Neurokernel: An Open Scalable Software Framework for Emulation and Validation of Drosophila Brain Models on Multiple GPUs

Neurokernel RFC #1 - v1.0

Lev E. Givon, Aurel A. Lazar
Bionet Group
Department of Electrical Engineering
Columbia University

February 11, 2014

Abstract

The brain of the fruit fly *Drosophila melanogaster* is an extremely attractive model system for reverse engineering the emergent properties of neural circuits because it implements complex sensory-driven behaviors with a nervous system comprising a number of components that is five orders of magnitude smaller than those of mammals. A powerful toolkit of well-developed genetic techniques and advanced electrophysiological recording tools enables the fly's behavior to be experimentally linked to the function of its neural circuitry. To enable neuroscientists to use these strengths of fly brain research to surmount the structural complexity of its brain and create an accurate model of the entire fly brain, we have developed an open Python framework called Neurokernel designed to enable collaborative development of comprehensive fly brain models and their execution and testing on multiple Graphics Processing Units (GPUs). Neurokernel's model support architecture is motivated by the organization of the fly brain into fewer than 50 functional modules called local processing units (LPUs) that are each characterized by a unique population of local neurons. By defining communication interfaces that specify how spikes and neuron membrane states are transmitted between LPUs, Neurokernel enables researchers to collaboratively develop and refine whole-brain emulations by integration of independently developed processing units. Neurokernel will also empower researchers to leverage additional GPU resources and future improvements in GPU technology to accelerate model execution to the same time scale as a live fly brain; this will enable in vivo validation of Neurokernel-based models against real-time recordings of live fly brain activity. We demonstrate Neurokernel's module interfacing feature by using it to integrate independently developed models of olfactory and vision LPUs based upon experimentally obtained connectivity information.

Contents

1 Introduction

There exist a range of ongoing projects to develop cortical-scale computational models of the mammalian brain (i.e., containing $\sim 10^9$ neurons) at high levels of biophysical faithfulness [47, 26, 2]. These efforts have shed light on many important computational considerations that must be confronted in order to efficiently emulate the brain's massively parallel and fundamentally asynchronous architecture. The structural complexity and high computational demands of modeling the enormous number of neurons and synapses in the mammalian brain, however, severely complicates the reverse engineering of this architecture; simulations of very short intervals of activity in cortical-scale neuronal networks [26, 15] currently require hours - if not days - of processing time. Execution of these models also requires the use of supercomputers that are only accessible to a limited number of researchers. Successful development of human brain models should therefore be preceded by an increased understanding of the magnitude of the structural/architectural complexity of more tractable brains of simpler organisms and how they implement specific information processing functions and govern behavior [28]. In order to facilitate their improvement and extension by other neuroscience researchers, brain model implementations must also be executable on widely available parallel hardware platforms.

The nervous system of the fruit fly *Drosophila melanogaster* possesses a range of features that recommend it as a model organism for relating brain structure to function. Despite the obvious differences in size and complexity between the mammalian and fly brains, researchers dating back to Cajal have observed common design principles in the structure of their sensory subsystems [68]. Many of the genes and proteins expressed in the mammalian brain are also conserved in the genome of *Drosophila* [4]. These features strongly suggest that valuable insight into the workings of the mammalian brain can be obtained by focusing on that of *Drosophila*.

Remarkably, the fruit fly is capable of a host of complex nonreactive behaviors that are governed by a brain containing only 135,000 neurons [1]. The relationship between the fly's brain and its behaviors can be experimentally probed using a powerful toolkit of genetic techniques for manipulation of the fly's neural circuitry such as the GAL4 driver system [14, 66, 71, 76, 46], recent advances in experimental methods for precise recordings of the fly's neuronal responses to stimuli [31, 32, 34, 33, 35, 80], techniques for analyzing the fly's behavioral responses to stimuli [7, 30, 45, 11], and progress in reconstruction of the fly connectome, or neural connectivity map [12, 73]. These techniques have provided access to an immense amount of valuable structural and behavioral data that can be used to model how the fly brain's neural circuitry implements processing of sensory stimuli [18, 49, 10, 25, 51, 69].

Understanding how the brain's functionality is implemented by neural circuits requires a means of exploring the emergent properties of its constituent circuits [1]. Although existing cortical-scale models of mammalian brains do explicitly attempt to account for varying levels of structural complexity in the brain, these models provide virtually no architecture for the computational exploration of the brain's functional complexity. The need for such an architecture is evident from the requirements of engineering complex artificial systems that simultaneously implement multiple information processing or control operations by means

of interrelated functional modules; reverse-engineering the brain not only entails simulating its constituent neurons and synapses, but requires a means of ascribing function to specific parts of the brain and investigating how the interaction of those parts gives rise to complex information processing and behavior. In addition to its comparatively tractable number of neural components, the fruit fly brain possesses a modular structure that lends itself to the design of such an architecture.

Effectively capitalizing upon the opportunities afforded by the fly brain's modular structure and the techniques available to experimentally manipulate it requires the combined expertise of multiple researchers in the neuroscience community. Efforts to develop a working computational model of the fly brain therefore require a software platform that explicitly enables a collaborative approach to fly brain modeling using the most powerful computing hardware that is accessible to a wide range of researchers. Graphics Processing Units (GPUs) currently best fulfill the latter criterion; we will discuss this design choice in § 2.1.

Building a detailed model of the brain of even a relatively simple organism such as the fruit fly requires the use of multiple GPU resources to address the number of components it contains. The complexity of achieving scaling and - ultimately - real-time operation over such resources while providing the programmability required to model the constituent functional modules in the fly brain necessitates a software architecture akin to an operating system kernel; just as an operating system must be able to allocate computational resources and serve as an extended machine that provides the services needed by software applications, so too must a software platform for modeling the entire brain of an organism be able to allocate computational resources (i.e., available GPUs) and provide the programmability (i.e., software interfaces and computing models) required by researchers to access and connect a model's constituent elements.

To address the above requirements of modeling the entire fly brain, we have designed an open software framework called Neurokernel for implementing fly brain models based upon actual connectome data and executing them upon multiple GPUs. This article details the features of the first version of Neurokernel; we discuss the framework's general architecture in § 2.1 and the main biological entities in the fruit fly brain are reviewed in § 2.2. In contrast to general-purpose neural simulators, Neurokernel's design specifically targets the fruit fly brain's local processing structure and the synaptic connectivity patterns that link them; we discuss these structures and their design implications in § 2.3. In § 2.4, we describe Neurokernel's current module API. We tested Neurokernel's support for collaborative design and model integration by connecting Neurokernel-based models of the fly's olfactory and vision systems independently developed by two teams with a third module that performs multisensory coincidence detection; this integration is discussed in section § 3.

2 Framework Design and Features

2.1 Overall Architecture

We refer to our software framework for fly brain emulation as a *kernel* because it provides two main functions associated with traditional computer operating systems [38]: it serves as a *resource allocator* that enables the scalable use of parallel computing resources to accelerate the execution of an emulation, and it serves as an *extended machine* that provides software services and interfaces that can be programmed to emulate and integrate functional modules in the fly brain.

Neurokernel's architectural design consists of three planes that separate between the time scales of a model's representation and its execution on multiple parallel processors (Fig. 1). This enables the design of vertical APIs that permit development of new features within one plane while minimizing the need to modify code associated with the other planes. Services that implement the computational primitives and numerical methods required to execute supported models on parallel processors are provided by the framework's *compute plane*. Translation or mapping of a models' specified components to the methods provided by the compute plane and management of the parallel hardware and data communication resources required to efficiently execute a model is performed by Neurokernel's *control plane*. Finally, the framework's *application plane* provides support for specification of neural circuit models, connectivity patterns, and interfaces that enable independently developed models of the fly brain's functional subsystems to be interconnected. We discuss these interfaces in § 2.3. In the remaining paragraphs of this section, we present our rationale for choosing a GPU-based hardware platform as the substrate for the realization of Neurokernel's compute plane.

GPU technology affords researchers an increasingly powerful yet highly cost-effective parallel computing medium. Recently designed GPU-based systems for emulating neuronal networks of single spiking neuron types have demonstrated near real-time execution performance for networks of up to $\sim 10^5$ spiking neurons and $\sim 10^7$ synapses using single GPUs [53, 16, 65]; the increasing availability of commodity GPU hardware enables a wide range of researchers to take advantage of such neural modeling systems. There is also growing interest in combining the power of multiple GPUs to address more ambitious computational neuroscience experiments [74, 54]; these applications stand to benefit from developments in GPU technology that accelerate communication between GPUs [56, 57].

In contrast to other currently available GPU-based neural emulation packages [53, 52, 55, 65], Neurokernel is implemented entirely in Python, a high-level language with a rich ecosystem of scientific packages [27, 58, 61] that has enjoyed increasing popularity in neuroscience research. Although GPUs can be directly programmed using frameworks such as CUDA[1] and OpenCL[2], the difficulty of writing and optimizing code using these frameworks exclusively has led to the development of packages that enable run time code generation (RTCG) using higher level languages [6]. Neurokernel uses the PyCUDA package to provide RTCG support for NVIDIA's GPU hardware without forgoing the development advantages

[1] http://www.nvidia.com/cuda

[2] http://www.khronos.org/opencl

afforded by Python [36].

