

- A Distributed Saliency System using Ethernet AER (see Section ??)

# 1 A Distributed Saliency System using Ethernet AER

**Participants:** David Gamez, Sean Taffler, Tobi Delbruck, Filip Ponulak

## 1.1 Introduction

Models of bottom up saliency have been popular in recent years, with the work of Itti [2] being a key example. However there is a lot of evidence to suggest that there is a strong top-down influence on eye movements when viewing natural scenes - for example early work by Yarbus [4] showed that the trajectories followed by the gaze depend on the task that the observer is performing. This combination of top-down and bottom-up influence on saliency is shown in Figure 1.

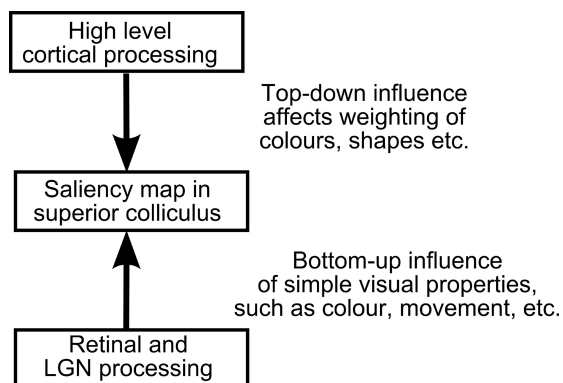


Figure 1: Vision architecture

The primary aim of this project was to build a system that used a combination of top-down and bottom-up influences to calculate the saliency of a scene in a layer of neurons controlling eye movements. In this system the neurons would be influenced in a bottom-up way by the luminance of the visual scene and by higher top-down processes that interpret the image in a more meaningful way and either inhibit or increase the lower level saliency. A secondary aim of this project was to explore the possibility of integrating hardware and software neurons by streaming spikes across a LAN in real time. The use of Ethernet to stream spike events is relatively new and one part of this project was to explore the feasibility and performance of this approach. As the project progressed and problems with the hardware interfaces were encountered, this method for streaming spikes became an increasingly important way of communicating between the different saliency maps.

## 1.2 System Components

The saliency system in this project integrated a number of different components:

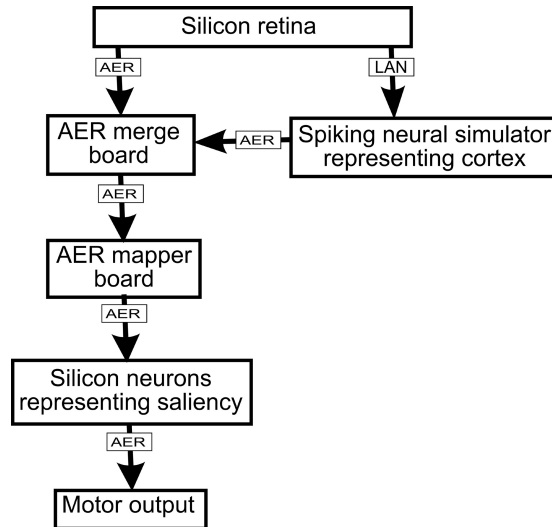


Figure 2: First AER-based architecture

1) Silicon retina. As a source of data for this system we used Tobi Delbruck’s silicon retina, which provides a stream of events linked to changes in luminance in the image. This is much more biologically realistic than a stream of images and it could be connected up either to AER or USB.

2) Spiking neural simulator. To calculate the cortical processing, we used the spiking neural simulator created by David Gamez and adapted it to use the learning algorithm for spiking neural networks developed by Filip Ponulak [3] (also see the other project in this report: ‘Classification of Visual Stimuli with Spiking Neural Networks’). A screen shot of this simulator is shown in Figure 8.

3) Silicon neurons. For the saliency map itself we used the silicon neurons developed by Giacomo Indiveri and Srinjoy Mitra.

4) AER communication. We planned to use this to interface between the hardware that was generating the bottom-up saliency system.

