1semi}^2}, \qquad (10)$$

where $\lfloor . \rfloor$ denotes half-wave rectification, and ¿ is the convolution operator. The scaling factors $\alpha_{v1rect}=1.9263$ and $\alpha_{v1semi}=0.1$ (the semi-saturation constant) had the same values as in the original S&H model. Instead of having a single global normalization, our normalization occurs within a large spatial neighborhood (Gaussian half-width $\sigma_{v1norm}=3.35$ pixels), which is thought to be more biologically realistic. Therefore the scaling factor $\alpha_{v1norm}=1.0$ had to be adjusted to compensate for the implementation difference. This was done simultaneously by setting $\alpha_{filt \to rate,r}=15\,Hz$, a scaling factor to map the unit-less filter responses at each scale r onto more meaningful mean firing rates, as will be explained below. In brief, we opted to reproduce the contrast sensitivity function reported for V1 cells projecting to MT (Movshon and Newsome 1996). Other than that, the computation in (10) is conceptually equivalent to Eqs. 3–4 in Simoncelli and Heeger (1998).

V1 complex cells. V1 complex cell responses were computed as local weighted averages of simple cell responses,

$$C_{kr}(x,y,t) = \alpha_{v1comp} \exp\left(\frac{-(x^2+y^2)}{2\sigma_{v1comp}^2}\right) * S_{kr}(x,y,t), \qquad (11)$$

where the half-width of the Gaussian was $\sigma_{v1comp}=1.6$, and $\alpha_{v1comp}=0.1$ is a scaling factor.

The responses $C_{kr}(x,y,t)$ described in (11) served as output of the CUDA implementation. These responses were interpreted as mean firing rates of Poisson spike generators, following the procedure described in the next subsection. V1 complex cells then projected to MT CDS cells as explained in Section 2.2.2.

Converting filter responses to firing rates. In order to find a meaningful mapping from unit-less filter responses to mean firing rates, we opted to reproduce the contrast sensitivity function reported for V1 cells projecting to MT (Movshon and Newsome 1996), which is shown in Fig. 3. The red line is the electrophysiological data adapted from Fig. 7 of Movshon and Newsome (1996), whereas the blue line is our simulated data. In order to arrive at this plot, we presented a drifting sinusoidal grating of varying contrast to V1 simple cells coding for scale $r=0$, and computed their mean response $\langle S_{k0} \rangle$ from (10) over a stimulation period of one second. The drifting grating had a spatial frequency of $\omega_{spat}=0.1205$ cycles/pixel and a temporal frequency of $\omega_{temp}=0.1808$ cycles/frame, which is equivalent to the one used in Section 3.1 for MT direction tuning. Because the grating was drifting to the right, we only looked at the subpopulation of V1 simple cells that responded maximally to this stimulus (which was true for $k=24$). The mean firing rate of neurons in this subpopulation, $\langle S_{24,0} \rangle$, was then averaged over all cells in the subpopulation and plotted in Fig. 3 (blue curve) for $\alpha_{v1norm}=1.0$ and

$\alpha_{filt \rightarrow rate,0} = 15\,Hz$. Vertical bars are the standard deviation on the population average. The scaling factor α_{v1norm} was gradually changed until the curvature of the blue graph approximated the curvature of the electrophysiological data. The scaling factor $\alpha_{filt \rightarrow rate,0}$ was then adjusted such that the simulated responses saturated at approximately 100 Hz.

In order to tune V1 simple cells at the other two scales, that is, S_{k1} and S_{k2} from (10), we used a RDK stimulus, which is depicted as the sample input in Fig. 1 and explained in detail in Section 3.3. We chose scaling factors that would give equal response magnitudes at all three scales in response to the RDK stimulus, which resulted in $\alpha_{filt \rightarrow rate,1} = 17\,Hz$ and $\alpha_{filt \rightarrow rect,2} = 11\,Hz$.

Because these filter response were transformed to mean firing rates, it was straight-forward to assign the responses $C_{kr}(x, y, t)$ described in (11) to mean firing rates of Poisson spike generators, which served as input to the spiking neurons in area MT. The exact mapping of V1 complex onto MT CDS cells is given in (12) (see Section 2.2.2).

2.2.2 Two-stage spiking model of MT

The two-stage model of MT is based on the idea that CDS cells represent an earlier stage of motion processing than PDS cells (Movshon et al. 1985; M. A. Smith et al. 2005). The present model is built on this idea, making MT CDS cells similar in terms of direction and speed tuning to the model V1 complex cells used by Simoncelli and Heeger (1998). In fact, it has been shown that MT cells exhibit speed tuning characteristics similar to V1 complex cells (Priebe et al. 2006), which has led to the suggestion that speed tuning in MT might be inherited from V1. Livingstone and Conway (2007) have shown that even some V1 simple cells are speed-tuned in macaque. Whereas CDS cells give responses whose selectivity is stable and consistent from the time they are first activated, PDS cells often respond with different and broader selectivity when first activated, sometimes even resembling CDS cells, and only over a time-course on the order of 100 ms do they establish pattern selectivity (M. A. Smith et al. 2005). At least in anesthetized monkeys, MT is believed to consist of roughly 40 % CDS cells, 25 % PDS cells, and 35 % unclassified cells (Movshon et al. 1985). However, in awake animals the situation might be more complicated (Pack et al. 2001).

All cells in MT were Izhikevich spiking neurons, whose membrane potential was thus described by a pair of coupled differential equations (see (1) and (2)).

Component-direction-selective cells. CDS cells are selective to a particular direction and speed of motion (an orientation in space-time). The name is an indication that these cells, when presented with a plaid stimulus consisting of two superimposed sine gratings, preferably respond to the motion of each grating (component) rather than the global motion pattern produced by the combination of the two gratings (Movshon et al. 1985).

MT CDS cells in our model responded preferentially to motion in one of eight different directions (in 45 degree increments) and three different speeds (1.5 pixels per frame, 0.125 pixels per frame, and 9 pixels per frame) at any pixel location. These values can be easily adjusted by running the Matlab script "scripts/v1MTLIP/projectV1toMT.m". The response properties of MT CDS cells were given by 1) a set of both excitatory and inhibitory interpolated weights (as explained next) coming from V1 complex cells (Simoncelli and Heeger 1998), and 2) projections from an inhibitory group of MT interneurons to account for response normalization.