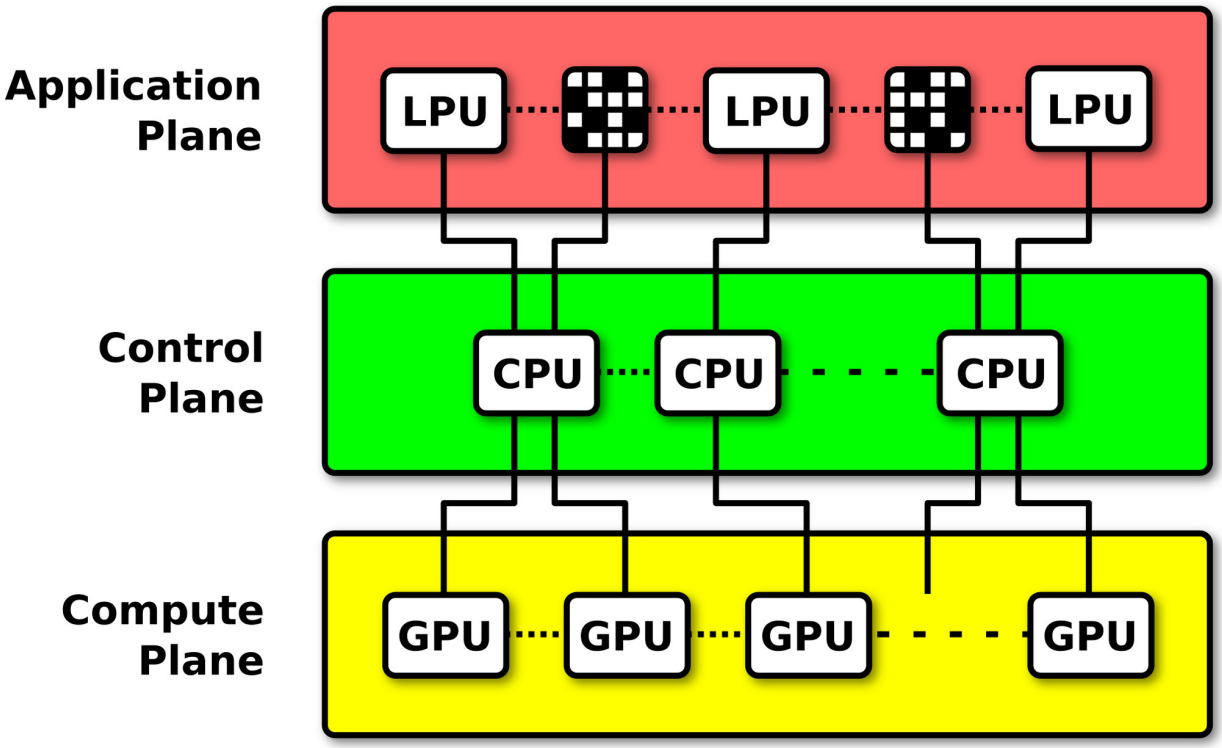

Figure 1: The three-plane structure of the Neurokernel architecture is based on the principle of separation of time scales. The application plane provides support for hardware-independent specification of LPUs and their interconnects. Services that implement the neural primitives and computing methods required to execute neural circuit model instantiations on GPUs are provided by the compute plane. Translation or mapping of specified model components to the methods provided by the compute plane and management of multiple GPUs and communication resources is performed by the control plane operating on a cluster of CPUs.

2.2 Biological Structures

A successful determination of how the brain's highly complex structure implements specific functions requires its decomposition into functional modules whose input-output relationships can be individually analyzed and whose interactions can be explained in terms of the groups of synaptic connections that exist between them [3]. Analysis of the *Drosophila* connectome has revealed that its brain can be decomposed into fewer than 50 distinct neural circuits, most of which correspond to anatomically distinct regions in the fly brain [10, 44]. These regions, or neuropils, include sensory processing structures such as the olfactory system's antennal lobe and the vision system's lamina and medulla, as well as higher level

structures such as the protocerebral bridge that receive input from sensory LPUs (Fig. 2). Most of these modules are referred to as local processing units (LPUs) because they are characterized by unique populations of local neurons whose processes are restricted to specific neuropils.

The axons of an LPU's local neurons and the synaptic connections between them and other neurons in the LPU constitute an internal pattern of connectivity that is distinct from the bundles, or tracts, of projection neuron processes that transmit data to neurons in other LPUs (Fig. 2); this suggests that the local neuron population is integral to determining an LPU's functional properties. The fly brain also comprises modules known as hubs that contain no local neurons; they appear to serve as communication relays between different LPUs. In contrast to a purely anatomical subdivision, the decomposition of the brain into functional modules casts the problem of reverse engineering the brain as one of discovering the processing performed by each individual LPU and determining how specific patterns of axonal connectivity between these LPUs integrates them into functional subsystems. Specification and interconnection of models of these functional modules constitute the fundamental design requirements of Neurokernel's application plane.

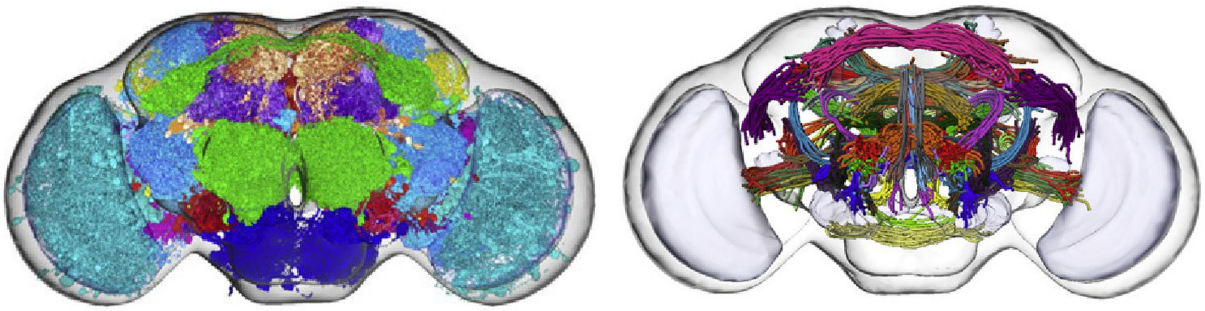

Figure 2: Modular structure of fly brain. Individual LPUs, hubs, and tracts are identified by different colors; for example, the central green structures in the left-hand figure are the antennal lobes, while the large peripheral cyan structures are the medullae. Most LPUs are paired across the fly's two hemispheres. Tracts depicted in the right-hand figure may connect pairs of LPUs located in each hemisphere or within a single hemisphere ([10], reproduced with permission).

2.3 Architecture Components

To enable integration of models of the LPUs and tracts respectively depicted in Fig. 2 into models of functional subsystems (Fig. 4), Neurokernel defines an API for creating LPU models that can interface with other Neurokernel-enabled model implementations. This API consists of a set of Python base classes from which all LPU models must be derived. These classes provide LPU designers with the freedom to organize the internal structure of their model implementations largely as they see fit; the classes require that (1) each LPU in

an emulation be uniquely identified, (2) that all operations performed by a single step of the LPU's execution be exposed in a single method, and that (3) an LPU must advertise the number of communication ports it publicly exposes to send or receive data to enable Neurokernel's communication mechanism to determine the size of the data arrays that must be transmitted between LPUs. An LPU's input and output ports can be thought of as respectively corresponding to the dendrites and axons of its exposed neuron. Neurons that do not communicate with other LPUs and the internal connectivity patterns defined between neurons within an LPU are not made accessible through the LPU's interface (Fig. 3). Neurokernel's LPU API distinguishes between ports that transmit spikes or graded-potential neuron states in order to enable transmission of both. It should be noted that the current LPU interface is not intended to be final; we anticipate its gradual extension to more accurately account for the range of neural interactions found in the fly's connectome.