The way in which these different components were integrated into a single system is described in the next section.

### 1.3 Development of the Architecture

Our first plan for the architecture was to use AER for the bottom-up event stream from the camera to the silicon neurons and to use the streaming of spike events over LAN to connect the silicon retina to the spiking neural simulator. Once the spiking simulator had calculated the top-down saliency, it was intended that the information would be injected back into the AER system, where it could selectively excite or inhibit the silicon neurons. This architecture is shown in Figure 2.

During the first week of the workshop we started work on this system, wrote C++ wrapper classes to enable the simulator to inject AER events and created the spike

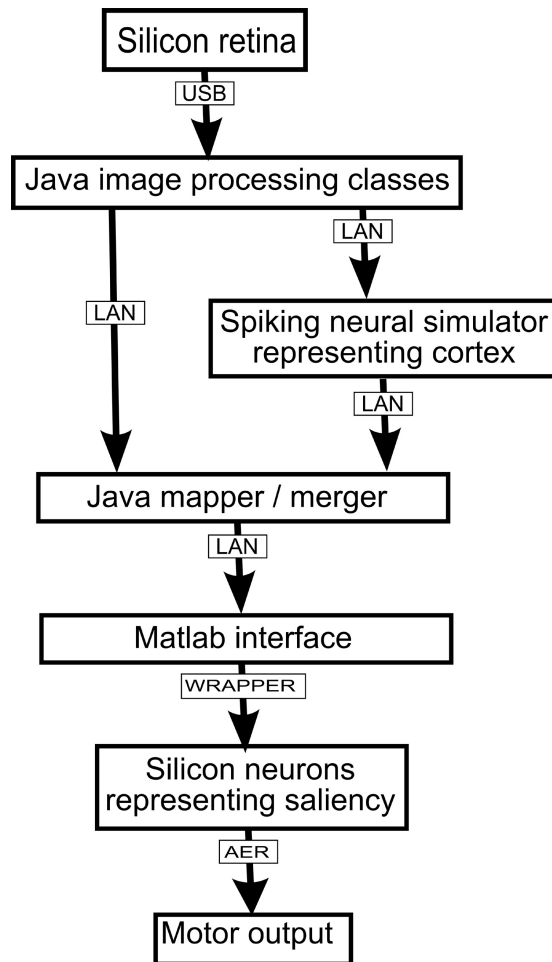


Figure 3: Second architecture based around Ethernet AER

streaming architecture. However, a major problem with this architecture appeared when the silicon neuron board proved to be unable to communicate with the AER boards from the University of Sevilla. This made it impossible to send a stream of merged AER events to the silicon neurons. By this stage the live spike streaming over LAN was working well and so it was decided to base the communications architecture around this technology and explore how effectively this form of 'Ethernet AER' could be used to create a distributed neural system. The final architecture is shown in Figure 3.

In this system the role of the Java mapper/ merger is to listen to the camera spike broadcast group and map the 128 x 128 stream of spikes onto a 2 x 2 array suitable for stimulating the silicon neurons. This software also listens to the saliency broadcast group and combines this data into a UDP stream that is passed to a Matlab wrapper

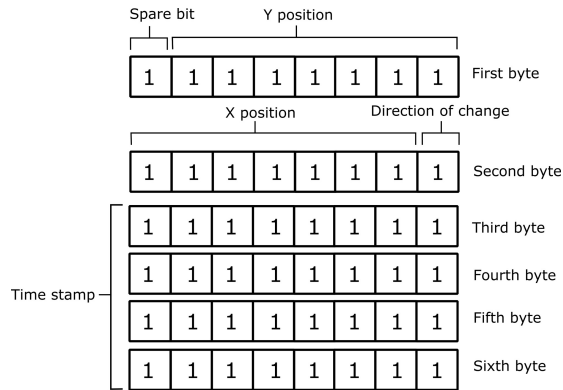


Figure 4: Protocol for spike events within UDP packets

that interfaces with the silicon neurons.