Because the directional derivatives of a Gaussian are steerable (Freeman and Adelson 1991), the response of an arbitrarily oriented filter can be synthesized from a fixed bank of basis filters (the third derivatives of a Gaussian). Thus the projection weights from V1 complex cells to MT were interpolated as follows. Let $\hat{\alpha}=(\hat{\alpha}_x,\hat{\alpha}_y,\hat{\alpha}_t)'$ be the unit vector parallel to an arbitrary space-time orientation (direction and speed of motion), akin to the unit vectors $\hat{u}_k$ described in Section 2.2.1. Then we can write the third directional derivative in direction of $\hat{\alpha}$ analogously to (9) as:

$$\frac{\partial^3 f_r}{\partial \hat{\alpha}^3} = \left[v'(\hat{\alpha}) M^{-1} \right] b_r \text{¿} w_{\hat{\alpha}} b_r , \qquad (12)$$

where the matrix M and the vector b_r are the same as in (9), each element of the vector $v(\hat{\alpha})$ is a number $6!/(X!Y!T!)\hat{\alpha}_x^X \hat{\alpha}_y^Y \hat{\alpha}_t^T$ analogous to (8), and $'$ denotes vector transposition. The product $\left[v'(\hat{\alpha}) M^{-1} \right]$ thus is a set $w_{\hat{\alpha}} = (w_{\hat{\alpha},1}, \ldots, w_{\hat{\alpha},28})$ of interpolated weights, where the kth element of this vector, $w_{\widehat{\alpha},k}$, determined the strength of the projection from the kth V1 complex cell onto a MT CDS cell. The two cells were connected only if they were located at the same pixel location, (x, y). Speed tuning arose from the fact that $\hat{\alpha}$ corresponds to a specific direction and speed of motion. Thus, in order to achieve MT CDS cells tuned to different speeds, $\hat{\alpha}$ was the only parameter that needed to be adjusted (refer to the Matlab script mentioned above). A MT CDS cell received projections from V1 complex cells at all three spatiotemporal resolutions, r. Note that it is possible to construct a network with the same functionality by using only one spatiotemporal resolution, which has been shown in Simoncelli and Heeger’s own C/Matlab implementation. Using multiple spatiotemporal resolutions, however, makes the network more robust in responding to motion of different-sized objects.

Because the interpolated weights could assume both positive and negative values, it was necessary to relay the projections with negative weights to a population of inhibitory neurons. In this case (that is, if $w_{\hat{\alpha},k} < 0$), the weights in (12) are applied to excitatory projections from V1 complex cells to the MT inhibitory population (where $w_{\hat{\alpha},k,inh} = \left| w_{\hat{\alpha},k} \right|$), and the inhibitory population sends one-to-one connections back to the pool of MT CDS cells. Overall the interpolated weights are equivalent to the parameters p_{nm} in Eq. 5 of Simoncelli and Heeger (1998).

In order to model response normalization equivalent to the one in Eq. 6 of Simoncelli and Heeger (1998), we introduced another pool of inhibitory interneurons, which integrated the activity of all MT CDS cells within a large Gaussian neighborhood (across direction and speed), and projected back to all three pools of MT CDS cells with one-to-one connections. This response normalization is important to qualitatively reproduce the speed tuning curves (see Section 3.2).

Pattern-direction-selective cells. PDS cells differ from CDS cells in that they, when presented with a plaid stimulus consisting of two superimposed sine gratings, preferentially respond to the overall motion direction, not the individual components (Movshon et al. 1985). Because visual stimuli typically contain many oriented components, local motion measurements must be appropriately combined in order to sense the true global motion of the stimulus (aperture problem). Thus it has been suggested that PDS neurons reflect a higher-order computation that acts on V1 or MT CDS afferents (Movshon et al. 1985). MT PDS cells in our model received direct input from CDS cells, and thus conserved their speed and direction preferences.

Pooling over MT CDS cells and opponent suppression were implemented by pooling CDS responses across spatial position and across direction preference, such that the strength of a projection from a CDS cell selective to motion direction θ_{CDS} at location (x_{CDS}, y_{CDS}) to a PDS cell selective to motion direction θ_{PDS} at location (x_{PDS}, y_{PDS}) can be expressed as:

$$w_{CDS \to PDS} = \alpha_{CDS \to PDS} \cos(\Delta\theta) \exp\left(\frac{-\left((\Delta x)^2 + (\Delta y)^2 \right)}{2\sigma_{PDS,pool}^2} \right), \qquad (13)$$

where $\Delta\theta = \theta_{PDS} - \theta_{CDS}$, $\Delta x = x_{PDS} - x_{CDS}$, $\Delta y = y_{PDS} - y_{CDS}$, the half-width of the Gaussian neighborhood $\sigma_{PDS,pool} = 3$ pixels, and $\alpha_{CDS \to PDS}$ is a scaling factor. If the resulting weight was negative, due to $|\Delta\theta| > \frac{\pi}{2}$, the projection was relayed to a population of inhibitory interneurons. Following the reasoning of Rust et al. (2006), the pattern index of a MT cell can be reduced simply by sharpening the cosine tuning component in (13) (see third column of Fig. 6 in Rust et al. (2006)).

Tuned normalization was implemented by an inhibitory self-connection with a narrowly tuned Gaussian across direction (see second column of Fig. 6 in Rust et al. (2006)). Analogous to previous projections, this was implemented by relaying the inhibitory projection to a pool of inhibitory interneurons:

$$w_{PDS\rightarrow PDS,inh}=\exp\left(\frac{-(\Delta\theta)^2}{2\sigma^2_{PDS,tuned,dir}}\right)\exp\left(\frac{-\left((\Delta x)^2+(\Delta y)^2\right)}{2\sigma^2_{PDS,tuned,loc}}\right), \qquad (14)$$

where $\sigma_{PDS,tuned,dir}<45$ deg (such that only one of the eight subpopulations was activated), $\sigma_{PDS,tuned,loc}=2$ pixels, and the inhibitory population sent one-to-one connections back to the pool of MT PDS cells.

2.2.3 Spiking layer of LIP decision neurons

A layer of decision neurons was responsible for integrating over time the direction-specific sensory information that is encoded by the responses of MT PDS cells. This information was then used to make a perceptual decision about the presented visual stimulus, such as determining the global drift direction of a field of random moving dots in a motion discrimination task (presented in Section 3.3). A good candidate for such an integrator area in macaques might be LIP, where neurons have been found whose firing rate are predictive of the behavioral reaction time (RT) in a motion discrimination task (Shadlen and Newsome 2001; Roitman and Shadlen 2002).

Spiking neurons in a simulated LIP area were grouped into eight pools of 50 neurons, each pool receiving projections from exactly one of the eight pools of MT PDS cells with 10 % connection probability. As a result of this connectivity profile, each pool of decision neurons accumulated sensory evidence for a particular direction of motion, based on the response of MT PDS cells.

Additionally, each decision pool received inhibitory projections from other decision pools if the two preferred directions of motion were close to opposite. More precisely, a decision neuron in pool *i* (thus selective to direction θ_i) received an inhibitory projection from neurons in pool j (selective to direction θ_j) with strength

$$w_{dec,inh \rightarrow dec} = \lfloor \cos\left(\theta_i - \theta_j + \pi\right) \rfloor, \qquad (15)$$

and 10 % connection probability.

LIP decision neurons did not employ any internal noise.

2.2.4 Implementation details

In order for our implementation to be useful to researchers already working with the S&H model, we tried to stay as close to the S&H C/Matlab implementation as possible. However, there are a few minor differences worth mentioning. First, as explained in Section 2.2.1, we normalize V1 simple cell responses in a large Gaussian neighborhood rather than across the whole population. Second, whereas the S&H model deals with edge effects by temporarily “padding” the input image with an invisible border, we opted for the computationally more economical alternative to simply decrease the responses of V1 simple cells located close to image borders. Third, in the S&H C/Matlab implementation there are two additional scaling factors (called `v1Blur` and `v1Complex`, with values 0.99 and 1.02, respectively) that we do not apply in order to save execution time. Fourth, our model processes input images at three different scales as described in (6), which is a feature that is not implemented in the original S&H model.

