Table of Contents

GPU Pattern Generation for Retina Stimulation Experiments

László Szécsi[1]

Budapest University of Technology and Economics
szecsi@iit.bme.hu

Abstract. The mechanisms of how the cells in the retina process light are varied, complex, and under heavy research. Retina cells are studied by placing retinas surgically removed from test animals on multi-electrode arrays, while projecting carefully designed light patterns on them. This paper details the process of light pattern generation. We study the system requirements arising from the measurement setup and classify the kinds of visual stimuli widely used in research. In order to meet the image processing capabilities required to meet researcher needs, we introduce a GPU framework, and propose a specialized fast filtering method for the required stimuli.

1 Introduction

The retina is more than a transduction system turning light into neuronal spikes that are transmitted to the brain through the optic nerve. It also performs a complex task of image processing. Beside an adaptation to about ten logunits of light, it also provides a set of *multiplex movies* to higher brain centers, enhancing motion, brightness changes, and other information important for survival.

Image-forming vision starts with light absorbtion by rods and cones. These cells pass analog electric signals to bipolar cells, which transmit to retinal ganglion cells, which in turn send the information as a series of spikes to the brain. Two cell types, the horizontal and amacrine cells are involved in establishing connections between neighboring retinal regions and various aspects of image processing [3].

In certain forms of blindness, like retinitis pigmentosa, the light transduction ability of photoreceptor cells can be impaired, or these cells can be missing. Since most of the retinal circuitry is still present, by making the remaining cones, or even some types of bipolar and amacrine cells photosensitive, partial vision can be achieved. As shown experimentally in mice, this goal can be achieved by optogenetic tools, like expressing channel rhodopsin or halorhodopsin in these cell types [1]. By studying the behavior of healthy, degenerate, and artificially restored retinas, researchers hope to achieve better treatments for these visual impairments [4].

Retina cells are studied by placing retinas surgically removed from test animals on multi-electrode arrays, while projecting carefully designed light patterns on them. In this paper we discuss the process of light pattern generation,

but we expressly avoid mentioning any results or research directions in biology or medicine. We only address visual stimuli widely used in research. In order to meet the temporal and spatial filtering capabilities required, we introduce a GPU framework, and propose a specialized fast filtering method for the required stimuli.

2 Experiment setup

Experimental animals (typically mice, but also macaques) are bred for this purpose, exhibiting either healthy, degenerate, or artificially restored retinal function. The animal is killed, and its retina surgically removed. The retina can be kept alive in a solution for approximately eight hours. The experiment has to be conducted in this timeframe, which puts heavy strain on the scientist conducting the experiment. Therefore, it is not possible to redesign experiment parameters or evaluate results during the measurement. Instead, experiments have to be carefully designed and set up in advance, and all possible data for later analysis robustly recorded.

The measurement setup consists of a PC with a GPU, a signal recorder, a multielectrode-array sensor, and a projector system. The retina sample (approx. 2 mm $\times$2 mm) is placed on the sensor. The program running on the PC sends control and synchronization signals to the signal recorder while driving the projector system through the GPU to display stimulus patterns. The patterns are projected onto the retina. The values measured by the sensor are recorded by the signal recorder. For analysis the measurement has to be linked to the stimuli, so they have to be synchronized.

The total extent of the projected image is called the *full field*. The field is measured in pixels in *screen space* and measured in micrometers (um) in *retina space*. The *pattern* is a rectangle within the full field. Outside of the pattern the field is black. The mapping of image space to retina space, performed by the optical system, must be defined by specifying the pattern size both in image space pixels and retina space micrometers. All measurements of length, specified in micrometers, are converted by the program to screen space using this established relation between the two spaces.

3 Stimuli

There are two basic types of stimuli: shapes and grids. Shapes can be changing their positions, colors and intensities. Grids are made up of uniform quadrilaterals of different colors and intensities (Figure 1). All stimuli can be subject to temporal and spatial filtering. Stimuli must be projected at a full screen resolution with a strictly constant frame rate, defined by the refresh rate of the projector, which is nominally 60 Hz (59.94 Hz is the standard refresh rate for display devices). Spatial filtering can easily require convolving with a 512×512 matrix. This is not feasible with straightforward convolution.

