Vision Project

By Patrick Clare, Max Miller, and Nicole Rivilis

UNIs: pc2620, mlm2290, njr2133

Introduction:

In this project we more or less successfully implemented a parallelized model simulating the vision system of *Drosophila melanogaster*, the common fruit fly. Using GPUs and parallel programming, we implemented 768 photoreceptor neurons in 128 ommatidia, based off papers by Song et al.[1][2] The model takes in an input of photons and uses realistic models of the phenomena of light absorption and biochemical signal cascade, as well as the well-established Hodgkin-Huxley neuron model, to translate photons into an electrical pulse.

The Model:

A brief explanation of the science behind the model, though of course the actual model can be found in the two Song papers. A light stimulus is somehow fed to the photoreceptors. Based on Poisson statistics, photons are absorbed by certain microvilli within certain photoreceptors. Photon interactions can also result in failures, which means the photon is absorbed by no photoreceptor. Within a microvillus, an absorbed photon activates a metarhodopsin, the first molecule in the signal cascade. The signal cascade goes through at least 6 different molecules, each in turn being activated and deactivated. Eventually the final molecule opens cation selective ion channels, which allow influxes and outfluxes of Na^+, Ca^{2+}, and Mg^{2+}, creating "light-induced current," or LIC. The LICs of the 30,000 microvilli per photoreceptor are summed to one LICtotal. Each photoreceptor has its own light-independent basic neuron, which takes the total LIC as input and outputs a voltage trace.

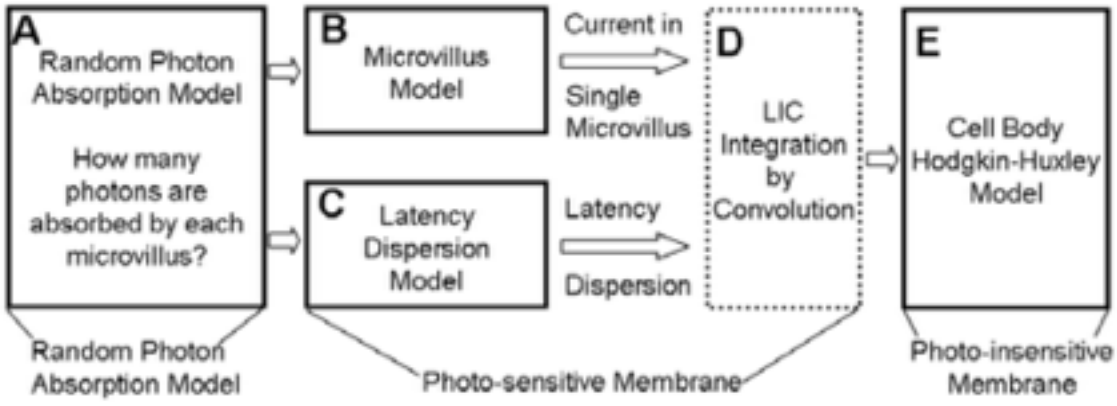

Figure 1: Diagram of general photoreceptor model. From Song et al. 2009.

We implemented the photon absorption model using basic Poisson statistics. Photons are processed every 0.1 milliseconds. Each microvillus has a fairly equal chance of absorbing each photon. Each signal molecule's state in every microvillus is contained in a state vector we called X. When a photon is absorbed, that microvillus's metarhodopsin (M*) number increases by 1. Then, every microvillus has that 0.1 millisecond to go through the signal cascade. A random time step (within the 0.1 millisecond) is determined, and a signal cascade reaction (1 of 12 possible ones) is also randomly determined. Calcium dynamics are also modeled independently, according to the proscribed calcium equations. At the end of the 0.1 millisecond, each microvillus's changes are accepted, and the total output currents of each photoreceptor are calculated. The current of each photoreceptor then goes to its base neuron, which we implemented using a modified Hodgkin-Huxley neuron (graduated potential model, not spiking). This outputs a voltage, which is meant to be eventually fed to the lamina for visual processing.