The most crucial mathematical operation in the V1 stage of the model is the convolution. Because the filter kernels used in our implementation are relatively small, employing the fast Fourier transform (FFT) would actually hurt performance. Instead we perform all convolution operations in the space-time domain using a custom function, which makes use of the fact that the Gaussian filter and its derivative are dimensionally separable. Future work could be directed towards further optimizing the convolution operation in CUDA.

3. Results

We conducted a number of experiments to ensure the accuracy and efficiency of our implementation. Here we demonstrate that the network is able to exhibit direction and speed tuning for drifting bar and plaid stimuli that are in agreement with neurophysiological recordings, and that the network qualitatively reproduces both the psychometric and chronometric function in a 2AFC motion discrimination task. Additionally, we measured both the computational performance and memory consumption of our model and compared it to the S&H C/Matlab implementation.

GPU simulations were run on a NVIDIA Tesla M2090 (6 GB of memory) using CUDA, and CPU simulations (including Matlab) were run on an Intel Xeon X5675 at 3.07 GHz (24 GB of RAM). The same exact network running on a single GPU produced all results; the only difference per experiment was the presented input stimulus. The full network consisted of 153,216 neurons and approximately 33 million synapses, which corresponds to a 32×32 pixels input resolution.

3.1 Direction tuning

We tested the ability of our model MT cells to signal the direction of motion for drifting grating and plaid stimuli. Responses were simulated for CDS cells and PDS cells in MT. The first stimulus was a drifting sinusoidal grating consisting of spatial and temporal frequency components that were preferred by MT neurons selective to a speed of 1.5 pixels per frame (that is, $\omega_{spat}=0.1205$ cycles/pixel, $\omega_{temp}=0.1808$ cycles/frame¿. The second stimulus was a pair of superimposed gratings drifting in a direction orthogonal to their orientation, which together formed a coherently drifting plaid pattern. The two gratings both had the same spatial frequency ω_{spat}, but their orientation and drift direction differed by 120 degrees. The direction of these particular patterns lay equidistant between the directions of motion of the two component gratings. The stimulus contrast for both grating and plaid was 30 %.

Our model was able to reproduce direction tuning curves that are in agreement with single-cell electrophysiological data (Movshon et al. 1985; Rodman and Albright 1989; Movshon and Newsome 1996) for V1 cells, MT CDS cells, and MT PDS cells. Fig. 4 shows polar plots of direction tuning for V1 neurons (Panels b and f), MT CDS cells (Panels c and g), and MT PDS cells (Panels d and h), where the angle denotes motion direction and the radius is the firing rate in spikes per second (compare also Fig. 9 in Simoncelli and Heeger (1998) and Fig. 1 in Rust et al. (2006)). Tuning curves were obtained by calculating the mean firing rate of a neuron's response to a drifting grating during two seconds of stimulus presentation. These responses were averaged over all neurons in the population selective to the same direction of motion (black: mean neuronal response, blue: mean plus standard deviation on the population average, green: mean minus standard deviation). As a result of suppressing edge effects, neurons that coded for locations closer than five pixels from the image border were only weakly activated, and were thus excluded from the plot. The tuning curves in the top row were generated in response to the sinusoidal grating drifting upwards, which is illustrated in Panel a. Analogously, the tuning curves in the bottom row were generated in response to the plaid stimulus drifting upwards, which is illustrated in Panel e (red arrow: pattern motion direction, black arrows: motion direction of the grating components). The direction tuning curve for gratings is unimodal for all three neuron classes, but the direction tuning curve for plaids shows two distinct lobes for V1 complex cells (Panel f) and MT CDS cells (Panel g). Each lobe corresponds to one of the component gratings of the plaid. Only MT PDS cells (Panel h) responded to the motion of the entire plaid pattern rather than to the motions of the individual component gratings.

In order to quantify the pattern selectivity of our model PDS cells, we computed the pattern index for each CDS and PDS cell (see Fig. 5) using the standard technique (Movshon et al. 1985; Movshon and Newsome 1996; M. A. Smith et al. 2005). Based on the tuning curve for the drifting grating described above, we generated two predictions for each cell's tuning curve to drifting plaids (Fig. 5a); either the cell would respond to the plaid in the same way as it responded to the grating ("pattern" prediction, black solid line), or it would respond independently to the two grating components ("component" prediction, black dashed line). We then computed the correlation (r_c, r_p) between the cell's actual response to a plaid stimulus and the component and pattern predictions. To remove the influence of correlations between the predictions themselves, we calculated partial correlations R_c and R_p for the component and pattern predictions, respectively, using the standard formulas:

$$R_c = \frac{r_c - r_p r_{pc}}{\sqrt{\left(1 - r_p^2\right)\left(1 - r_{pc}^2\right)}} R_p = \frac{r_p - r_c r_{pc}}{\sqrt{\left(1 - r_c^2\right)\left(1 - r_{pc}^2\right)}}, \qquad (16)$$

where r_c and r_p are the simple correlations between the data and the component and pattern predictions, respectively, and r_{pc} is the simple correlation between the predictions (Movshon and Newsome 1996). Because the sampling distribution of Pearson's r is not normal, we converted the correlation measures R_c and R_p to a Fisher Z-score,

$$Z_c = \frac{0.5 \ln\left(\frac{1+R_c}{1-R_c}\right)}{\sqrt{\frac{1}{df}}} = \frac{atanh\left(R_c\right)}{\sqrt{\frac{1}{df}}} Z_p = \frac{atanh\left(R_p\right)}{\sqrt{\frac{1}{df}}}, \qquad (17)$$

where the numerator is the Fisher r-to-Z transformation and df is the degrees of freedom, equal to the number of values in the tuning curve (in our case 24) minus three (M. A. Smith et al. 2005). The Z-scores of all CDS and PDS cells (excluding neurons coding for locations closer than five pixels from the image border) in the network are plotted in Fig. 5b. Each value of Z_c and Z_p was tested for significance using a criterion of 1.28, which is equivalent to $P = 0.90$ (M. A. Smith et al. 2005). For a PDS cell (red) to be judged as pattern-selective, the value of Z_p had to exceed the value of Z_c by a minimum of 1.28 (black solid lines). All PDS cells in Fig. 5b met this criterion and, therefore, were indeed pattern-selective. Analogously, all CDS cells (blue) could be judged as component-selective.

3.2 Speed tuning

We next considered the ability of our implementation to reproduce MT speed tuning curves as demonstrated in Simoncelli and Heeger (1998). MT neurons have been divided into three distinct classes based on their speed tuning properties (Rodman and Albright 1987). The first class of neurons is relatively sharply tuned for a particular speed and direction of motion ("speed-tuned" or "band-pass"). This class of neurons is also strongly suppressed by motion in the anti-preferred (opposite) direction; the suppression is strongest when the stimulus moves in the opposite direction at roughly the preferred speed. The second class of neurons prefers low speeds in both the preferred and anti-preferred direction ("low-pass"). The third class responds to high speed stimuli in both directions ("high-pass").