## 1.4 Spike Streaming

The Ethernet AER spike streaming was developed by this project to enable distributed hardware and software simulations to communicate with each other across a standard LAN. This approach has the advantage that it is extremely flexible and can take advantage of networking tools and protocols that have been developed and optimized over many years. To send these spike events over the network, it was decided to use the stateless UDP network broadcast protocol instead of TCP. Although UDP does not guarantee the order of arrival of the spike packets, we were more interested in developing a 'live' system that passed data as quickly as possible, rather than a simulation system that offers complete repeatability.

The great advantage of network broadcast is that it enables different processes across the network to start and stop completely asynchronously without any explicit communication between them. For example, if a process is broadcasting its spikes on group 230.0.0.1, port 4445, then any other process can listen in to these spikes simply by opening an appropriate socket. The same process can then broadcast its output to a different address, for example 230.0.0.2, which in turn can be received by an arbitrary number of processes. In a simulation environment this makes it very easy for different processes to send and receive different spike streams in an extremely flexible manner. This approach has the further advantage that it is not limited by the local network and it would even be possible to use the UDP network broadcast protocol for live collaboration between different institutions.

The protocol for the data passed inside the UDP packets was developed by Tobi Delbruck as part of his work developing Java classes to process the data from the silicon retina. Each spike event is represented by six bytes with the structure shown in Figure 4.

One advantage of the spike streaming approach is the large number of tools that are available for Ethernet monitoring and management. This makes it easy to see what

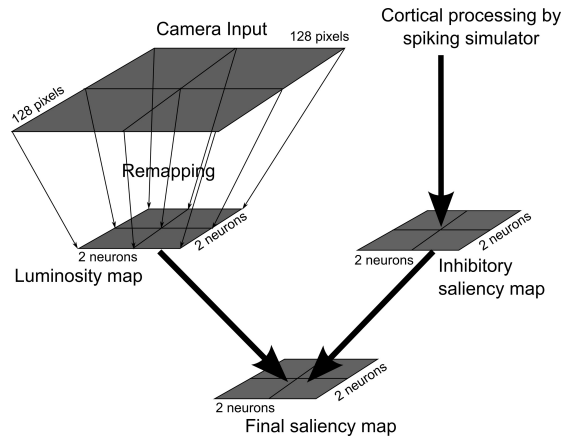


Figure 5: Bottom-up neural architecture

is going on in the spike stream and to record and play back spikes. In this project we used Ethernet to sniff the spike packets from the network and record them to files, and TCPReplay to play the packets back over the network. The latter tool also offers highly flexible rewriting of the packets: making it possible to send a recorded stream of spikes to a different broadcast group, change the port, and so on.

To evaluate the performance of the spike streaming approach we wrote a couple of simple Java classes to send and receive spike packets over the network. The only network available to us for this project was the Telluride workgroup network. This was a 100 MB/s LAN with other traffic present as well and so the results from these tests should be seen as a lower limit for this approach.

## 1.5 Experiments

In the limited time available for this project, there was not time to create a sophisticated bottom-up saliency system and so we elected to use a simple luminosity map in which the brightest point was the most salient. Because of the limited number of silicon neurons, we divided the visual field into four quadrants, with a neuron representing the saliency of each quadrant. The spike activity from the camera for each quadrant was then merged down into a luminosity signal for the neuron representing that quadrant. This luminosity-driven activity was then combined with a 2 x 2 top-down saliency map that was also topographically mapped onto the camera quadrants. This top down saliency map inhibited the neurons in the luminosity map, creating a final output that was a combination of the bottom-up and top-down saliency signals. This neural architecture is shown in Figure 5.