Development:

The first thing we did was to get the Hodgkin-Huxley neuron functioning on the GPU. This was fairly simple, though it took us a bit of time to get all the kinks worked out.

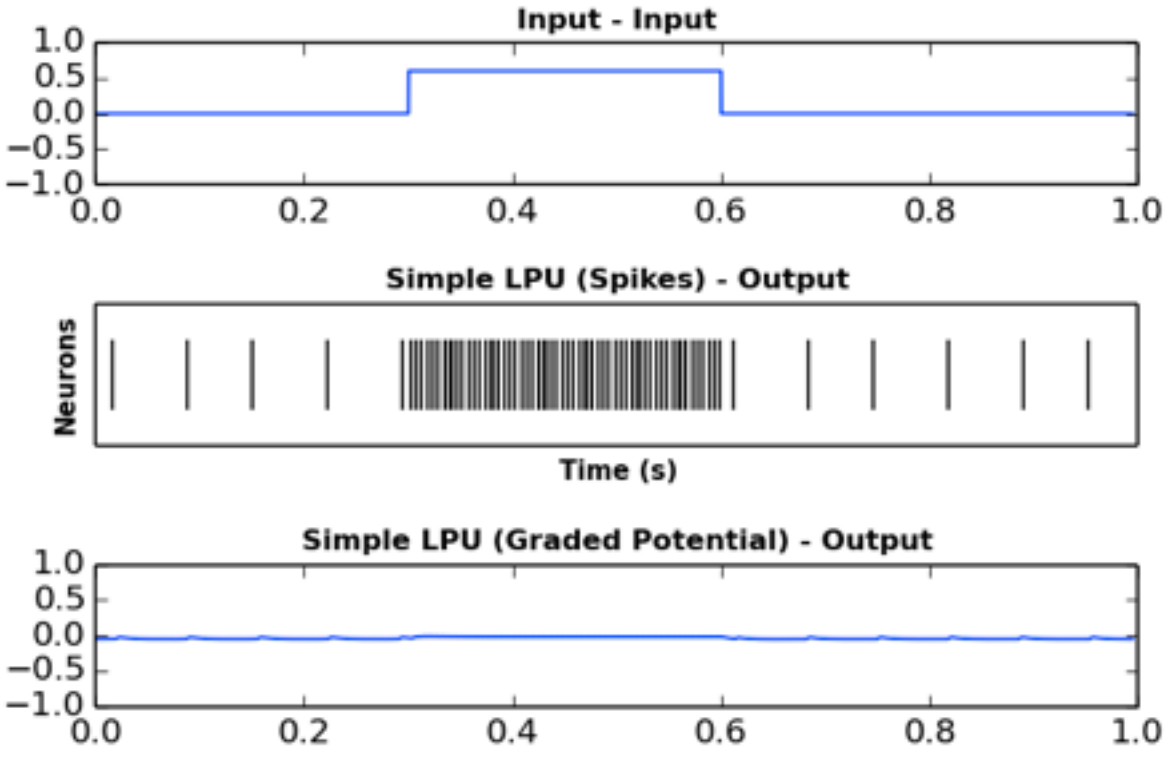

Figure 2: Spiking and graduated potential outputs of the simulated Hodgkin-Huxley

After this, we implemented the various differential equations of the two Song papers into code. This process was initially done in MATLAB, and eventually moved over to PyCUDA after we determined our model was accurate. Coding for the photon absorption model was

fairly simple, but the signal cascade and calcium dynamics models were no simple task, even for a single microvillus. (We mainly based our code off the 2012 paper and supplemental information; in the three years between the two papers, Song et al. greatly simplified and tightened up their model.) There were certainly some difficulties with this, considering there are several ambiguities in the Song papers about the model. One particular example was going between concentrations and absolute numbers for calcium, calmodulin, and C*; adjusting the equations for concentrations vs. numbers became very confusing. Certain parameters were given in the wrong units and orders of magnitude, and it was not easy determining which should be changed. Finally though, with some help from our fellow students, we were able to get a model working in MATLAB producing similar results to the papers.