Fig. 6 faithfully reproduces the speed tuning characteristics of these three distinct classes (compare also Fig. 10 in Simoncelli and Heeger (1998)). The stimulus consisted of a single bar drifting over the entire visual field either to the right (preferred direction) or to the left (anti-preferred direction) at different speeds. Each data point is the mean firing rate of a particular MT CDS neuron located near the center of the visual field, averaged over the time course of a specific speed and direction configuration. The relatively low mean firing rates can be explained by the fact that the stimulus resides outside the neuron's receptive field for most of the time. The first neuron class (Panel a, "band-pass") preferentially responded to a bar moving at 1.5 pixels per frame to the right, and was strongly suppressed when the bar moved at the same speed to the left. The second neuron class (Panel b, "low-pass") exhibited a preference for low speeds (0.125 pixels per frame) in both directions. With increasing speed the response of the neuron to dots moving in the anti-preferred direction weakened. This behavior can be explained by the fact that the Fourier planes corresponding to low speed motions in opposite directions are both close to the $\omega_t = 0$ plane, and thus close to each other (Simoncelli and Heeger 1998). Also, this class of neurons was suppressed by fast stimuli moving in either direction. Similarly, the third neuron class (Panel c, “high-pass”), which had a high preferred speed (9 pixels per frame) in one direction, was excited by fast stimuli moving in the opposite direction, but was suppressed by slow stimuli moving in either direction.

3.3 Random dot kinematogram

In order to compare the performance of the model with behavioral data from 2AFC motion discrimination tasks, we developed a paradigm equivalent to the RDK experiments performed with monkeys and humans (Roitman and Shadlen 2002; Resulaj et al. 2009). We constructed a simple decision criterion based on the race model (Shadlen and Newsome 2001; P. L. Smith and Ratcliff 2004), in which eight pools of decision neurons (one for each of the directions of motion, 50 neurons per pool) sum the responses of MT PDS cells selective to a particular direction and speed of motion. The first decision pool to emit 500 spikes (on average ten spikes per neuron) “won the race” and thus signaled a choice for that direction. A correct decision was the event in which the winning decision pool was selective to the actual motion direction of the stimulus. The time it took the network to reach the decision threshold was termed the reaction time (RT).

The RDK stimulus was constructed out of approximately 150 dots (15 % dot density, maximum stimulus contrast) on a 32x32 input movie. An example frame is shown as the input stimulus in Fig. 1. Each stimulus frame was presented to the network for 50 ms. A trial consisted of 20 stimulus frames of a particular motion direction and coherence level. Motion coherence in the stimulus was varied between 0 and 50 %. Coherently moving dots drifted in one of eight possible directions, in 45 degree increments, at a speed of 1.5 pixels per frame. Note that, therefore, only MT PDS cells that were selective to this particular stimulus speed were connected to the decision layer.

Choice accuracy and RT as a function of task difficulty (coherence of dot motion) are shown in Fig. 7 (Panel a and b, respectively), where the thick red lines are human behavioral data extracted from a RT experiment (see Fig. 3 and Table 2 in Roitman and Shadlen (2002)) and simulated data is shown in blue. Each data point (blue) is the mean outcome of 80 trials (fixed coherence level, ten repetitions per motion direction), and the vertical bars are the standard error and standard deviation for accuracy (Panel a) and RT (Panel b), respectively. As in Fig. 3 in Roitman and Shadlen (2002), we did not show RTs on error trials.

Our network performance is comparable to human accuracy, and it qualitatively emulates the effect of motion strength on RT. Decreasing RT for a relatively easy task (e.g., high motion coherence) is a direct consequence of the race model. Conversely, when the difficulty of a decision is high (e.g., low coherence level), information favoring a particular response grows more slowly (P. L. Smith and Ratcliff 2004), and the probability of making an error is higher (Shadlen and Newsome 2001). The quantitative difference between behavioral and simulated RT in Fig. 7 could be eradicated by fine-tuning the excitatory weights from MT cells to the decision layer. However, such an exercise would be meaningless, because our model does not take into consideration neural areas involved in characteristics of the decision-making process that influence the length of RT, such as the time-course of LIP neuronal dynamics or the gating of saccadic eye movements (Shadlen and Newsome 2001), which have been successfully modeled in detail by others (Grossberg and Pilly 2008).

3.4 Computational performance

In order to compare our CUDA implementation of V1 (that is, the file v1colorME.cu) to the original, unmodified S&H implementation (which features code in both C and Matlab) we computed V1 complex cell responses (see Section 2.2.1) at a single spatiotemporal scale to a drifting sinusoidal grating (the same stimulus as described in Section 3.1) and recorded the model's execution time. The S&H C/Matlab code was executed as shModel(stim,pars,'v1Complex'), where stim was the input stimulus, and pars were the default parameters (shPars). Fig. 8a shows the execution time per video frame for both models. Our GPU implementation (red) was not only faster (except for relatively small networks) than the S&H C/Matlab implementation (blue), but it also scaled better with network size. Note that the C/Matlab implementation was a single-threaded computation. The largest speedup, a factor of 12, was observed for a network consisting of $96 \times 96 = 9,216$ neurons. It is likely that even greater speedups could have been achieved on larger networks, but these networks could not run with the S&H C/Matlab implementation because they ran out of memory. Timing was performed using standard commands tic and toc in Matlab, and the <ctime> function time in C++/CUDA. For the S&H C/Matlab implementation, the time it took to create the stimulus was not included in the time measurement. On the other hand, in the CUDA implementation the stimulus had to be read from file frame-by-frame and copied to the GPU card. However, we did not include the time it takes to transfer the response back from the device to the host.

Additionally, the S&H C/Matlab implementation is memory-intensive (see Fig. 8b), and execution times for networks above size $128 \times 128 = 16,384$ could not be computed because the CPU ran out of memory, even though we had a relatively large amount of RAM (24 GB) available. Measuring memory usage in Matlab is not straight-forward. In order to demonstrate the excessive memory consumption of the S&H C/Matlab implementation (see Fig. 8b) we opted to measure two metrics: the size of the output argument ans to function call shModel (blue, filled circle in Fig. 8b) and the maximum memory usage of the Matlab process at any point in time (blue, open circle). The first was measured with native Matlab command whos, and the latter was measured by running a bash script in the background that reported the memory usage of the process every second (using linux command ps). The blue dashed line is the 24 GB limit of the system's RAM. Note the log scale on the ordinate. Less memory was required to run the process than to store the output argument, which consisted of a matrix whose size was proportional to the product of the stimulus dimensions and the number of frames. A straightforward way of making the S&H C/Matlab implementation capable of handling large inputs would thus be to break up the output argument into smaller chunks of data. On the other hand, the memory usage of the GPU implementation was significantly lower (red line in Fig. 8b) and scaled better with network size. We used CUDA command cuMemGetInfo to identify the amount of allocated memory on the GPU. The red dashed line is the upper limit of GPU memory available to the user (roughly 5.2 GB on our card).

