Manual

Structure of the repository

This repository contains data from 13 sessions of multi-electrode array recordings (252 channels) from mouse retinas that were stimulated with patterns of light. The data accompany the manuscript by Karamanlis and Gollisch: "Nonlinear spatial integration underlies the diversity of retinal ganglion cell responses to natural images". Each .rar file (named by recording date and an identifier for right or left eye, YYYYMMDD_XX.rar) comprises one recording session including following:

- *list_of_good_cells.txt* (text file containing a table of identifiers for good units of spike-sorted data)
- *frametimes* (folder with Matlab files containing time stamps of specific frames for each stimulus)
- *spiketimes* (folder with text files containing spike times of all sorted units, also containing bad units)
- *stimuli* (folder with files containing stimulus parameters)

Details of the applied stimuli and their reconstruction can be found in Section 2.

1.1 Spike times

The spike responses of retinal ganglion cells to a visual stimulus were extracted with a custom-made spike sorting algorithm from the multielectrode-array recorded data. The folder *spiketimes* contains spike times of each spike-sorted unit in seconds. Good units can be found in *list_of_good_cells.txt* where the first column is the first channel of a 4-channel grouping and the second is the number of the cluster. The third column indicates the goodness of each cluster. The spikes of each unit in response to a certain stimulus can then be found in the file <stimulus number>_SP_C<channel number><2-digit cluster number>.txt

1.2 Frame times

In order to align recorded spike times and the timing of the stimulus, the time of appearance of specific stimulus frames for each stimulus is saved in the variable *ftimes* of the file

<stimulus number>_<stimulus name and parameters>_frametimings.mat

Frame times were saved in milliseconds.

1.3 Stimulus parameters

Stimulus-specific parameters are saved in the *stimpara* structure inside the file

<stimulus number>_<stimulus name and parameters>.mat

2 Stimulus details and reconstruction

Seven types of visual stimuli were used to characterize and analyze the responses of retinal ganglion cells. The field *stimulus* of the parameters structure contains the identifier for each type.

1) Receptive fields were determined from responses to a spatio-temporal white-noise stimulus (stimulus id: *checkerflicker* or *frozennoise*).
2) Responses to flashed natural images where obtained under the (stimulus id: *imagesequence* or *imagesequenceblur*).
3) For determining the spatial sensitivity of ganglion cells, we used contrast-reversing gratings of different grating widths (stimulus id: *reversinggratingswithvaryingspatialperiod*).

4) To determine how spatially separated visual signals are integrated in ganglion cells, we presented a series of checkerboard flashes to the retina (stimulus id: *subunitflash*).
5) To focus on spatial integration only in the receptive field center, we used a locally sparse version of checkerboard flashes (stimulus id: *locallysparsesubunitflash*).
6) For identifying image-recurrence-sensitive cells, we used a series of gratings randomly shifting between different spatial phases (stimulus id: *saccadegrating*).
7) For identifying direction- and orientation-selective cells, we used drifting gratings of different spatial frequencies/speeds (stimulus id: *directiongratingsequence*).

All stimuli were presented to the retina through an 8bit 800x600 OLED monitor (eMagin), with a refresh rate of 60 Hz. Each pixel of the OLED monitor had a size of 7.5 μm on the retina.

2.1 Checkerflicker or FrozenNoise

A spatio-temporal white-noise stimulus of black and white squares (100% contrast) was used to estimate a cell's receptive field. Each square was randomly assigned to black (0) and white (1) with a probability of 50% each, and had a side of `stimpara.stixelwidth` pixels. Spatial patterns were updated with a frequency of `60/stimpara.Nblinks` Hz. Frame times were always recorded with a frequency of 30 Hz. To cover screen updates for 60 Hz refresh rates, we linearly interpolated between the 30 Hz frame timings.

Check the repository https://github.com/gollischlab/RecreateWhiteNoiseStimuliWithMatlab for reconstructing the frame sequence. Example usage of repository function:

```
stimulus = recreateBinaryWhiteNoiseStimulus(stimpara.Nx, stimpara.Ny, Nframes, stimpara.seed)
```

FrozenNoise

In the case of a FrozenNoise stimulus, a fixed white noise sequence was repeated periodically between series of "running" white noise sequences. In particular, after `stimpara.RunningFrames`, a number of `stimpara.FrozenFrames` was presented with a fixed seed `stimpara.secondseed`. After each run of the fixed sequence, "running" presentations were resumed with the last seed that was generated before the fixed sequence presentation.

2.2 ImageSequence or ImageSequenceBlur

A series of flashed images was presented to the retina, interleaved with a gray screen. Every image trial lasted for `stimpara.trialduration` frames and the flash was presented between frames `stimpara.flashstart` and `stimpara.flashstop` of that duration. Frame times mark the beginning of each image trial (e.g. for a trial duration of 60, there is a pulse every second). Images were flashed in the central `stimpara.Nx x stimpara.Ny` pixels of the screen, and their bottom left point was in the same quadrant as the bottom left point of the screen (corresponding to the (0,0) coordinate for whine noise stimuli). Images were saved as 8bit .raw files, and where based on images from three different databases.

The actual list of images presented can be obtained by reading either of the text files (mixed200.txt or mixed300.txt) in the folder code_for_stim_reconstruction. The folder of `stimpara.path` determines which of the lists to use. For each row of those lists, there is a reference to the database that the images were obtained from (Berkeley [1], vanHateren [2], or McGill [3]), and a unique identifier for each image in that list. Images were pre-processed as indicated in our paper (see *Methods* section, *Natural image response predictions with a linear-nonlinear model*), and example code can be found code_for_stim_reconstruction/prepareNaturalImages.m.