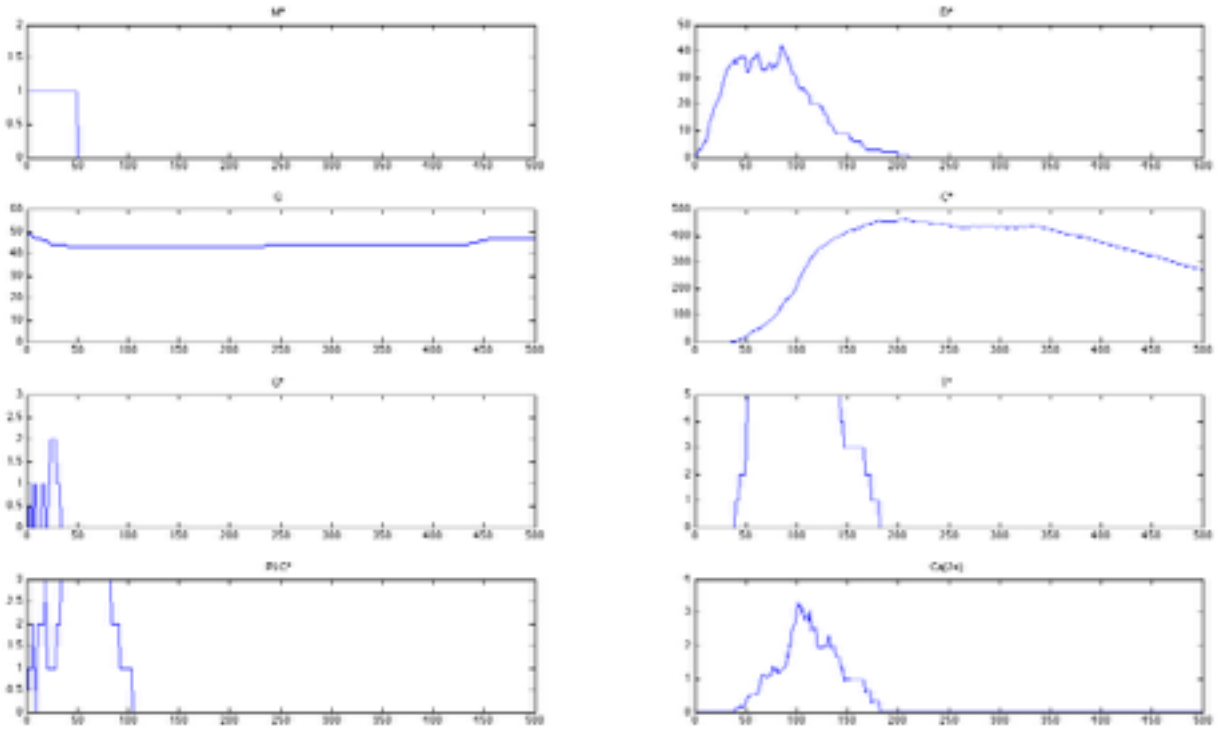

Figure 3: Signal Cascade in MATLAB. Left column, from top to bottom: M*, G, G*, PLC*. Right column, from top to bottom: D*, C*, T*, Ca^{2+} (concentration in mM). Uses a constant photon input as opposed to any dynamic input.

The T* bump directly produces the current pulse, similar in shape, which is then fed into the neuron.