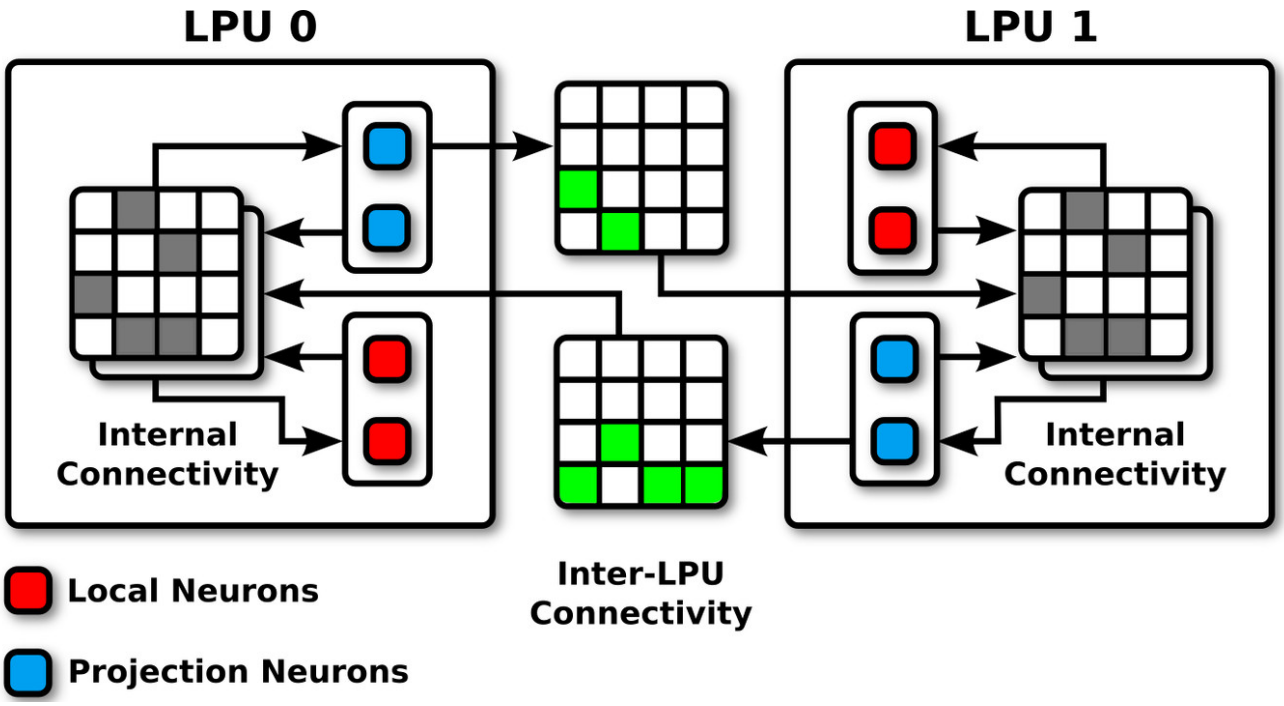

Figure 3: LPU components and interface. Internal connectivity patterns (gray) consist of the axons of local neurons (red), the axons of projection neurons (blue) that connect both to neurons within and external to an LPU, and the synapses associated with all dendrites within the LPU; these patterns are not externally accessible. Inter-LPU connectivity patterns (green) exclusively describe the connectivity patterns of the axons of projection neurons in one LPU that project to neurons in another LPU.

Neurokernel also defines a representation for inter-LPU connectivity patterns; these correspond to the tracts described in Fig. 2. By separating the internal processing logic of an LPU from its external connectivity, Neurokernel enables the testing of different hypotheses regarding internal and external connectivity patterns; for example, different models of a

single LPU can be evaluated when "plugged into" the same external connectivity fabric. A connectivity pattern between two LPUs is represented as two incidence matrices that respectively describe the presence (or absence) of axons from projection neurons in each LPU to neurons in the other LPU; Neurokernel provides a Python class for storing this connectivity information as a sparse multidimensional array. In addition to their use in determining how much data to transmit between LPUs, the number of neurons exposed by each LPU is also used by Neurokernel to validate connections defined between it and other LPUs; an LPU that exposes specific numbers of spiking and graded potential ports may only be connected to other LPUs using connectivity patterns that are compatible with those numbers. Although spiking and graded potential ports must be separately identified, the LPU API permits one to define connections between either neuron type. LPU designers must ensure that their models can properly process both transmitted spikes and graded potentials; likewise, each LPU must also implement the synapses associated with the dendrites that correspond to an LPU's input ports.

In addition to the classes required to represent LPUs and connectivity patterns, Neurokernel provides two additional classes that provide various services required to manage instances of LPUs and connectivity patterns: (1) The communications broker class constructs a routing table using specified LPU connectivity data and uses it to automatically transfer data between LPUs at each step of model execution; this obviates the need for implemented LPUs to explicitly invoke the communication mechanism that actually propagates data between LPUs. (2) The emulation manager class provides methods for incorporating LPU instances into an emulation and connecting them with connectivity patterns using the communications broker class. Once fully instantiated, the emulation can be run for a specified interval by the manager class.

Apart from the API requirements discussed above, Neurokernel currently places no explicit restrictions upon an LPU model's contents, how it interacts with available GPUs, or the topology of interconnections between different LPUs; compatible LPUs and inter-LPU patterns may be arbitrarily composed to construct subsystems (Fig. 4). Neurokernel's current service classes support instantiation and execution of a brain emulation on a single computer containing multiple GPUs. Communication between LPU instances in a running emulation is currently performed using ZeroMQ[3], a highly flexible networking stack with low end-to-end latency; this will enable future extension of Neurokernel to run emulations on computer clusters.

2.4 Using the Neurokernel Framework

In addition to advertising the numbers of graded potential and spiking ports it exposes, every LPU must provide a `run_step()` method that processes incoming neuron state data from source modules, updates the states of the module's neurons, and provides the updated states of its exposed projection neurons for transmission to destination modules:

```
from neurokernel.core import Module
```

[3] http://zeromq.org

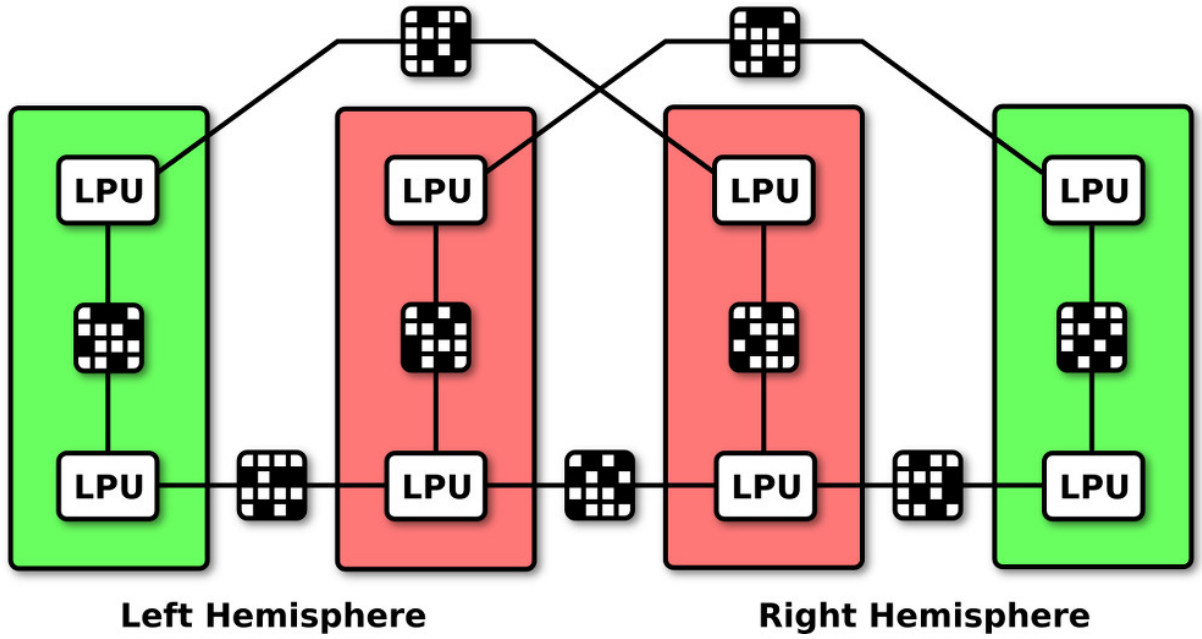

Figure 4: Hierarchy of modeling components in a Neurokernel fly brain model. LPUs and connectivity patterns may be composed into subsystems (red, green) which may in turn be connected to other subsystems.

```
class MyLPU0(Module):
    gpot_data = np.zeros(5, float)
    spike_data = np.zeros(10, int)

    # Numbers of graded potential and spiking ports:
    @property
    def N_gpot(self):
        return len(self.gpot_data)
    @property
    def N_spike(self):
        return len(self.spike_data)

    # Process incoming data and set outgoing data:
    def run_step(self, in_gpot_dict, in_spike_dict,
                 out_gpot, out_spike):
        super(MyModule, self).run_step(in_gpot_dict, in_spike_dict,
                                       out_gpot, out_spike)

        # Use incoming data from `in_gpot_dict` and
        # `in_spike_dict` to compute presynaptic inputs to internal neurons:
        # ...

        # Set contents of `out_gpot` and `out_spike`
        # using updated projection neuron states:
```

```
    # ...
```

Neuron state data generated by each LPU is propagated to the relevant destination LPUs by Neurokernel before the next iteration of the emulation's execution. LPU implementations are responsible for transferring generated data for graded potential and spiking neurons from GPU memory to the arrays `out_gpot` and `out_spike`, respectively. Since updated graded neuron states must be propagated at each step of model execution, the length of `out_gpot` is fixed to the number of graded potential neurons exposed by the LPU; the index of each array element identifies the originating neuron. The `out_spike` array contains the indices of those spiking neurons that have emitted spikes at the current model execution step; its length may therefore change during model execution. Data transmitted to an LPU is accumulated in the dictionaries `in_gpot_dict` and `in_spike_dict` for use in the LPU's internal computations; the keys of this dictionary correspond to the originating LPUs' unique identifiers, while the values possess the same structure as `out_gpot` and `out_spike`.