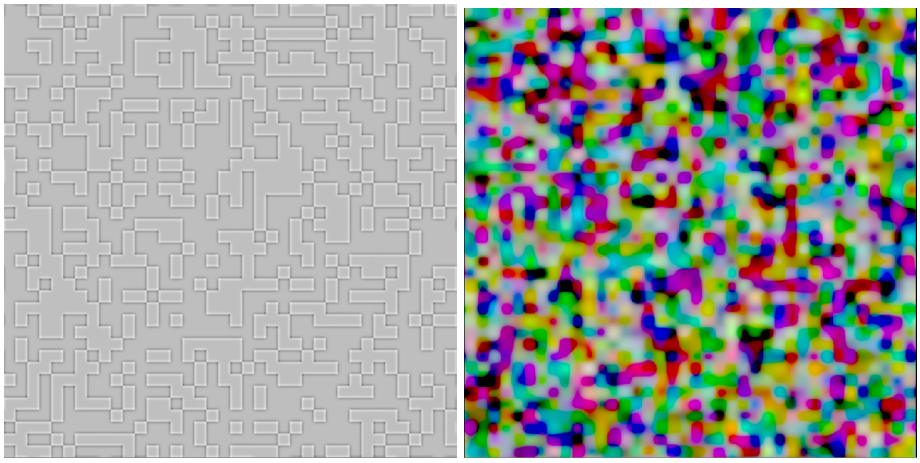

Fig. 1. Example grid-based stimuli.

4 Computation pipeline

The pipeline is primarily designed to display random grids spatially and temporally filtered. Other stimuli can be achieved by modifying elements of this pipeline. The grid dimensions are given as $(n_\mathrm{x}, n_\mathrm{y})$ for every experiment.

All passes in the computation pipeline write results to a single texture (2D array). The GPU renders a full-viewport quadrilateral in all cases. Therefore, the vertex program is fixed as the simple passthrough program. The fragment programs computing the elements of the output arrays are, however, customizable.

The computation pipeline consists of the following passes:

1. Pseudo-random number generation
2. Stimulus computation
3. Temporal filtering
4. Spatial filtering

The following data resources are used:

Random sequence buffers $\mathbf{R}_0 \dots \mathbf{R}_4$ hold the state of the pseudo-random sequences. There is one sequence for every grid cell, so buffer dimensions are $(n_\mathrm{x}, n_\mathrm{y})$.

Stimulus queue $\mathbf{G}_0 \dots \mathbf{G}_T$ a GPU-addressable array of 2D textures of unfiltered stimulus grids of dimensions $(n_\mathrm{x}, n_\mathrm{y})$. Texture $\mathbf{G}_0$ always holds the current unfiltered stimulus, the rest hold the unfiltered stimulus from previous frames.

Stimulus grid $\tilde{\mathbf{G}}$ is a 2D texture of stimulus values, with dimensions $(n_\mathrm{x}, n_\mathrm{y})$. It holds current stimulus values after temporal filtering.

Temporal weights array W is simple array that contains the temporal filtering weights. This is a read-only resource, it does not change during rendering.

Spatial SAT F_{SAT} is a 2D texture of floating point values. This is a summed-area-table of the spatial filter kernel. This is a read-only resource, it does not change during rendering.

The final image $\tilde{S}$ will be written to the *frame buffer*. The frame buffer serves as the output (render target) of the spatial filtering pass.

5 Pseudo-random number generation

This pass computes new pseudo-random numbers, based on the random numbers of the previous four frames. This is required to ensure a long enough cycle length. Randoms are computed using the Linear Feedback Shift Register (LFSR) algorithm:

$$\mathbf{R}_4 \leftarrow \mathbf{R}_3, \mathbf{R}_3 \leftarrow \mathbf{R}_2, \mathbf{R}_2 \leftarrow \mathbf{R}_1, \mathbf{R}_1 \leftarrow \mathbf{R}_0,$$
$$\mathbf{R}_0 \leftarrow \mathrm{LFSR}(\mathbf{R}_1, \mathbf{R}_2, \mathbf{R}_3, \mathbf{R}_4).$$

While this pseudorandom sequence is deemed appropriate for the purposes of the experiments, we should note that there are pseudo-random number generators (PRNGs) with better properties. The pseudo-random sequences of different texels (and within texels, the different channels) should be initialized using seeds generated by a different (possible more complex) pseudo-random number generator (PRNG). In the current implementation, it is accomplished using the C rand() function.