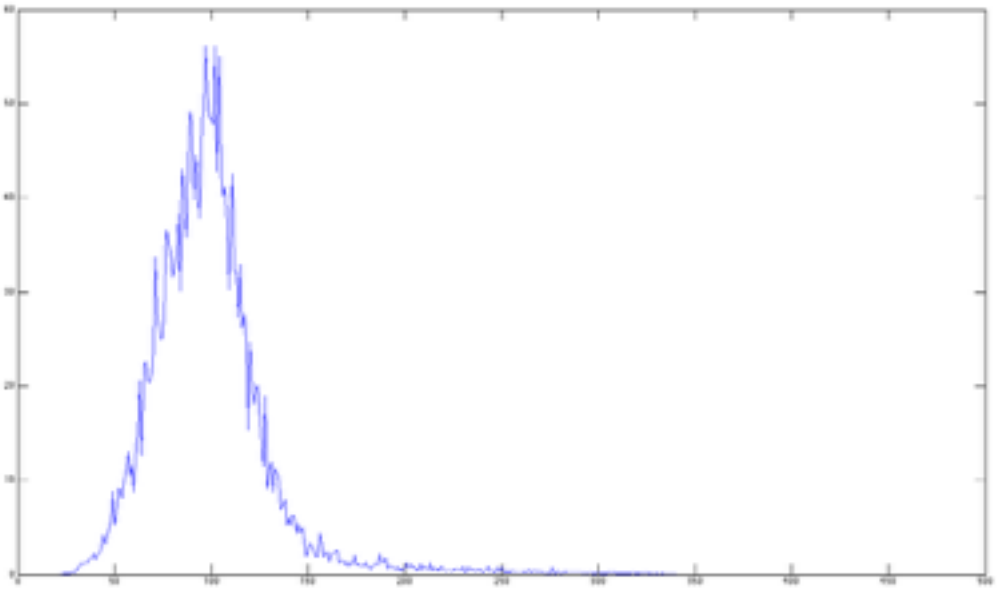

Figure 4: Current pulse resulting from above signal cascade.

After the model was working for a single microvillus, there were two massively parallel components: summing the 30,000 microvilli of a photoreceptor to a single current to be fed to the neuron, and then each of our 768 photoreceptors acting independently. This is where the GPU comes into play. Moving it over to a combination of Python and PyCUDA wasn't terribly difficult, though we did not have too much previous experience with GPU programming. Each MATLAB function had to translated to a CUDA kernel, and then the outer program (in Python) had to be written from scratch, though it was based on the Neurokernel model.

While translating we found several bugs in our model (evidenced by non-functionality) that helped in perfecting our signal cascade. Once a single photoreceptor with 30,000 microvilli was up and running, we moved to running several photoreceptors in parallel; we encountered several other GPU-related bugs, based on block and thread indexing and memory allocation. Luckily we had the TAs to help us through this portion. The GPU is not especially communicative when an error is thrown, so it took a lot of combing through the code to find all the bugs. Eventually, however, the code ran properly. We were able to get it to run with a total of 128 ommatidia, 768 photoreceptors. Because of memory constraints discovered on the final days of debug, we weren't able to fully realize the model of 768 ommatidia, 6,144 photoreceptors. However, given more tinkering (or waiting 18 months for Moore's Law), we believe our code would scale fully.

We chose to encapsulate all of the microvilli per photoreceptor in a single thread block to simplify CUDA memory indexing. While the GPU supports up to 1024 threads per block, we found due to other bottlenecks the most we could get away with was 512 per block. Thus within each kernel we looped over the 512 threads with offsets to account for the total 30000 microvilli. This allowed us to use the neuron number as the block index for these kernels.

To get a closer look at our development methodology, feel free to explore our Github fork of the neurokernel: <www.github.com/noukernel/neurokernel>, named after 脳 (nou), the Japanese word for brain.