Connectivity patterns between LPUs may be constructed directly in Python using the data structure described in § 2.3. For example, one may define the presence of feed-forward connections from all output ports exposed by one LPU to all input ports exposed by another as follows; note that the connections may be referenced specifically in terms of the source and destination port types. For simplicity, Neurokernel does not require that a port be explicitly labeled as receiving input or emitting output; it is up to the LPU designer to determine whether to produce or consume data via a port.

```
import numpy as np
from neurokernel.core import Connectivity
conn = Connectivity(m1.N_gpot, m1.N_spike,
                    m2.N_gpot, m2.N2_spike, 1,
                    m1.id, m2.id)
conn[m1.id, 'all', :, m2.id, 'all', :] = \
    np.ones((m1.N_gpot+m1.N_spike,
             m2.N_gpot+m2.N_spike))
```

In addition to manually specifying inter-LPU connectivity patterns using the above data structure, Neurokernel supports loading connectivity patterns from GEXF[4] files. A GEXF file that describes a connectivity pattern between the ports exposed by two LPUs respectively named `A` and `B` must contain a multigraph that comprises two groups of nodes labeled `A:0, A:1, ...` and `B:0, B:1, ...`, with edges between these groups defining the connections between the LPUs. Neurokernel also supports loading of internal neural and synaptic model parameters from GEXF files; GEXF representations of the LPU models described in § 3 are included with the Neurokernel source code. Inter-LPU connections currently remain static throughout an emulation; future versions of Neurokernel will support dynamic instantiation and removal of connections while a model is being executed.

After all LPUs and connectivity patterns are instantiated, the emulation may be launched as follows:

```
from neurokernel.core import Manager
```

[4] http://gexf.net

```
man = Manager()
man.add_brok()
m0 = man.add_mod(MyLPU0(man.port_data, man.port_ctrl))
m1 = man.add_mod(MyLPU1(man.port_data, man.port_ctrl))
man.connect(m0, m1, conn)

# Set emulation to run a specific number of execution steps:
duration = 10.0
dt = 1e-2
man.start(int(duration/dt))
man.stop()
```

When a Neurokernel emulation is launched, the `run_step` method is invoked at each time step of a model's execution by Neurokernel's emulation manager.

3 Results

To demonstrate and test Neurokernel's support for interfacing functional brain modules, we have used it to integrate independently implemented models of LPUs in the olfactory and vision sensory systems constructed using experimentally obtained connectivity information. These models are briefly described below; IPython notebooks containing more information about the models and the results of their execution are available on the Neurokernel website (http://neurokernel.github.io/docs.html).

3.1 Olfactory System Model

The early olfactory system in *Drosophila* consists of two antennal lobes, one on each side of the fly brain. Each of these LPUs contain 49 glomeruli that differ in functionality, size, shape, and relative position. Each glomerulus receives axons from about 50 olfactory receptor neurons (ORNs) on each of the fly's two antennae that express the same odorant receptor. The axons of each ORN connect to the dendrites of 3 to 5 projection neurons (PNs) in the glomeruli. In addition to the PNs - which transmit olfactory information to the higher regions of the brain - the antennal lobes contain local neurons (LNs) whose connections are restricted to the lobes; inter-glomerular connectivity therefore subsumes synaptic connections between ORNs and PNs, ORNs and LNs, LNs and PNs, and feedback from PNs to LNs. The entire early olfactory system in *Drosophila* contains approximately 4000 neurons. [75]

The current model of the each antennal lobe comprises 49 glomerular channels with full intraglomerular connectivity in both hemispheres of the fly brain. The entire model comprises 2800 neurons, or 70% of the fly's entire antennal lobe. All neurons in the system are modeled using the Leaky Integrate-and-Fire (LIF) model [37] and all synaptic currents elicited by spikes are modeled using alpha functions [63]. Parameters for 24 of the glomerular channels are based upon currently available ORN type data [22]; all other channels are configured with artificial parameters. In accordance with Neurokernel's module API, all projection neurons comprised by the glomeruli may emit output visible to

other LPUs; the local interneurons that connect glomeruli do not emit any output to other LPUs. A more detailed description of the implementation of the fruit fly olfactory system is available at http://nbviewer.ipython.org/github/neurokernel/neurokernel/blob/master/notebooks/olfaction.ipynb.

3.2 Vision System Model

In addition to the retina where the photo-transduction takes place, the optic lobe of the *Drosophila* can be divided into 4 major LPUs on each side of the fly brain respectively referred to as the lamina, medulla, lobula and lobula plate. Visual information progresses along a processing path that starts at the retina and successively passes through the lamina, medulla, and either the lobula or the lobula plate. The spatial structure of the visual stimulus is preserved by the retinotopic columnar organization of most of these LPUs.

There are at least 120 different types of neurons in the optic lobe [17]. Most of the neurons in the optic lobe (if not all) do not emit spikes; rather, they communicate via chemical synapses where neurotransmitter is tonically released based on the graded potential of the presynaptic neurons. The synapses can have varying amount of delays based on the different neurotransmitters. Many neurons in the optic lobe also communicate through gap junctions.

The current vision system model is based upon available connectome data for the lamina [67] and medulla [17, 23]. The model consists of two LPUs; the first contains 9516 neurons (or about 90% of the cells) in the retina and lamina, while the second contains 6920 (or about 17% of the cells) in the medulla and several neurons that connect to both the medulla and first layer of the lobula. All neurons are modeled using the Morris-Lecar model [50] with parameters selected to not elicit spiking activity. Synapses are modeled using a simple model of tonic neurotransmitter release and its effect upon postsynaptic conductance. The model does not currently comprise gap junctions. A more detailed description of the implementation of the fruit fly olfactory system is available at http://nbviewer.ipython.org/github/neurokernel/neurokernel/blob/master/notebooks/vision.ipynb.

3.3 Subsystem Integration Models

To illustrate integration of individual sensory LPUs into functional subsystems, we connected the vision and olfactory models described above to an independently developed LPU that detects correlated visual and olfactory stimulus events (Fig. 5). This LPU contains 8 spiking neurons modeled as LIF neurons that receive inputs from 8 wide field tangential neurons in the medulla model; the latter innervate 4 vertical and 4 horizontal groups of columns in the medulla. It should be noted that the integration circuit does not explicitly model a specific known biological LPU. Rather, it demonstrates Neurokernel's ability to enable experimentation using multiple LPUs independently designed by different researchers. A snapshot of the model's response to multimodal stimuli is depicted in Fig. 6; an animation of the model's output may be viewed at http://nbviewer.ipython.org/github/neurokernel/neurokernel/blob/master/notebooks/sensory_integration.ipynb.

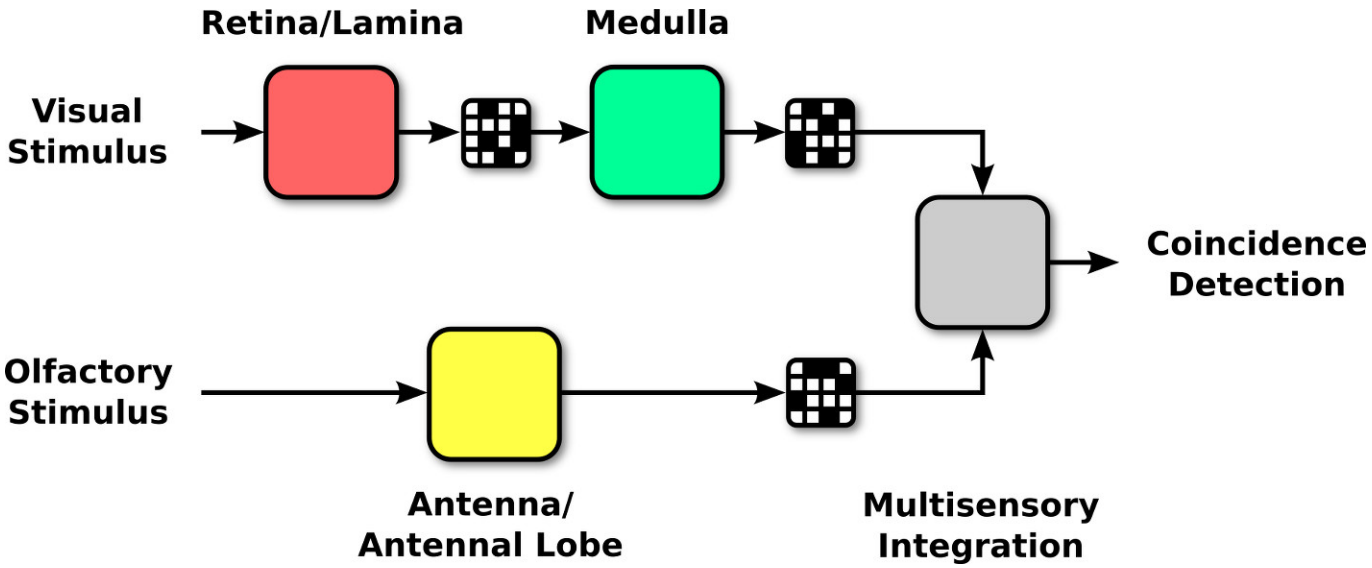

Figure 5: Integration of vision and olfactory models. The outputs of independently developed LPUs (retina/lamina and medulla in the vision system and antenna/antennal lobe in the olfactory system) are fed to a multisensory integration LPU that responds to correlated visual and olfactory stimulus events.