Comparing the performance between GPU simulation mode and CPU simulation mode with the full network on the specific processor remains to be demonstrated. Recall from Section 2.1.2 that in GPU mode all data structures are allocated on the GPU, whereas in CPU mode the network would be allocated on the CPU's memory, and only the generation of motion energy responses (written in CUDA) would be delegated to the GPU. Hence we evaluated the computational performance by running the full network in both CPU and GPU mode with input images from 16×16 pixels (38,784 neurons) to 64×64 pixels (610,944 neurons). The simulation speed is given as the ratio of execution time over the simulation time (see Fig. 9a) for networks run in CPU mode (blue) and GPU mode (red). Note that in both modes, the V1 CUDA implementation was executed (green), whose run-time is part of the total simulation time (in blue and red). The GPU simulations not only ran faster, but also simulation speed scaled better with network size. Note that the CPU simulation was a single-threaded computation. The full network at 40×40 input resolution (239,040 neurons) ran in real-time on the GPU. At 32×32 input resolution (153,216 neurons) the simulation was 1.5 times faster than real-time. This result compares favorably with previous releases of our simulator (Nageswaran et al. 2009; Richert et al. 2011), which is partly due to code-level optimizations, but mostly due to differences in GPU hardware and the V1 stage of the network being spatiotemporal filters instead of spiking neurons. As the network size increased, the GPU simulations showed a significant speedup over the CPU (see Fig. 9b). Speedup was computed as the ratio of CPU to GPU execution time. The largest network we could fit on a single GPU roughly corresponded to 64×64 input resolution (610,944 neurons), which ran approximately 30 times faster than on the CPU. Larger networks currently do not fit on a single GPU and as such must be run on the CPU, which would be more than 70 times slower than real-time judging from Fig. 9a.

4. Discussion

We presented a large-scale spiking model of visual area MT that 1) is capable of exhibiting both component and pattern motion selectivity, 2) generates speed tuning curves that are in agreement with electrophysiological data, 3) reproduces behavioral responses from a 2AFC task, 4) outperforms a previous rate-based implementation of the motion energy model (Simoncelli and Heeger 1998) in terms of computational speed and memory usage, 5) is implemented on a publicly available SNN simulator that allows for real-time execution on off-the-shelf GPUs, and 6) is comprised of a neuron model, synapse model, and address-event representation (AER), which is compatible with recent neuromorphic hardware (Srinivasa and Cruz-Albrecht 2012).

The model is based on two previous models of motion processing in MT (Simoncelli and Heeger 1998; Rust et al. 2006), but differs from these models in several ways. First, our model contains the tuned normalization in the MT stage that was not present in Simoncelli and Heeger (1998) but introduced by Rust et al. (2006). Second, the implementation by Rust et al. (2006) was restricted to inputs that are mixtures of 12 sinusoidal gratings of a fixed spatial and temporal frequency, whereas our model can operate on any spatiotemporal image intensity. Third, MT PDS cells in our model sum over inputs from MT CDS cells as opposed to inputs from V1 cells, although the two approaches are conceptually equivalent. Fourth, instead of using linear summation and a static nonlinear transformation, all neuronal and synaptic dynamics in our model MT were achieved using Izhikevich spiking neurons and conductance-based synapses.

One could argue that the inclusion of Izhikevich spiking neurons and conductance-based synapses is unnecessary, since previous incarnations of the motion energy model did not feature these mechanisms yet were perfectly capable of reproducing speed tuning and motion selectivity. However, our approach is to be understood as a first step into modeling large-scale networks of visual motion processing in more biological detail, with the ultimate goal of understanding how the brain solves the aperture problem, among other open issues in motion perception. Integrating the functionality demonstrated in previous models with more neurobiologically plausible neuronal and synaptic dynamics is a necessary first step into analyzing the temporal dynamics of model neurons in MT, which may 1) help to explain how MT PDS cell establish their pattern selectivity not instantly but over a time-course on the order of 100 ms (M. A. Smith et al. 2005) and 2) enable the addition of spike-based learning rules such as STDP; both of which might be harder to achieve with previous model incarnations. Additionally, the introduction of the present neuron model, synapse model, and address-event representation (AER) did not affect performance, yet enabled the integration of the S&H model with recent neuromorphic hardware (Srinivasa and Cruz-Albrecht 2012) (see also Section 4.3).

On the other hand, it is possible (if not likely) that some response dynamics produced by the neural circuitry in the retina, the lateral geniculate nucleus (LGN), and V1 may account for certain response properties of neurons in MT. Thus future work could be directed towards implementing the entire early visual system in the spiking domain. However, for the purpose of this study we deem a rate-based preprocessor to be an adequate abstraction, as the core functionality of directionally selective cells in V1 seem to be well-characterized by local motion energy filters (Adelson and Bergen 1985; DeAngelis et al. 1993; Movshon and Newsome 1996).

4.1 Neurophysiological evidence and model alternatives

There is evidence that MT firing rates represent the velocity of moving objects using the IOC principle. A psychophysical study showed that the perception of moving plaids depends on conditions that specifically affect the detection of individual grating velocities (Adelson and Movshon 1982). This is consistent with a two-stage model in which component velocities are first detected and then pooled to compute pattern velocity. Subsequent physiological studies broadly support such a cascade model (Perrone and Thiele 2001; Rust et al. 2006; M. A. Smith et al. 2005).

However, other psychophysical results exist where the perceived direction of plaid motion deviates significantly from the IOC direction (Ferrera and Wilson 1990; Burke and Wenderoth 1993). Alternatives to the IOC principle are, for example, vector average (VA) or feature tracking. VA predicts that the perceived pattern motion is the vector average of the component velocity vectors. Blob or feature tracking is the process of locating something (a “feature”) that does not suffer from the aperture problem, such as a bright spot or a T-junction, and tracking it over time (Wilson et al. 1992). Ultimately, one needs to consider the interactions of the motion pathway with form mechanisms (Majaj et al. 2007), and model the processing of more complex stimuli (e.g., motion transparency, additional self-motion, multiple moving objects) (Raudies et al. 2011; Layton et al. 2012). Clarifying by which rule (or combination of rules) the brain integrates motion signals is still a field of ongoing research. For recent reviews on the topic see (Bradley and Goyal 2008; Nishida 2011).

Although clear evidence for spatiotemporal frequency inseparability in MT neurons has been found (Perrone and Thiele 2001), which supports the idea of a motion energy model, later studies reported it to be a weak effect (Priebe et al. 2003; Priebe et al. 2006). The actual proportion of neurons in the primate visual system that are tuned to spatiotemporal frequency is currently not known.