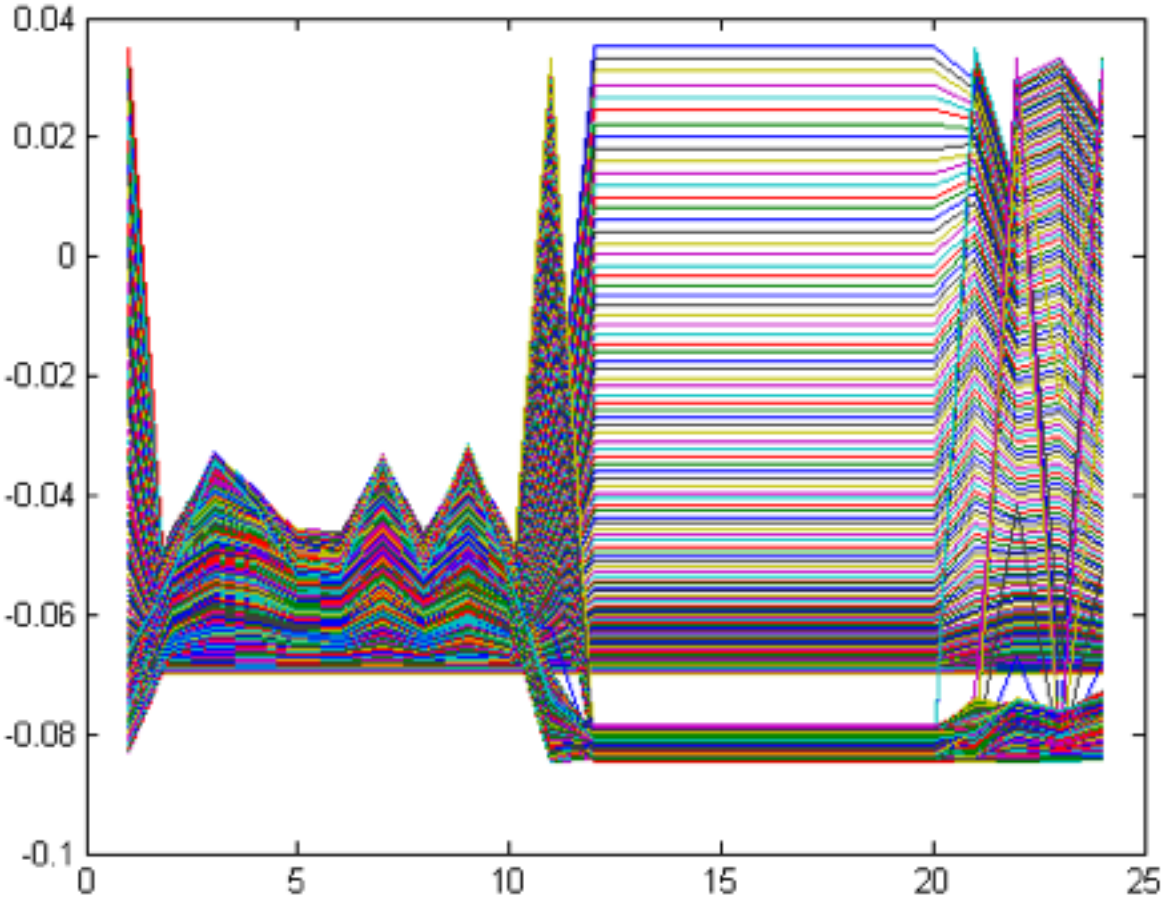

Figure 5: 24 photoreceptors processing the video input file. Clearly shows parallel, individual spiking amongst neurons.