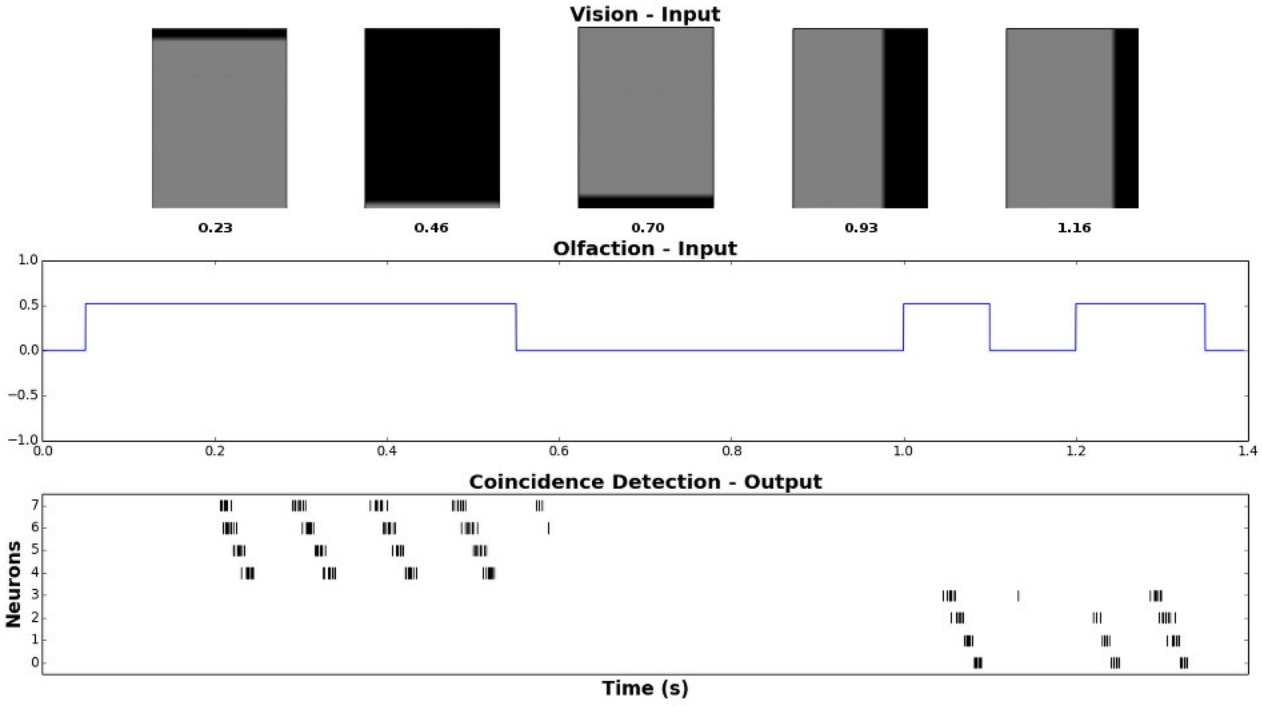

Figure 6: Snapshot of multisensory integration model output. The first row depicts snapshots of a visual stimulus, while the second depicts a profile of the olfactory stimulus. The raster plot on the third row depicts the output of a coincidence detection model that responds to concurrent events associated with odorant and visual stimuli.

4 Discussion

Neuromorphic platforms whose design is directly inspired by the brain have the potential to execute large-scale models with limited functionality at speeds that significantly exceeds those achievable with traditional von Neumann computer architectures [70, 72, 64, 15, 62]. The limited availability and potentially high costs of the various neuromorphic hardware systems currently under development, however, constrains the growth of the development communities needed to facilitate the use and improvement of such platforms. Although the usability of neuromorphic platforms is increasing due to the support of high-level software interfaces such as PyNN [13], they still afford less programming flexibility than commodity computing platforms. GPU-based neural emulation platforms such as Neurokernel, by contrast, capitalize on a cost-effective parallel computing resource whose use in a wide range of applications enables computational neuroscientists to avail themselves of the flourishing GPU software development community to support the increasingly complex models that will be needed to accurately emulate the fly brain. Such platforms also stand to benefit from the already-rapid improvement in GPU performance driven by the increasingly high computational requirements of computer games.

There exist a wide range of general-purpose neural simulation packages [8, 21] that provide highly flexible programming interfaces that permit the quantitative description of a wide range of neural circuit models in terms of individual or homogeneous populations of neurons and synapses. The limited support currently afforded by these packages for GPU hardware has spurred the recent development of an array of GPU-based simulators for spiking neural networks [53, 16, 52, 55]. Most of these simulators are capable of executing spiking neural network models containing numbers of neurons and synapses comparable to the numbers found in the fruit fly brain. Additionally, some of these simulators have also demonstrated that the execution time of large networks that cannot currently be executed in real time can be appreciably reduced by partitioning the networks across multiple GPUs; for example, one such simulator exhibits a speed increase of almost 20% when simulating 100 seconds of activity if the number of GPUs used is increased from 32 to 160 [48]. In light of continuous improvements in both GPU performance and the achievable data transfer throughput between GPUs, we project that near-real-time performance for a whole fly brain model comprising both spiking and graded potential neurons using a number of GPUs within the above range will be achievable in a few years time given widely predicted architectural improvements in GPU technology.

Despite the impressive performance GPU-based spiking neural network software can achieve for simulations comprising increasingly large numbers of neurons and synapses, they and other simulator packages do not adequately address the architectural complexity of modeling the entire fly brain either in the design phase when multiple designers need to combine their respective contributions or in the run time phase when the emulated system must efficiently avail itself of available computational resources. Just as the architectural complexity of a PC is quite different from that of a microprocessor and hence requires quite different tools and methodologies in design, so too does an emulation of the entire fly brain require a platform that provides services and scalability that go beyond the features of cur-

rent general-purpose neural circuit simulators. Neurokernel aims to remedy this lacuna by providing both the high-level APIs needed to modularly develop whole brain models and the low-level computational substrate required to enable those models' implementations to scalably leverage multiple GPUs.

In light of their increasing availability and low costs, there is growing interest in leveraging the power of multiple GPUs to support increasingly large-scale neural simulations [74, 54, 48]. We believe that the current trajectory of GPU technology points to the development of an increasingly asynchronous parallel computing platform in which multiple GPUs constitute a "first-class" resource that are not architecturally subordinated to a computer's microprocessor [29]. In addition to obtaining the near-term advantages of using currently available GPU technology to accelerate execution of neural models, Neurokernel aims to provide a software foundation that can be extended to support anticipated improvements in GPU technology.

A fundamental modeling limitation of existing neural simulator packages is their lack of support for studying emergent functions that cannot be exclusively attributed to a single circuit or module in the brain. Fly locomotion, for instance, is evidently mediated by premotor processing that integrates multiple sensory modalities [78]. The difficulty of modeling such behaviors is compounded by the incompatibility of independently developed models of specific processing units in the brain. Although resources for model sharing [24, 20] and interoperability [19] have facilitated the sharing of computational neural models, researchers who wish to build upon existing models must still manually extract and reimplement the relevant portions of those models to incorporate into their work. This lack of interoperability severely hampers the extent to which neuroscientists can progress from modeling individual neural circuits to emulation of the entire brain. Neurokernel will mitigate this problem by ensuring compatibility between LPU models that adhere to the APIs it provides.

A mechanism for connecting neural elements using interface ports that support communication of both spike and analog data was implemented in the PCSIM neural simulator [60]. Such elements can also encapsulate internal implementations written using other simulation packages. The software also enabled the specification of sets of multiple connections between elements. Although PCSIM is able to scale over multiple CPUs and can in principle enable encapsulation and integration of GPU-based circuit models, the software itself does not natively support the use of GPUs and provides no infrastructure for scaling across multiple GPUs. Additionally, development of PCSIM appears to have stalled as of 2010.

Neurokernel bears similarities to other ongoing efforts to develop connectome-based emulations of the neural circuitry of other relatively tractable model organisms. The OpenWorm project[5], for instance, is currently developing several open software technologies for modeling of the entire neuromuscular system of the nematode *Caenorhabditis elegans* [59]. It capitalizes on the extremely small number of neurons in the worm's nervous system and the full reconstruction of its connectome [79]. While reconstruction of the fly connectome is still in progress, we anticipate that ongoing advances in our understanding of the fly brain's connectivity [12, 10, 44, 73] will provide an increasing amount of information that can be

[5] http://www.openworm.org/

incorporated into models of the fly's brain.