Knowing the seeds for all texels, it is possible to re-create the same sequence of random numbers in any experiment result analysis environment (e.g. Matlab).

Randoms are always generated. If the experiment does not require randoms, the randoms are still generated, but are ignored by later passes.

6 Stimulus generation

This pass maps the random numbers to stimulus intensities or colors. Depending on the desired stimulus pattern, mappings may be different, from producing a binary, dark or light value to mapping RGB channels linearly. The random numbers are ignored for deterministic stimuli. The output can be time-dependent, e.g. modulated by sine or square wave signals.

The stimulus grid generation process may use the randoms and time t

$$\mathbf{G}_T \leftarrow \mathbf{G}_{T-1}, \ldots, \mathbf{G}_1 \leftarrow \mathbf{G}_0$$
$$\mathbf{G}_0 \leftarrow \mathrm{StimGen}(\mathbf{R}, \mathrm{t}).$$

The output of this pass is a texture of size $(n_{\mathbf{x}}, n_{\mathbf{y}})$, and assumed to be relatively small. Thus, the generated stimulus pattern must be low-res, composed of large rectangular texels. Any details beyond that (like circular spots or moving bars) can only be added later, in the spatial filtering pass.

The stimulus generator pass is always run. Later shaders may choose to ignore the results (this happens in case of shape stimuli).

7 Temporal filtering

This pass computes a weighted average of unfiltered stimulus grids in the stimulus queue. Weights are obtained from the temporal weights array. Temporal filtering is a weighting of grids $\mathbf{G}_i$ with array elements $\mathbf{W}_i$:

$$\tilde{\mathbf{G}} = (\mathbf{G}_0, \ldots, \mathbf{G}_T) \cdot \mathbf{W}.$$

8 Spatial filtering

This pass computes the image displayed in the current frame (Figure 2). It is possible that it does not actually perform spatial filtering, if it is not required. In any case, this shader is responsible for creating the full-resolution version of the stimulus.

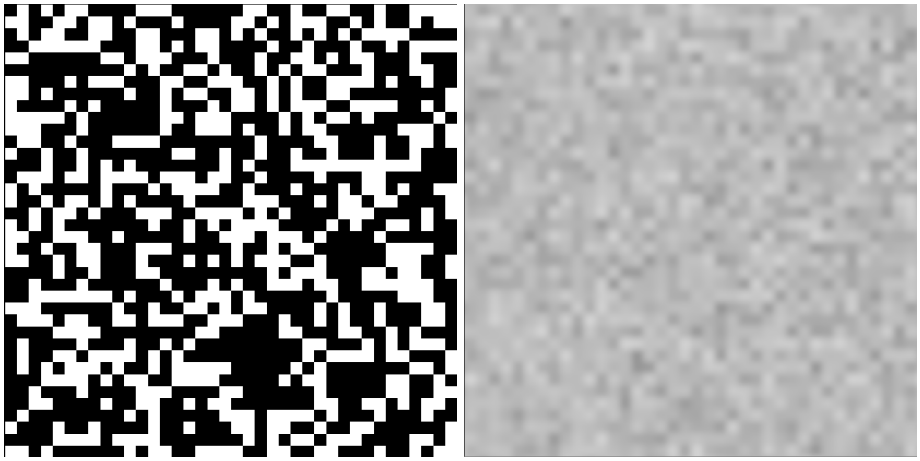

Fig. 2. Random grid pattern without spatial filtering and with spatial filtering.

Convolution matrices are large, but they have to be convolved with a low-resolution grid image. With pattern size $(p_\mathrm{x}, p_\mathrm{y})$. Cell size $(c_\mathrm{x}, c_\mathrm{y})$ is

$$(c_\mathrm{x}, c_\mathrm{y}) = (p_\mathrm{x}/n_\mathrm{x}, p_\mathrm{y}/n_\mathrm{y}).$$

The continuous grid image $S(x, y)$ is obtained from the discrete grid as

$$S(x, y) = \tilde{\mathbf{G}}\,[\lfloor x/c_\mathrm{x} \rfloor, \lfloor y/c_\mathrm{y} \rfloor]$$