4.2 Model limitations

Although our model is able to capture many attributes of motion selectivity (e.g., direction selectivity, speed tuning, component and pattern motion), it is not yet complete for the following reasons. First, it does not explicitly specify the exact pattern velocity, but instead reports an activity distribution over the population of MT neurons, whose firing rates are indicative of the observed pattern motion. In order to estimate the speed of a target stimulus, it has been proposed to use a suitable population decoding mechanism that operates on MT responses (Perrone 2012; Hohl et al. 2013). Second, our model does not attempt to predict the temporal dynamics of MT PDS cells, which often respond with broad selectivity when first activated, sometimes even resembling CDS cells, and only over a time-course on the order of 100 ms establish their pattern motion selectivity (M. A. Smith et al. 2005). A possible explanation for these temporal dynamics is given in Chey et al. (1997). Third, it does not consider the visual form pathway and abstracts early visual details that may be critical for operation in natural settings. Fourth, the extent to which each stage in the motion energy model can be mapped onto specific neuronal populations is rather limited. Tiling the spatiotemporal frequency space according to the motion energy model is biologically implausible, and the temporal extent of the filters is unrealistically long (especially the low speed filters). However, a way to combine spatiotemporal filters based on V1 neuron properties into a pattern motion detector has been proposed in Perrone and Thiele (2002).

Another more fundamental limitation is that the S&H model (or for that matter, any spatiotemporal-energy based model including the elaborated Reichardt detector) can only sense so-called first-order motion, which is defined as spatiotemporal variations in image intensity (first-order image statistics) that give rise to a Fourier spectrum. Second-order stimuli, such as the motion of a contrast modulation over a texture, are non-Fourier and thus invisible to the model, yet can be readily perceived by humans (Chubb and Sperling 1988). In addition, the existence of a third motion channel has been suggested, which is supposed to operate through selective attention and saliency maps (Lu and Sperling 1995). Also, MT has been shown to be involved in color-based motion perception (Thiele et al. 2001).

There is also a plainly technical limitation to our model, which is manifested in the amount of available GPU memory. Due to their size, large-scale spiking networks have demanding memory requirements. The largest network that could fit on a single NVIDIA Tesla M2090 (with 6 GB of memory) was comprised of 610,944 neurons and approximately 137 million synapses, which corresponds to processing a 64×64 input video. In order to run larger networks on current-generation GPU cards, a change in model or (software and hardware) architecture is required. One should note that this is only a temporary limitation and could become obsolete as soon as with the next generation of GPU cards. Another possible solution would be to employ multi-GPU systems; however, more work is required to efficiently integrate our SNN simulator with such a system.

4.3 Practical implications

The present network might be of interest to the neuroscientist and computer vision research communities for the following reasons.

First, our implementation outperforms the S&H C/Matlab implementation by orders of magnitude in terms of computational speed and memory usage. Thus our CUDA implementation can be used to save computation time, as well as be applied to input resolutions that the C/Matlab implementation cannot handle due to memory constraints. Additionally, the CUDA implementation can act as a stand-alone module that could potentially be used in computer vision as an alternative to computationally expensive operations such as Gabor filtering for edge detection or dense optic flow computations.

Second, we have demonstrated that our approach is fast, efficient, and scalable; although current GPU cards limit the size of the simulations due to memory constraints. Nevertheless, our model processes a 40×40 input video at 20 frames per second in real-time, which corresponds to a total of 239,040 neurons in the simulated V1, MT, and LIP areas, at 20 frames per second using a single GPU, which enables the potential use of our software in real-time applications ranging from robot vision to autonomous driving.

Third, our implementation might be of particular interest to the neuromorphic modeling community, as the present neuron model, synapse model, and AER are compatible with recent neuromorphic hardware

(Srinivasa and Cruz-Albrecht 2012). Thus our algorithm could be used as a neural controller in neuromorphic and neurorobotics applications. Future work could be directed toward creating an interface by which networks can be automatically exported onto neuromorphic hardware.

Fourth, because of the modular code structure, our implementation can be readily extended to include, for example, higher-order visual areas or biologically plausible synaptic learning rules such as STDP. Thus our implementation may facilitate the testing of hypotheses and the study of the temporal dynamics that govern visual motion processes in area MT, which might prove harder to study using previous (rate-based) model incarnations.

Lastly, the network was constructed using a SNN simulator that is publicly available at http://www.socsci.uci.edu/~jkrichma/CARLsim/. The present release features the complete source code for the simulator, the network, and analysis scripts. As such it is the next step towards our goal of making efficient simulations of large-scale spiking networks available to a wide range of researchers, without the need of a cluster or supercomputer.

5. Information Sharing Statement

The source code for the simulator, for the network, and analysis scripts are publicly available at http://www.socsci.uci.edu/~jkrichma/CARLsim/. This website does also feature installation instructions, source code documentation and a tutorial on how to set up, run, and interact with a simulation. In order to run the simulator in CUDA mode, the NVIDIA CUDA software developer kit must be installed (freeware, available at https://developer.nvidia.com/cuda-downloads).

6. Acknowledgments

This work was supported by the Defense Advanced Research Projects Agency (DARPA) subcontract 801888-BS. We thank Jayram M. Nageswaran for his work developing the custom spiking neural network simulator. We also thank Michael Avery, Kris Carlson, and Steve Grossberg for valuable feedback and discussion on this project.

7. References

Adelson, E. H., & Bergen, J. R. (1985). Spatiotemporal energy models for the perception of motion. *J Opt Soc Am A, 2*(2), 284-299.

Adelson, E. H., & Movshon, J. A. (1982). Phenomenal coherence of moving visual patterns. *Nature, 300*(5892), 523-525.

Bradley, D. C., & Goyal, M. S. (2008). Velocity computation in the primate visual system. *Nature Reviews Neuroscience, 9*(9), 686-695, doi:Doi 10.1038/Nrn2472.

Browning, N. A., Grossberg, S., & Mingolla, E. (2009a). Cortical dynamics of navigation and steering in natural scenes: Motion-based object segmentation, heading, and obstacle avoidance. *Neural Networks, 22*(10), 1383-1398, doi:DOI 10.1016/j.neunet.2009.05.007.

Browning, N. A., Grossberg, S., & Mingolla, E. (2009b). A neural model of how the brain computes heading from optic flow in realistic scenes. *Cogn Psychol, 59*(4), 320-356, doi:10.1016/j.cogpsych.2009.07.002.

Burke, D., & Wenderoth, P. (1993). The Effect of Interactions between One-Dimensional Component Gratings on 2-Dimensional Motion Perception. *Vision Research, 33*(3), 343-350, doi:Doi 10.1016/0042-6989(93)90090-J.

Chey, J., Grossberg, S., & Mingolla, E. (1997). Neural dynamics of motion grouping: from aperture ambiguity to object speed and direction. *Journal of the Optical Society of America a-Optics Image Science and Vision, 14*(10), 2570-2594, doi:Doi 10.1364/Josaa.14.002570.

Chubb, C., & Sperling, G. (1988). Drift-Balanced Random Stimuli - a General Basis for Studying Non-Fourier Motion Perception. *Journal of the Optical Society of America a-Optics Image Science and Vision, 5*(11), 1986-2007, doi:Doi 10.1364/Josaa.5.001986.