5 Future Development

Having implemented the APIs within Neurokernel's application plane that enable integration of independently developed models of LPUs and connectivity patterns, our next major goal is to implement the APIs exposed by Neurokernel's control and compute planes that will enable construction of LPU models without having to explicitly implement LPU models using Python and CUDA. To this end, we are incorporating support for a modular model specification format that will capitalize on the identification of canonical circuits in the fly's sensory systems [39] to enable the construction of LPU models in terms of predefined circuits rather than populations of neurons and synapses [9]. The mechanism responsible for loading models expressed in this format will generate the GPU code needed to execute the models.

Although Neurokernel currently permits brain models to make use of multiple locally hosted GPUs, it requires programmers to explicitly manage the GPU resources used by a model's implementation. We aim to implement a prototype GPU resource allocation component to obviate the need to explicitly select and manage individual GPUs when constructing a fly brain model. This mechanism will permit experimentation with different allocation policies as LPU models become more complex. We are also extending Neurokernel's communication mechanism to take advantage of NVIDIA's GPUDirect technology for accelerated communication between GPUs [56, 57] to improve the performance of brain emulations that require the use of multiple GPU resources.

The future availability of new connectome and experimental data will necessitate the modification and reevaluation of existing fly brain models. As Neurokernel's model execution efficiency enables model execution to approach the time scale of the actual fly brain, researchers will be able to compare the input-output characteristics of specific virtual neurons in a running model with their counterparts in the live fly's brain using increasingly precise electrophysiological techniques and novel system identification techniques [33, 35, 40, 41, 42, 43]. This will enable researchers to perform real-time in vivo validation of a model's correctness (Fig. 7).

The scope of Neurokernel's goals and the need to support the revision of models in light of new data requires a structured means of advancing the framework's design. To this end, the Neurokernel project employs Requests for Comments documents (RFCs) as a tool both for advancing the design of Neurokernel architecture and the LPU models built with it. First employed in the development of the Internet [5] and used more recently in the development of the Python programming language in the guise of Python Enhancement Proposals (PEPs) [77], RFCs are publicly accessible design proposals for major new components or features of a complex computing system made by the system's developers and other interested parties in order to elicit feedback (or "comments"). Once published, an RFC may be superseded by a newer RFC that incorporates modifications made in response to feedback. This process enables RFCs to both serve as authoritative specifications of a system's design (if accepted for implementation) and an archive of that system's design development (if rejected or su-

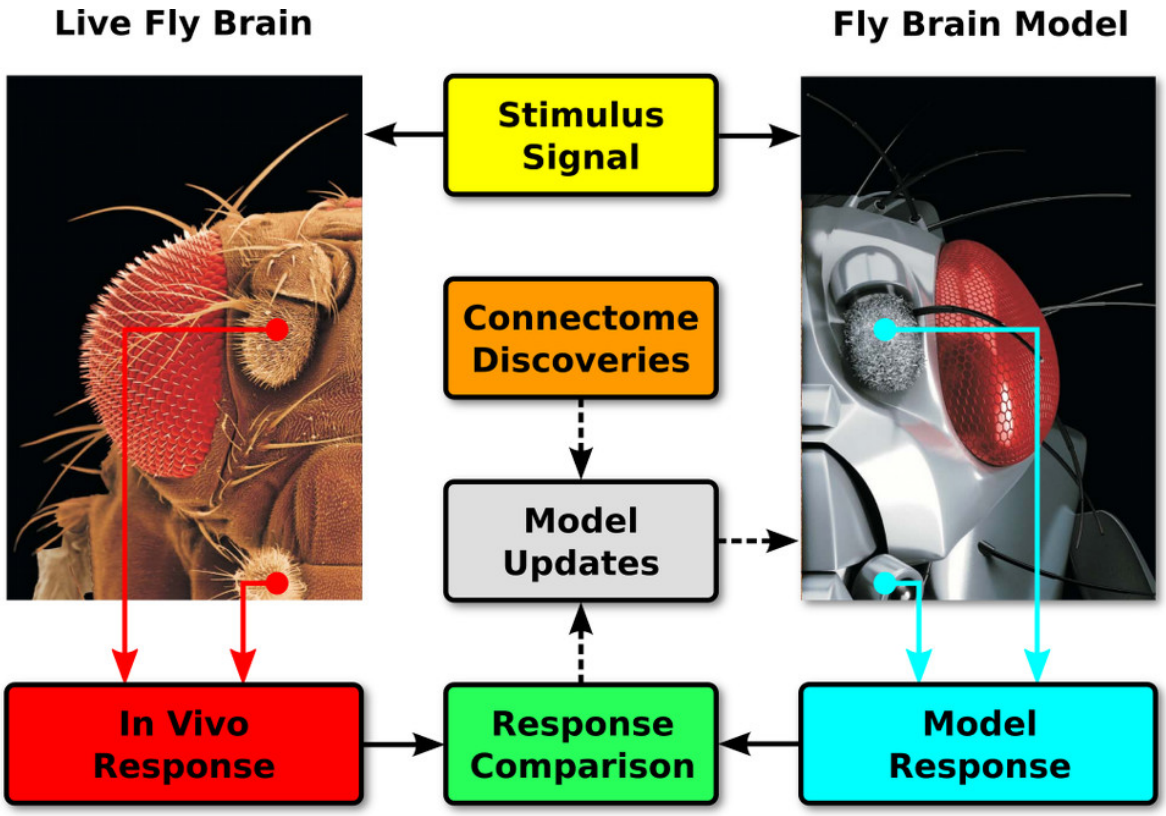

Figure 7: In vivo validation is essential to the development of accurate fly brain models. Neural responses to sensory stimuli are recorded from the live fly brain in real time and compared to the computed responses of the corresponding components in a fly brain model executed by Neurokernel on the same time scale. Discrepancies between these responses and new connectome data may be used to improve the model's accuracy (fly photograph adapted from Berger and fly robot image adapted from Vizcano, Benton, Gerber, and Louis, both reproduced with permission).

perseded by a newer RFC). All current and future Neurokernel RFCs will be available on the project website http://neurokernel.github.io/docs.html.

6 Conclusion

Development of an accurate model of the entire fruit fly brain has the potential to provide important insights essential to successfully modeling the human brain. Despite the fly brain's relative numerical tractability, its successful emulation is an ambitious goal that will require the joint efforts of multiple researchers from different disciplines. Neurokernel's open design, support for commodity parallel computing technology, and ability to integrate independently developed models of the brain's functional subsystems all facilitate this joining of forces. The framework's first release is a step in this direction; we expect and anticipate that aspects of

the current design such as connectivity structure and module interfaces will be superseded by newer designs informed by the growing body of knowledge regarding the structure and function of the fly brain. We invite the research community to join this effort on Neurokernel's GitHub website [6] and development mailing list [7].

7 Acknowledgements

This work was supported in part by the AFOSR under grant #FA9550-12-10232, in part by the NIH under grant #R021 DC012440001, and in part by Columbia University's Electrical Engineering Department. The authors would like to thank Nikul H. Ukani, Chung-Heng Yeh, and Yiyin Zhou for developing the sensory system and integration models used to test the software and Daniel S. Chevitarese for useful discussions. The authors would also like to thank Juergen Berger for kindly permitting reuse of his fly photograph and thank Nacho Vizcano, Richard Benton, Bertram Gerber, and Matthieu Louis for permitting reuse of the robot fly image they composed for the ESF-EMBO 2010 Conference on Functional Neurobiology in Minibrains.

References

[1] A. Paul Alivisatos, Miyoung Chun, George M. Church, Ralph J. Greenspan, Michael L. Roukes, and Rafael Yuste. The brain activity map project and the challenge of functional connectomics. *Neuron*, 74(6):970–974, June 2012.

[2] Rajagopal Ananthanarayanan, Steven K. Esser, Horst D. Simon, and Dharmendra S. Modha. The cat is out of the bag: cortical simulations with 10^9 neurons, 10^{13} synapses. In *Proceedings of SC 2009*, November 2009.

[3] Michael Arbib. Modular models of brain function. *Scholarpedia*, 2(3):1869, 2007.

[4] J. Douglas Armstrong and Jano I. van Hemert. Towards a virtual fly brain. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 367(1896):2387 –2397, June 2009.

[5] S. Bradner. The internet standards process - revision 3. *Internet RFCs, ISSN 2070-1721*, RFC 2026, October 1996.

[6] Romain Brette and Dan F. M. Goodman. Simulating spiking neural networks on GPU, December 2012.

[7] Seth A. Budick and Michael H. Dickinson. Free-flight responses of Drosophila melanogaster to attractive odors. *Journal of Experimental Biology*, 209(15):3001 –3017, 2006.

[6] http://neurokernel.github.io/

[7] https://lists.columbia.edu/mailman/listinfo/neurokernel-dev

[8] Nicholas T Carnevale and Michael L Hines. *The NEURON book*. Cambridge University Press, Cambridge; New York, 2005.