Spatial filtering is a continuous convolution

$$\tilde{S} = S * F,$$

where F is a filter function, with a support within the origin-centered $r_\mathrm{x} \times r_\mathrm{y}$ quadrilateral. We can write the convolution integral as

$$\tilde{S}(x,y) = \int_{-r_\mathrm{x}/2}^{r_\mathrm{x}/2} \int_{-r_\mathrm{y}/2}^{r_\mathrm{y}/2} S(x-u, y-v) \cdot F(u,v) \mathrm{d}v\mathrm{d}u.$$

As S is piecewise constant, this can be written as a sum. Let us choose *neighborhood distance* $(m_\mathrm{x}, m_\mathrm{y})$ so that

$$(2m_\mathrm{x}+1) \cdot c_\mathrm{x} < r_\mathrm{x} \text{ and } (2m_\mathrm{y}+1) \cdot c_\mathrm{y} < r_\mathrm{y},$$

meaning a neighborhood of $(2m_\mathrm{x}+1) \times (2m_\mathrm{y}+1)$ cells covers the extents of the filter function. Then the integral can be written for all neighborhood cells separately, and summed:

$$S(x,y) = \sum_{i=-m_\mathrm{x}}^{m_\mathrm{x}} \sum_{j=-m_\mathrm{y}}^{m_\mathrm{y}} \tilde{G}\left[\lfloor x/c_\mathrm{x} \rfloor - i, \lfloor y/c_\mathrm{y} \rfloor - j)\right] \cdot \int_{(-\langle x/c_\mathrm{x}\rangle+i)c_\mathrm{x}}^{(-\langle x/c_\mathrm{x}\rangle+i+1)c_\mathrm{x}} \int_{(-\langle y/c_\mathrm{y}\rangle+j)c_\mathrm{y}}^{(-\langle y/c_\mathrm{y}\rangle+j+1)} F(u,v) \mathrm{d}v\mathrm{d}u,$$

where $\lfloor\rfloor$ is the integer part and $\langle\rangle$ is the fractional part.

The integral of the filter function can be rapidly evaluated with the use of a pre-calculated Summed Area Table (SAT)[2]. A SAT stores values of the cumulative filter function

$$F_\mathrm{SAT}(x,y) = \int_{-\infty}^{x} \int_{-\infty}^{y} F(u,v) \mathrm{d}v\mathrm{d}u,$$

with which any integral over a rectangular area can be written as:

$$\int_{x_0}^{x_1} \int_{y_0}^{y_1} F(u,v) \mathrm{d}v\mathrm{d}u = F_\mathrm{SAT}(x_1,y_1) - F_\mathrm{SAT}(x_0,y_1) - F_\mathrm{SAT}(x_1,y_0) + F_\mathrm{SAT}(x_0,y_0).$$

With the SAT pre-computed into a texture, the evaluation of the integral becomes possible with just four texture lookups per neighborhood cell.

9 Calibration for dynamic range

The dynamic range of the projector is limited. While it is theoretically possible to calculate parameters for experiments so that the stimulus values remain in this range, or remain in this range with high probability, this computation can be substituted by a measurement. Thus, the program offers parameters to specify the mapping of stimulus values to the projector's dynamic range, and a simulation mode where the stimulus values are displayed on a histogram, visualizing outliers.

10 Calibration for device gamma

The projected intensities are subject to the characteristics of the projector device. For the precise evaluation of the measurements, the gamma distortion of the device must be compensated. The characteristics are given by the manufacturer, but can also be measured.

11 Conclusions

Our system provides a means for designing and performing measurements that meet researcher needs. The GPU implementation and the SAT-based convolution make it possible to maintain the required frame rate. In the future, it may be necessary to use frequency-domain processing, if filtering needs to be performed on high-resolution stimuli.

Acknowledgements

This work has been supported by OTKA PD-104710 and the MTA János Bolyai scholarship.

References

1. Shannon E Boye, Sanford L Boye, Alfred S Lewin, and William W Hauswirth. A comprehensive review of retinal gene therapy. *Molecular Therapy*, 21(3):509–519, 2013.
2. Franklin C Crow. Summed-area tables for texture mapping. *ACM SIGGRAPH computer graphics*, 18(3):207–212, 1984.
3. Robert W Rodieck. The vertebrate retina: Principles of structure and function. 1973.
4. Bernhard A Sabel, Petra Henrich-Noack, Anton Fedorov, and Carolin Gall. Vision restoration after brain and retina damage: the residual vision activation theory. *Prog Brain Res*, 192:199–262, 2011.