Dayan, P., & Abbott, L. F. (2001). *Theoretical neuroscience : computational and mathematical modeling of neural systems* (Computational neuroscience). Cambridge, Mass.: Massachusetts Institute of Technology Press.

DeAngelis, G. C., Ohzawa, I., & Freeman, R. D. (1993). Spatiotemporal organization of simple-cell receptive fields in the cat's striate cortex. II. Linearity of temporal and spatial summation. *Journal of Neurophysiology, 69*(4), 1118-1135.

Ferrera, V. P., & Wilson, H. R. (1990). Perceived direction of moving two-dimensional patterns. *Vision Research, 30*(2), 273-287.

Fidjeland, A. K., & Shanahan, M. P. Accelerated simulation of spiking neural networks using GPUs. In *Neural Networks (IJCNN), The 2010 International Joint Conference on, 18-23 July 2010 2010* (pp. 1-8). doi:10.1109/IJCNN.2010.5596678.

Freeman, W. T., & Adelson, E. H. The design and use of steerable filters. In *IEEE Pattern Analysis and Machine Intelligence, 1991* (Vol. 13, pp. 891-906)

Grossberg, S., & Pilly, P. K. (2008). Temporal dynamics of decision-making during motion perception in the visual cortex. *Vision Research, 48*(12), 1345-1373, doi:DOI 10.1016/j.visres.2008.02.019.

Hohl, S. S., Chaisanguanthum, K. S., & Lisberger, S. G. (2013). Sensory population decoding for visually guided movements. *Neuron, 79*(1), 167-179, doi:10.1016/j.neuron.2013.05.026.

Indiveri, G., Chicca, E., & Douglas, R. (2006). A VLSI array of low-power spiking neurons and bistable synapses with spike-timing dependent plasticity. *Ieee Transactions on Neural Networks, 17*(1), 211-221, doi:Doi 10.1109/Tnn.2005.860850.

Izhikevich, E. M. (2003). Simple model of spiking neurons. *Ieee Transactions on Neural Networks, 14*(6), 1569-1572, doi:Doi 10.1109/Tnn.2003.820440.

Izhikevich, E. M. (2004). Which model to use for cortical spiking neurons? *Ieee Transactions on Neural Networks, 15*(5), 1063-1070, doi:Doi 10.1109/Tnn.2004.832719.

Izhikevich, E. M. (2007). *Dynamical systems in neuroscience : the geometry of excitability and bursting* (Computational neuroscience). Cambridge, Mass.: MIT Press.

Izhikevich, E. M., Gally, J. A., & Edelman, G. M. (2004). Spike-timing dynamics of neuronal groups. *Cereb Cortex, 14*(8), 933-944, doi:DOI 10.1093/cercor/bhh053.

Khan, M., Lester, D., Plana, L., Rast, A., Jin, X., & Painkras, E. SpiNNaker: Mapping neural networks onto a massively-parallel chip multiprocessor. In *IEEE International Joint Conference on Neural Networks, 2008* (pp. 2849-2856)

Koch, C. (1999). *Biophysics of computation : information processing in single neurons* (Computational neuroscience). New York: Oxford University Press.

Layton, O. W., Mingolla, E., & Browning, N. A. (2012). A motion pooling model of visually guided navigation explains human behavior in the presence of independently moving objects. *J Vis, 12*(1), doi:10.1167/12.1.20.

Livingstone, M. S., & Conway, B. R. (2007). Contrast affects speed tuning, space-time slant, and receptive-field organization of simple cells in macaque V1. *Journal of Neurophysiology, 97*(1), 849-857, doi:10.1152/jn.00762.2006.

Lu, Z. L., & Sperling, G. (1995). Attention-Generated Apparent Motion. *Nature, 377*(6546), 237-239, doi:Doi 10.1038/377237a0.

Majaj, N. J., Carandini, M., & Movshon, J. A. (2007). Motion integration by neurons in macaque MT is local, not global. *Journal of Neuroscience, 27*(2), 366-370, doi:10.1523/JNEUROSCI.3183-06.2007.

Merolla, P. A., Arthur, J. V., Shi, B. E., & Boahen, K. A. (2007). Expandable networks for neuromorphic chips. *Ieee Transactions on Circuits and Systems I-Regular Papers, 54*(2), 301-311, doi:Doi 10.1109/Tcsi.2006.887474.

Movshon, J. A., Adelson, E. H., Gizzi, M. S., & Newsome, W. T. (1985). *The analysis of moving visual patterns* (Pattern recognition mechanisms). New York: Springer.

Movshon, J. A., & Newsome, W. T. (1996). Visual response properties of striate cortical neurons projecting to area MT in macaque monkeys. *Journal of Neuroscience, 16*(23), 7733-7741.

Nageswaran, J. M., Dutt, N., Krichmar, J. L., Nicolau, A., & Veidenbaum, A. V. (2009). A configurable simulation environment for the efficient simulation of large-scale spiking neural networks on graphics processors. *Neural Networks, 22*(5-6), 791-800, doi:DOI 10.1016/j.neunet.2009.06.028.

Nishida, S. (2011). Advancement of motion psychophysics: Review 2001-2010. *Journal of Vision, 11*(5), doi:Artn 11
Doi 10.1167/11.5.11.

Pack, C. C., Berezovskii, V. K., & Born, R. T. (2001). Dynamic properties of neurons in cortical area MT in alert and anaesthetized macaque monkeys. *Nature, 414*(6866), 905-908, doi:10.1038/414905a.

Perrone, J. A. (2012). A neural-based code for computing image velocity from small sets of middle temporal (MT/V5) neuron inputs. *J Vis, 12*(8), doi:10.1167/12.8.1.

Perrone, J. A., & Thiele, A. (2001). Speed skills: measuring the visual speed analyzing properties of primate MT neurons. *Nat Neurosci, 4*(5), 526-532.

Perrone, J. A., & Thiele, A. (2002). A model of speed tuning in MT neurons. *Vision Research, 42*(8), 1035-1051.

Priebe, N. J., Cassanello, C. R., & Lisberger, S. G. (2003). The neural representation of speed in macaque area MT/V5. *Journal of Neuroscience, 23*(13), 5650-5661.

Priebe, N. J., Lisberger, S. G., & Movshon, J. A. (2006). Tuning for spatiotemporal frequency and speed in directionally selective neurons of macaque striate cortex. *Journal of Neuroscience, 26*(11), 2941-2950, doi:10.1523/JNEUROSCI.3936-05.2006.

Raudies, F., Mingolla, E., & Neumann, H. (2011). A model of motion transparency processing with local center-surround interactions and feedback. *Neural Comput, 23*(11), 2868-2914, doi:10.1162/NECO_a_00193.

Resulaj, A., Kiani, R., Wolpert, D. M., & Shadlen, M. N. (2009). Changes of mind in decision-making. *Nature, 461*(7261), 263-U141, doi:Doi 10.1038/Nature08275.

Richert, M., Nageswaran, J. M., Dutt, N., & Krichmar, J. L. (2011). An efficient simulation environment for modeling large-scale cortical processing. *Front Neuroinform, 5*, 19, doi:10.3389/fninf.2011.00019.