The top-down cortical network within the spike simulator consisted of an input layer of 128 x 128 neurons driven by the camera signal. This was fully connected to two 2 x 2 maps, one of which learnt to recognize crosses in the input signal, and the other learnt to recognize squares in the input signal. A second set of 2x2 maps was created to provide a teaching signal for the learning maps. This neural architecture is

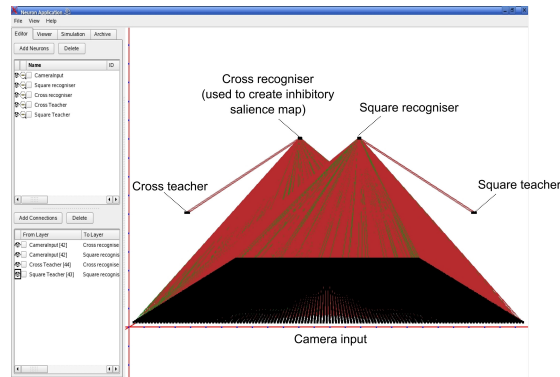


Figure 6: Top-down neural architecture

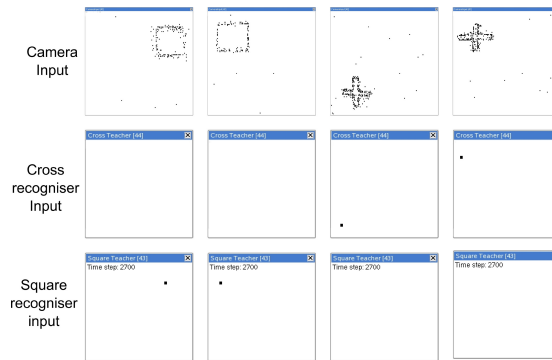


Figure 7: Training data for top-down neural network (note that the images of spikes have been compressed for this illustration, when in fact they were much more distributed over time)

shown in Figure 6.

To train the cortical network we used *Ethereal* to record spike streams from the silicon retina when there was a square or cross in one of the four quadrants of the visual field. These spike streams were then played back to the neural network along with a teaching signal consisting of activation in one of the four quadrants when there was either a cross or a square in that part of the visual field. The *ReSuMe* learning algorithm in the neurons and synapses then looked for correlations between the visual input and the training signal and adjusted the synaptic weights accordingly. Although separate layers were trained up to recognize squares and crosses, in the final experiments only the output from the cross recognizer was used to generate the inhibitory saliency map shown in Figure 5. Some examples of the training data are given in Figure 7.

During the testing phase, the system was exposed to either crosses or squares in each of its four quadrants and the direction of the saccade was recorded.

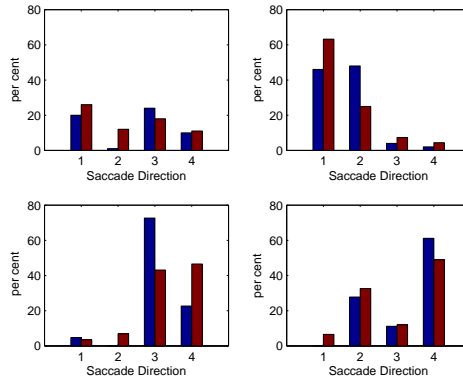


Figure 8: Saccade directions for different stimulus positions. Each graph is for the presentation of the stimulus in a particular quadrant of the visual field and the columns show the proportion of saccades to each portion of the visual field

## 1.6 Results

After training of the cortical neural network it was found that each of the neurons within the cross recognizer map responded with a higher firing rate when there was a cross in their quadrant of the visual field, and with a lower firing rate when there was a square in their quadrant of the visual field. Unfortunately there was not time to prepare a graphical representation of these results. The results for the testing of the whole system are given in Figure 8.

The graphs in Figure 8 show the saccade directions output by the saliency system for different locations of the stimulus - for example the top left graph is for presentation of the stimulus in quadrant 1 and the bottom right graph is for presentation in quadrant 4. The blue columns show the proportion of saccades to a particular quadrant when a square is presented (not recognized by the top-down saliency system) and the red columns show the proportion of saccades to a particular quadrant when a cross is presented (should be recognized by the top-down saliency system).