[9] Daniel S. Chevitarese, Lev E. Givon, Aurel A. Lazar, and Marley Vellasco. CircuitML: a modular language for modeling local processing units in the Drosophila brain. In *Frontiers in Neuroinformatics. Conference Abstract: Neuroinformatics 2013*, Aug 2013.

[10] Ann-Shyn Chiang, Chih-Yung Lin, Chao-Chun Chuang, Hsiu-Ming Chang, Chang-Huain Hsieh, Chang-Wei Yeh, Chi-Tin Shih, Jian-Jheng Wu, Guo-Tzau Wang, and Yung-Chang Chen. Three-dimensional reconstruction of brain-wide wiring networks in Drosophila at single-cell resolution. *Current Biology*, 21(1):1–11, January 2011.

[11] M. Eugenia Chiappe, Johannes D. Seelig, Michael B. Reiser, and Vivek Jayaraman. Walking modulates speed sensitivity in Drosophila motion vision. *Current Biology*, 20(16):1470–1475, August 2010.

[12] Dmitri B. Chklovskii, Shiv Vitaladevuni, and Louis K. Scheffer. Semi-automated reconstruction of neural circuits using electron microscopy. *Current Opinion in Neurobiology*, 20(5):667–675, October 2010.

[13] Andrew P. Davison, Daniel Brüderle, Jochen Eppler, Jens Kremkow, Eilif Muller, Dejan Pecevski, Laurent Perrinet, and Pierre Yger. PyNN: a common interface for neuronal network simulators. *Frontiers in Neuroinformatics*, 2:11, 2009.

[14] Joseph B. Duffy. GAL4 system in Drosophila: a fly geneticist's Swiss Army knife. *Genesis (New York, N.Y.: 2000)*, 34(1-2):1–15, October 2002. PMID: 12324939.

[15] Chris Eliasmith, Terrence C. Stewart, Xuan Choo, Trevor Bekolay, Travis DeWolf, Yichuan Tang, and Daniel Rasmussen. A large-scale model of the functioning brain. *Science*, 338(6111):1202–1205, November 2012.

[16] A.K. Fidjeland and M.P. Shanahan. Accelerated simulation of spiking neural networks using GPUs. In *Neural Networks (IJCNN), The 2010 International Joint Conference on*, pages 1–8, 2010.

[17] K.-F. Fischbach and A.P.M. Dittrich. The optic lobe of Drosophila melanogaster. I. a Golgi analysis of wild-type structure. *Cell and Tissue Research*, 258(3), December 1989.

[18] Mark A. Frye and Michael H. Dickinson. Closing the loop between neurobiology and flight behavior in Drosophila. *Current Opinion in Neurobiology*, 14(6):729–736, December 2004.

[19] Padraig Gleeson, Sharon Crook, Robert C. Cannon, Michael L. Hines, Guy O. Billings, Matteo Farinella, Thomas M. Morse, Andrew P. Davison, Subhasis Ray, Upinder S. Bhalla, Simon R. Barnes, Yoana D. Dimitrova, and R. Angus Silver. NeuroML: a language for describing data driven models of neurons and networks with a high degree of biological detail. *PLoS Comput Biol*, 6(6):e1000815, June 2010.

[20] Padraig Gleeson, Eugenio Piasini, Sharon Crook, Robert Cannon, Volker Steuber, Dieter Jaeger, Sergio Solinas, Egidio D'Angelo, and R. Angus Silver. The Open Source Brain Initiative: enabling collaborative modelling in computational neuroscience. *BMC Neuroscience*, 13(Suppl 1):O7, July 2012.

[21] Dan F. M. Goodman and Romain Brette. The Brian simulator. *Frontiers in Neuroscience*, September 2009.

[22] Elissa A. Hallem and John R. Carlson. Coding of odors by a receptor repertoire. *Cell*, 125(1):143–160, April 2006.

[23] Charles M. Higgins, John K. Douglass, and Nicholas J. Strausfeld. The computational basis of an identified neuronal circuit for elementary motion detection in dipterous insects. *Visual Neuroscience*, 21(04):567–586, 2004.

[24] Michael L. Hines, Thomas Morse, Michele Migliore, Nicholas T. Carnevale, and Gordon M. Shepherd. ModelDB: a database to support computational neuroscience. *Journal of Computational Neuroscience*, 17(1):7–11, August 2004. PMID: 15218350.

[25] Stephen J Huston and Vivek Jayaraman. Studying sensorimotor integration in insects. *Current Opinion in Neurobiology*, 21(4):527–534, August 2011.

[26] Eugene M. Izhikevich and Gerald M. Edelman. Large-scale model of mammalian thalamocortical systems. *Proceedings of the National Academy of Sciences*, 105(9):3593 –3598, March 2008.

[27] Eric Jones, Travis Oliphant, and Pearu Peterson. SciPy: open source scientific tools for Python, 2001.

[28] Eric R. Kandel, Henry Markram, Paul M. Matthews, Rafael Yuste, and Christof Koch. Neuroscience thinks big (and collaboratively). *Nature Reviews Neuroscience*, 14(9):659–664, September 2013.

[29] Shinpei Kato, Michael McThrow, Carlos Maltzahn, and Scott Brandt. Gdev: First-class GPU resource management in the operating system. In *4th USENIX Workshop on Hot Topics in Parallelism*, volume 12, June 2012.

[30] Alexander Y. Katsov and Thomas R. Clandinin. Motion processing streams in Drosophila are behaviorally specialized. *Neuron*, 59(2):322–335, July 2008.

[31] Anmo J. Kim, Aurel A. Lazar, and Yevgeniy B. Slutskiy. 2D encoding of concentration and concentration gradient in Drosophila ORNs. In *Computational and Systems Neuroscience Meeting*, Salt Lake City, Utah, February 2010.

[32] Anmo J. Kim, Aurel A. Lazar, and Yevgeniy B. Slutskiy. System identification of DM4 glomerulus in the Drosophila antennal lobe using stationary and non-stationary odor stimuli. In *CNS*2010*, San Antonio, TX, USA, July 2010.

[33] Anmo J. Kim, Aurel A. Lazar, and Yevgeniy B. Slutskiy. Drosophila projection neurons encode the acceleration of time-varying odor waveforms. In *Computational and Systems Neuroscience Meeting 2011*, Salt Lake City, Utah, Feb 2011.

[34] Anmo J. Kim, Aurel A. Lazar, and Yevgeniy B. Slutskiy. Investigating odor identity encoding in Drosophila OSNs. In *Computational and Systems Neuroscience Meeting 2011*, Salt Lake City, Utah, Feb 2011.

[35] Anmo J. Kim, Aurel A. Lazar, and Yevgeniy B. Slutskiy. System identification of Drosophila olfactory sensory neurons. *Journal of Computational Neuroscience*, 30(1):143–161, August 2011.

[36] Andreas Klöckner, Nicolas Pinto, Yunsup Lee, Bryan Catanzaro, Paul Ivanov, and Ahmed Fasih. PyCUDA and PyOpenCL: a scripting-based approach to GPU run-time code generation. *Parallel Computing*, 38(3):157–174, March 2012.

[37] Louis Lapicque. Recherches quantitatives sur l'excitation électrique des nerfs traitée comme une polarisation. *J. Physiol. Pathol. Gen.*, 9:620–635, 1907.

[38] Aurel A. Lazar. Programming telecommunication networks. *IEEE Network*, 11(5):8–18, October 1997.

[39] Aurel A. Lazar, Wenze Li, Nikul H. Ukani, Chung-Heng Yeh, and Yiyin Zhou. Neural circuit abstractions in the fruit fly brain. In *Society for Neuroscience Abstracts*, Nov 2013.

[40] Aurel A. Lazar and Yevgeniy B. Slutskiy. Identifying dendritic processing. In *Advances in Neural Information Processing Systems 23*, pages 1261–1269. J. Lafferty and C. K. I. Williams and J. Shawe-Taylor and R.S. Zemel and A. Culotta, 2010.

[41] Aurel A. Lazar and Yevgeniy B. Slutskiy. Channel identification machines. *Journal of Computational Intelligence and Neuroscience*, 2012:1–20, July 2012.

[42] Aurel A. Lazar and Yevgeniy B. Slutskiy. Multisensory encoding, decoding, and identification. In *Advances in Neural Information Processing Systems 26*. C.J.C. Burges, L. Bottou, M. Welling, Z. Ghahramani and K.Q. Weinberger, December 2013.

[43] Aurel A. Lazar and Yevgeniy B. Slutskiy. Functional identification of spike-processing neural circuits. *Neural Computation*, 26(2), February 2014.

[44] Chih-Yung Lin, Chao-Chun Chuang, Tzu-En Hua, Chun-Chao Chen, Barry J. Dickson, Ralph J. Greenspan, and Ann-Shyn Chiang. A comprehensive wiring diagram of the protocerebral bridge for visual information processing in the Drosophila brain. *Cell Reports*, 3(5):1739–1753, May 2013.

[45] Gaby Maimon, Andrew D. Straw, and Michael H. Dickinson. A simple vision-based algorithm for decision making in flying Drosophila. *Current Biology*, 18(6):464–470, March 2008.