Rodman, H. R., & Albright, T. D. (1987). Coding of Visual Stimulus Velocity in Area Mt of the Macaque. *Vision Research, 27*(12), 2035-2048, doi:Doi 10.1016/0042-6989(87)90118-0.

Rodman, H. R., & Albright, T. D. (1989). Single-unit analysis of pattern-motion selective properties in the middle temporal visual area (MT). *Exp Brain Res, 75*(1), 53-64.

Roitman, J. D., & Shadlen, M. N. (2002). Response of neurons in the lateral intraparietal area during a combined visual discrimination reaction time task. *Journal of Neuroscience, 22*(21), 9475-9489.

Rust, N. C., Mante, V., Simoncelli, E. P., & Movshon, J. A. (2006). How MT cells analyze the motion of visual patterns. *Nat Neurosci, 9*(11), 1421-1431, doi:Doi 10.1038/Nn1786.

Shadlen, M. N., & Newsome, W. T. (2001). Neural basis of a perceptual decision in the parietal cortex (area LIP) of the rhesus monkey. *Journal of Neurophysiology, 86*(4), 1916-1936.

Simoncelli, E. P., & Heeger, D. J. (1998). A model of neuronal responses in visual area MT. *Vision Research, 38*(5), 743-761, doi:Doi 10.1016/S0042-6989(97)00183-1.

Smith, M. A., Majaj, N. J., & Movshon, J. A. (2005). Dynamics of motion signaling by neurons in macaque area MT. *Nat Neurosci, 8*(2), 220-228, doi:Doi 10.1038/Nn1382.

Smith, P. L., & Ratcliff, R. (2004). Psychology and neurobiology of simple decisions. *Trends in Neurosciences, 27*(3), 161-168, doi:DOI 10.1016/j.tins.2004.01.006.

Srinivasa, N., & Cruz-Albrecht, J. M. (2012). Neuromorphic Adaptive Plastic Scalable Electronics Analog Learning Systems. *Ieee Pulse, 3*(1), 51-56, doi:Doi 10.1109/Mpul.2011.2175639.

Thiele, A., Dobkins, K. R., & Albright, T. D. (2001). Neural correlates of chromatic motion perception. *Neuron, 32*(2), 351-358.

van Santen, J. P. H., & Sperling, G. (1985). Elaborated Reichardt Detectors. *Journal of the Optical Society of America a-Optics Image Science and Vision, 2*(2), 300-321.

Vogelstein, R. J., Mallik, U., Culurciello, E., Cauwenberghs, G., & Etienne-Cummings, R. (2007). A multichip neuromorphic system for spike-based visual information processing. *Neural Comput, 19*(9), 2281-2300, doi:DOI 10.1162/neco.2007.19.9.2281.

Wilson, H. R., Ferrera, V. P., & Yo, C. (1992). A Psychophysically Motivated Model for 2-Dimensional Motion Perception. *Visual Neuroscience, 9*(1), 79-97.

Yudanov, D., Shaaban, M., Melton, R., & Reznik, L. GPU-based simulation of spiking neural networks with real-time performance &amp; high accuracy. In *Neural Networks (IJCNN), The 2010 International Joint Conference on, 18-23 July 2010 2010* (pp. 1-8). doi:10.1109/IJCNN.2010.5596334.

8. Figure captions

Fig. 1 Network architecture. 32×32 grayscale images are fed through model V1, MT, and LIP (as explained in Sections 2.2.1 – 2.2.3). Shown is a snapshot in time of the network's response to an example RDK stimulus in which 50 % of the dots drift to the right. Black bold arrows denote synaptic projections. Inhibitory projections and populations are not shown. Numbers in parentheses next to an element are the equations that describe the corresponding neuronal response or synaptic projections (see text). V1 filter responses were mapped onto mean firing rates by reproducing the contrast sensitivity function reported for V1 cells projecting to MT, as explained in Section 2.2.1

Fig. 2 A drifting dot traces out a path (dashed line) in space (x, ignoring y) and time (t). The colored ovals correspond to the orientation of the positive (green) and negative (red) lobes of a spatiotemporal filter **a** If the filter is oriented in the same way as the dot's space-time path it could be activated by this motion **b** A dot moving in the opposite direction would always contact both positive and negative lobes of the filter and therefore could never produce a strong response. Adopted from (Bradley and Goyal 2008)

Fig. 3 The contrast sensitivity function of model V1 simple cells (blue) is plotted against electrophysiological data adapted from Fig. 7 of (Movshon and Newsome 1996). Each data point is a V1 mean response to a drifting grating, averaged over both one second of stimulus presentation and all neurons in the subpopulation. Vertical bars are the standard deviation on the population average

Fig. 4 Polar plots of direction tuning for a sinusoidal grating **a–d** and a plaid stimulus **e–h** drifting upwards, where the angle denotes motion direction and the radius is the firing rate in spikes per second. Tuning curves were obtained by taking the mean firing rate of a neuron to a drifting grating during two seconds of stimulus presentation, averaged over all neurons in

the population selective to the same stimulus direction (black: mean neuronal response, blue: mean plus standard deviation on the population average, green: mean minus standard deviation). Shown are mean responses for V1 complex cells (**b** and **f**), MT CDS cells (**c** and **g**), and MT PDS cells (**d** and **h**). Only MT PDS cells **h** responded to the motion of the entire plaid pattern rather than to the motions of the individual component gratings

Fig. 5 The pattern index is computed for all MT CDS cells (blue) and all MT PDS cells (red), and plotted as a Fisher Z-score. The black solid lines are the classification region boundaries, indicating that all MT CDS cells have indeed been classified as component-selective, and all MT PDS cells have been classified as pattern-selective

Fig. 6 Speed tuning curves for three different classes of MT neurons. The stimulus consisted of a single bar drifting over the entire visual field either to the right (preferred direction) or to the left (anti-preferred direction) at different speeds **a** Response of a "speed-tuned" neuron (selective to motion at 1.5 pixels per frame) **b** Response of a "low-pass" neuron (selective to motion at 0.125 pixels per frame) **c** Response of a "high-pass" neuron (selective to motion at 9 pixels per frame)

Fig. 7 Random dot kinematogram. The RDK stimulus was constructed out of approximately 150 dots (15 % dot density, maximum stimulus contrast) on a 32x32 input movie **a** Psychometric function. The network's accuracy increased with increasing motion strength (coherence level) **b** Chronometric function. The network's RT decreased with increasing motion strength

Fig. 8 a Execution time of a Matlab implementation (blue) of V1 complex cells versus a CUDA implementation (red) **b** Observed memory usage for the Matlab implementation (blue) and CUDA implementation (red)

Fig. 9 a Simulation speed is given as the ratio of execution time over the simulation time for networks run in CPU mode (blue) and GPU mode (red). In both cases, the V1 CUDA implementation was executed (green), which is part of the total simulation time (in blue and red). Note the log scale on the ordinate. The GPU simulations did not only run faster, but simulation speed scaled better with network size **b** Speedup is given as the ratio of CPU execution time over GPU execution time