If the system was working perfectly, we would expect to see a high value on the blue column corresponding to each quadrant, and a low value for the red column. In the results shown in Figure 8 this is clearly the case in graph 2, and it also occurs to a lesser extent in graphs 3 and 4, where the response to crosses in quadrants 3 and 4 is proportionally much lower than the response to squares. The results in graph 1 are somewhat indecisive and could be due to noisy measurements or poor training of the top-down system.

The results of the very preliminary tests that we ran on on the spike passing rate over the LAN are shown in Figure 9.

With a small packet size the spike rate over a relatively busy 100 MB/s LAN was approximately 500,000 spikes per second at its peak rate. The average rate for larger packets was about 100,000 spikes per second and since the average packet size from the

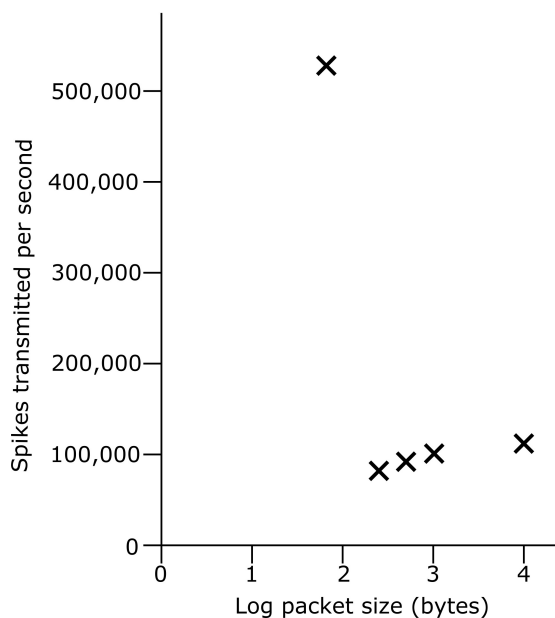


Figure 9: Preliminary performance results for Ethernet AER

silicon retina was a few thousand bytes, this is probably closer to the performance level that could be expected from this type of network. Over a dedicated gigabit network, this performance could be substantially improved, and so one could probably expect to achieve around 500,000 spikes per second under ideal conditions. It would also be possible to replace the absolute time with the relative time between spikes in the protocol, which would increase the spike bandwidth by another 50 percent. Using an event-based architecture this kind of bandwidth of 750,000 spikes per second could be used to comfortably simulate networks of up to 150,000 neurons firing at up to 5 Hz (assuming that a spike is only transmitted for every neuron that fires, not for every synapse that the firing neuron connects to).

## 1.7 Conclusions

This project created a saliency system that combined bottom-up luminosity-driven information with top-down inhibition that prevented the system from saccading to known parts of the visual field. This system included a silicon retina, silicon neurons, a spiking neural simulator and remapping/merging classes and integrated everything together by streaming spike events over a local area network. This Ethernet-based spike messaging framework, combined with the tools available for monitoring, logging and transmitting data, proved to be a very successful way of moving information between hardware and software neural simulations.

## 1.8 References

[1] David Gamez, Richard Newcombe, Owen Holland and Rob Knight, Two Simulation Tools for Biologically Inspired Virtual Robotics, IEEE 5th Chapter Conference on Advances in Cybernetic Systems 2006, forthcoming.

[2] L. Itti, Models of Bottom-Up and Top-Down Visual Attention, California Institute of Technology, Jan 2000.

[3] Filip Ponulak, ReSuMe - New Supervised Learning Method for Spiking Neural Networks, Technical Report, Institute of Control and Information Engineering, Poznan University of Technology, Poland, 2005. Available at <http://d1.cie.put.poznan.pl/fp/> or source:spiking/Ponulak\_TechRep2005.pdf

[4] A. L. Yarbus, Eye Movements and Vision. New York: Plenum Press, 1967. (Translated from Russian by Basil Haigh).