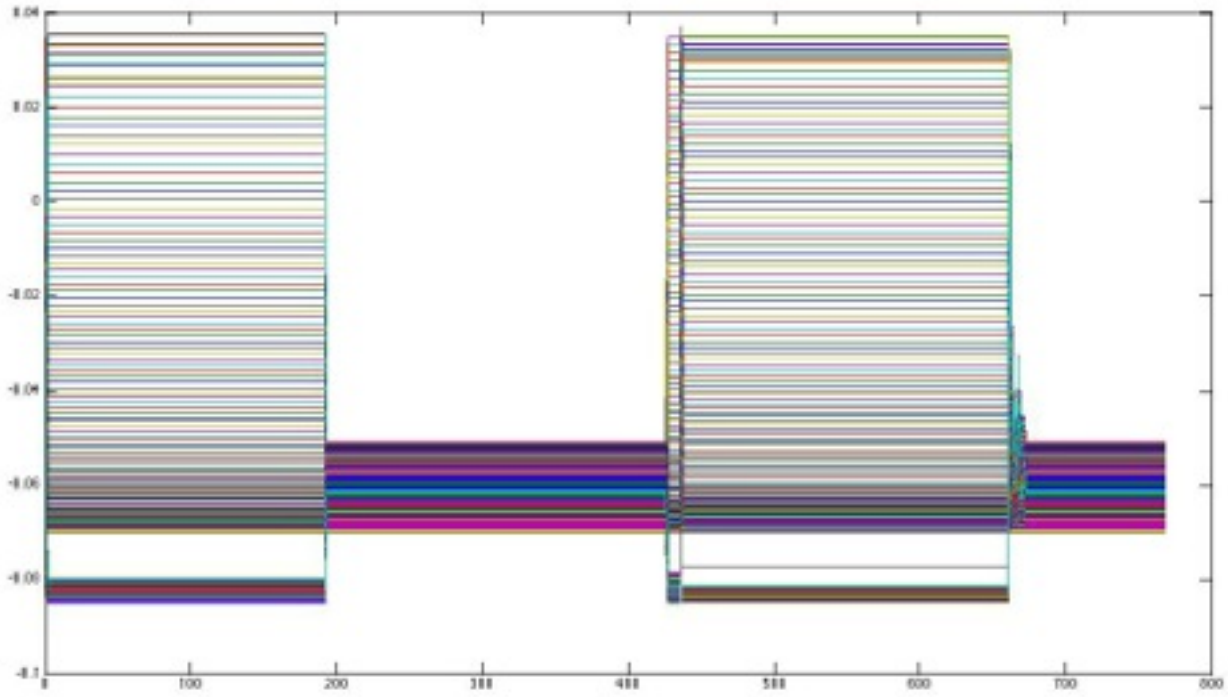

Figure 6: Voltage traces of 768 neurons. Here, the spiking pattern is less easy to categorize. Not positive of the accuracy of this.

Individual Contributions:

Initially we divided work based on each teammate's experience with the languages. Max focused on getting the signal cascade functioning in MATLAB, Patrick focused on translating to pyCUDA, and Nicole focused on helpfully bridging the gap with her knowledge of Python. However, as everything was shifted to GPU, Max and Nicole both also worked on aspects of the GPU code, especially the signal cascade and calcium dynamics kernels. Patrick remained the authority on most of the GPU implementation design. The debugging was highly collaborative and was greatly helped by the TAs, particularly Yiyin and Nikul.

Analysis

Overall, we succeeded in simulating a portion of the fly's vision system. We unfortunately did not have time to connect our photoreceptors to the lamina, which does the image processing. Had we done this, we would have been able to really see the video input file via our model photoreceptors and their processing. However, it is clear from running our program that each photoreceptor is spiking individually and of its own accord based on the light from its own area of the video input. Comparing the signal cascade figures to those of Song, we can definitively say we implemented those differential equations properly. Of course, with more time we could have proceeded with up to 6,144 neurons total, and a connection with the image processing stage of the lamina.

Our model comparison of a single functioning microvilli in MATLAB seems to closely resemble Song's results. However, it's difficult to extrapolate what a fully functional retina would produce without doing the calculations. In early stages we tried to do this, extrapolating the results of a single microvillus to 30,000, but after concerning results we found it to not make much sense. At such a scale, we were basically declaring all 30,000 microvilli as either simultaneously absorbing or not absorbing photons, which goes against the assumption made by Song et al. that the number of photons absorbed is based on Poisson statistics, on the order of 1000.

As for our CUDA results, we attained similar functionality as MATLAB, and we were able to observe 768 unique photoreceptor results. So while our victory may not be in our complete accuracy in simulating a fly's visual system, there was a technical victory in our newfound grasp of the power and intricacies of GPGPU programming.

References:

1- Zhuoyi Song, Marten Postma, Stephen Billings, Daniel Coca, Roger C. Hardie, and Mikko Juusola. Stochastic, adaptive sampling of information by microvilli in fly photoreceptors. *Current Biology*, 22:1371-1380, 2012.

2- Zhouyi Song, Daniel Coca, Stephen Billings, Marten Postma, Roger C. Hardie, and Mikko Juusola. Biophysical modeling of a drosophila photoreceptor. In ChiSing Leung, Minho Lee, and Jonathan H. Chan, editors, *Neural Information Processing*, volume 5863 of *Lecture Notes in Computer Science*, pages 57-71. Springer Berlin Heidelberg, 2009.