[46] Matthew S. Maisak, Juergen Haag, Georg Ammer, Etienne Serbe, Matthias Meier, Aljoscha Leonhardt, Tabea Schilling, Armin Bahl, Gerald M. Rubin, Aljoscha Nern, Barry J. Dickson, Dierk F. Reiff, Elisabeth Hopp, and Alexander Borst. A directional tuning map of Drosophila elementary motion detectors. *Nature*, 500(7461):212–216, August 2013.

[47] Henry Markram. The Blue Brain Project. *Nat Rev Neurosci*, 7(2):153–160, February 2006.

[48] K. Minkovich, C.M. Thibeault, M.J. O'Brien, A. Nogin, Y. Cho, and N. Srinivasa. HRLSim: a high performance spiking neural network simulator for GPGPU clusters. *IEEE Transactions on Neural Networks and Learning Systems*, 25(2):316–331, 2014.

[49] Javier Morante and Claude Desplan. The color-vision circuit in the medulla of Drosophila. *Current Biology*, 18(8):553–565, April 2008.

[50] C. Morris and H. Lecar. Voltage oscillations in the barnacle giant muscle fiber. *Biophysical Journal*, 35(1):193–213, July 1981.

[51] Laiyong Mu, Kei Ito, Jonathan P. Bacon, and Nicholas J. Strausfeld. Optic glomeruli and their inputs in drosophila share an organizational ground pattern with the antennal lobes. *The Journal of Neuroscience*, 32(18):6061–6071, May 2012. PMID: 22553013.

[52] Jim Mutch, Ulf Knoblich, and Tomaso Poggio. CNS: a GPU-based framework for simulating cortically-organized networks. Technical Report MIT-CSAIL-TR-2010-013, MIT, 2010.

[53] Jayram Moorkanikara Nageswaran, Nikil Dutt, Jeffrey L. Krichmar, Alex Nicolau, and Alexander V. Veidenbaum. A configurable simulation environment for the efficient simulation of large-scale spiking neural networks on graphics processors. *Neural Networks*, 22(5-6):791–800, July 2009.

[54] Andrew Nere, Sean Franey, Atif Hashmi, and Mikko Lipasti. Simulating cortical networks on heterogeneous multi-GPU systems. *Journal of Parallel and Distributed Computing*, 2012. Article in press.

[55] Thomas Nowotny. Flexible neuronal network simulation framework using code generation for NVidia® CUDA(TM). *BMC Neuroscience*, 12(Suppl 1):P239, July 2011. PMID: null PMCID: PMC3240344.

[56] NVIDIA. CUDA Toolkit 4.0 Readiness for CUDA Applications, March 2011.

[57] NVIDIA. Kepler GK110 whitepaper, 2012.

[58] Travis Oliphant. *Guide to NumPy*. Trelgol Publishing, 2006.

[59] Andrey Palyanov, Sergey Khayrulin, Stephen D. Larson, and Alexander Dibert. Towards a virtual C. elegans: a framework for simulation and visualization of the neuromuscular system in a 3D physical environment. *In Silico Biology*, 11(3):137–147, 2012. PMID: 22935967.

[60] Dejan Pecevski, Thomas Natschläger, and Klaus Schuch. PCSIM: a parallel simulation environment for neural circuits fully integrated with Python. *Frontiers in Neuroinformatics*, 3:11, 2009.

[61] F. Perez and B.E. Granger. IPython: a system for interactive scientific computing. *Computing in Science Engineering*, 9(3):21 –29, June 2007.

[62] Robert Preissl, Theodore M. Wong, Pallab Datta, Myron Flickner, Raghavendra Singh, Steven K. Esser, William P. Risk, Horst D. Simon, and Dharmendra S. Modha. Compass: a scalable simulator for an architecture for cognitive computing. In *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*, SC '12, pages 54:1–54:11, Los Alamitos, CA, USA, 2012. IEEE Computer Society Press.

[63] W. Rall. Distinguishing theoretical synaptic potentials computed for different soma-dendritic distributions of synaptic input. *Journal of Neurophysiology*, 30(5):1138–1168, September 1967.

[64] Alexander D. Rast, Xin Jin, Francesco Galluppi, Luis A. Plana, Cameron Patterson, and Steve Furber. Scalable event-driven native parallel processing: the SpiNNaker neuromimetic system. In *Proceedings of the 7th ACM international conference on Computing frontiers*, CF '10, page 21–30, New York, NY, USA, 2010. ACM. ACM ID: 1787279.

[65] Micah Richert, Jayram Moorkanikara Nageswaran, Nikil Dutt, and Jeffrey L. Krichmar. An efficient simulation environment for modeling large-scale cortical processing. *Frontiers in Neuroinformatics*, 5:19, 2011.

[66] Jens Rister, Dennis Pauls, Bettina Schnell, Chun-Yuan Ting, Chi-Hon Lee, Irina Sinakevitch, Javier Morante, Nicholas J. Strausfeld, Kei Ito, and Martin Heisenberg. Dissection of the peripheral motion channel in the visual system of Drosophila melanogaster. *Neuron*, 56(1):155–170, October 2007.

[67] Marta Rivera-Alba, Shiv N. Vitaladevuni, Yuriy Mishchenko, Zhiyuan Lu, Shin-Ya Takemura, Lou Scheffer, Ian A. Meinertzhagen, Dmitri B. Chklovskii, and Gonzalo G. de Polavieja. Wiring economy and volume exclusion determine neuronal placement in the Drosophila brain. *Current Biology*, 21(23):2000–2005, December 2011.

[68] Joshua R. Sanes and S. Lawrence Zipursky. Design principles of insect and vertebrate visual systems. *Neuron*, 66(1):15–36, April 2010.

[69] Johannes D. Seelig and Vivek Jayaraman. Feature detection and orientation tuning in the Drosophila central complex. *Nature*, advance online publication, October 2013.

[70] Rae Silver, Kwabena Boahen, Sten Grillner, Nancy Kopell, and Kathie L. Olsen. Neurotech for neuroscience: Unifying concepts, organizing principles, and emerging tools. *The Journal of Neuroscience*, 27(44):11807 –11819, October 2007.

[71] Zhuoyi Song, Marten Postma, Stephen A. Billings, Daniel Coca, Roger C. Hardie, and Mikko Juusola. Stochastic, adaptive sampling of information by microvilli in fly photoreceptors. *Current Biology*, 22(15):1371–1380, June 2012.

[72] Terrence C. Stewart, Bryan Tripp, and Chris Eliasmith. Python scripting in the Nengo simulator. *Frontiers in Neuroinformatics*, 3:7, 2009.

[73] Shin-Ya Takemura, Arjun Bharioke, Zhiyuan Lu, Aljoscha Nern, Shiv Vitaladevuni, Patricia K. Rivlin, William T. Katz, Donald J. Olbris, Stephen M. Plaza, Philip Winston, Ting Zhao, Jane Anne Horne, Richard D. Fetter, Satoko Takemura, Katerina Blazek, Lei-Ann Chang, Omotara Ogundeyi, Mathew A. Saunders, Victor Shapiro, Christopher Sigmund, Gerald M. Rubin, Louis K. Scheffer, Ian A. Meinertzhagen, and Dmitri B. Chklovskii. A visual motion detection circuit suggested by Drosophila connectomics. *Nature*, 500(7461):175–181, August 2013.

[74] C.M. Thibeault, R. Hoang, and F.C. Harris, Jr. A novel multi-GPU neural simulator. In *Proceedings of 3rd International Conference on Bioinformatics and Computational Biology 2011*, New Orleans, LA, March 2011.

[75] Leslie B. Vosshall and Reinhard F. Stocker. Molecular architecture of smell and taste in Drosophila. *Annual Review of Neuroscience*, 30(1):505–533, 2007.

[76] Trevor J. Wardill, Olivier List, Xiaofeng Li, Sidhartha Dongre, Marie McCulloch, Chun-Yuan Ting, Cahir J. O'Kane, Shiming Tang, Chi-Hon Lee, Roger C. Hardie, and Mikko Juusola. Multiple spectral inputs improve motion discrimination in the Drosophila visual system. *Science*, 336(6083):925–931, May 2012. PMID: 22605779.

[77] Barry Warsaw, Jeremy Hylton, David Goodger, and Nick Coghlan. PEP purpose and guidelines. *Python Enhancement Proposals*, PEP 1, June 2000.

[78] Jan Wessnitzer and Barbara Webb. Multimodal sensory integration in insects—towards insect brain control architectures. *Bioinspiration & Biomimetics*, 1(3):63–75, September 2006.

[79] J. G. White, E. Southgate, J. N. Thomson, and S. Brenner. The structure of the nervous system of the nematode Caenorhabditis elegans. *Philosophical Transactions of*

the Royal Society of London. B, Biological Sciences, 314(1165):1–340, November 1986. PMID: 22462104.

[80] Rachel I. Wilson. Understanding the functional consequences of synaptic specialization: insight from the Drosophila antennal lobe. *Current Opinion in Neurobiology*, 21(2):254–260, April 2011.