We collected `stimpara.nrepeats` trials for each image, by consecutively presenting `stimpara.nrepeats` different pseudo-randomly permuted sequences of all images. In the beginning of each permutation sequence, a blank image was presented.

Example code usage:

```
prs = findImageOrder(Nimages, Npresentations, stimpara.nrepeats, stimpara.seed);
```

Here, `Npresentations` can be read by the number of frame times, and `Nimages` corresponds to the number of images in the corresponding list. The numbers in the `prs` sequence correspond to the image ids from the text files we provide. The 0^{th} id corresponds to the blank screen at the beginning of each permutation sequence.

Blurring

For the experiments with blurred natural images, we used always used the same 40 images, which were blurred with circular Gaussians of different sizes. In different experiments, we used different blurring scales, specified in the `stimpara.scales`. The scale parameter corresponds to 2σ of the circular Gaussian. In the manuscript, we reported the 4σ value (diameter of 2σ contour).

The order of the 40 images can be found in code_for_stim_reconstruction/blur40.txt. To generate the order of presentation, the list has to be extended to include the blurred images. Specifically, for N scales of blurring, the order is [blur40, blur40_scale, blur40_scale2, …, blur40_scaleN], and there are 40*N total images. Similarly to above, a blank image was presented in the beginning of each permutation sequence.

2.3 ReversingGratingsWithVaryingSpatialPeriod

We measured the spatial integration properties of ganglion cells using square wave grating stimuli (100% contrast) with grating widths of `stimpara.stripewidths`. For each grating width, a different number of equidistant spatial phases were applied. The number of phases for each width was different, and can be found in `stimpara.Nphases`. The polarity of each grating was reversed every `stimpara.Nframes` frames for a total of `stimpara.Nreversals` reversals. In between the sequence of reversals of different gratings, a grey screen was presented for `stimpara.preframes` frames. The whole sequence was repeated for `stimpara.Nrepeats` trials.

Stimulus presentation was sequential, e.g. [gray screen, reversals of 1^{st} phase of 1^{st} stripe width, …, gray screen, reversals of n^{th} phase of 1^{st} stripe width, …, gray screen, reversals of 1^{st} phase of 2^{nd} stripe width, …, gray screen, reversals of n^{th} phase of k^{th} stripe width]. Frame times mark the beginning of the gray screen and each contrast reversal.

2.4 SubunitFlash

A series of flashed checkerboards was presented to the retina, interleaved with a gray screen. Every checkerboard trial lasted for `stimpara.stimduration` frames and the flash was presented between frames `stimpara.flashstart` and `stimpara.flashstop` of that duration. Frame times mark the beginning of each image trial (e.g. for a trial duration of 60, there is a pulse every second). The size of the checkerboard tiles (in pixels) was `stimpara.stixelwidth` x `stimpara.stixelheight`.

The brightness value of each tile checkerboard tile (A and B), varied between flashes, and it was sampled from a polar stimulus space. The following code provides the list of tile A and B brightness combinations (in `cta` and `ctb` variables), which were presented in the order given by `prs`.

```
[cta, ctb, ncombis, prs] = findNewOrder(stimpara, Npresentations, stimpara.nrepeats, stimpara.seed);
```

Brightness values are given in Weber contrast relative to the background. Here, `Npresentations` can be read by the number of frame times. Similar to the ImageSequence stimuli, the numbers in the `prs` sequence correspond to the stimulus space ids. The 0th id corresponds to the blank screen at the beginning of each permutation sequence.

2.5 LocallySparseSubunitFlash

A series of multiple locally sparse checkerboards was presented to the retina for `stimpara.stimduration` frames each. For every presentation, stimulus locations were randomly selected from a square grid (`stimpara.Nx` x `stimpara.Ny` vertices) so that, for every chosen vertex, vertices in a square area around it were avoided (number of avoided vertices is `stimpara.gapwidth`). A 2-by-2-tile checkerboard was presented with a tileA/tileB brightness combination, randomly chosen from a stimulus space of brightness pair (similar to 2.4). The checkerboard tiles had a side of `stimpara.stixelwidth`. The rest of the screen remained at background gray illumination. The simultaneously presented contrast combinations at different locations were chosen independently of each other. Frame times mark the beginning of each presentation.

We provide example code that can regenerate the sequence of all contrast combinations and their corresponding locations for each frame:

```
[smat, cta, ctb, screenMat, ~, ~] = ...
     findLocallySparseSubunitFlashOrder(Nframes, stimpara);
```

Here, `smat` is an `Nframes` x `stimpara.Ny` x `stimpara.Nx` sparse array that contains stimulus ids in that locations were occupied in each frame. The stimulus ids refer to the stimulus space of brightness values for tiles A and B in Weber contrast, stored in `cta` and `ctb` (similar to 2.4). A visualization of the different stimulus frames is provided in the `screenMat` array.

2.6 SaccadeGrating

To identify image-recurrence-sensitive cells, we presented a pseudorandom sequence of four different phases of a square wave grating (with a grating width of `stimpara.barwidth` pixels). Each phase was presented for `stimpara.fixation` frames; between the presentations of successive phases, a transition was shown for `stimpara.saccade` frames. The transition was either a gray screen or a saccade. Frame times mark the beginning of each transition and each fixation (presentation of the grating).

We provide example code that generates the list of all possible fixation and transitions:

```
[trcombis, stimseq, ~, ~] = generateSaccadeSeq(stimpara, Nframes);
```

Here, `Nframes` corresponds to the number of total fixation positions, and can be extracted by the frame timings. The provided function outputs a list of all possible combinations of starting and target phases, and transition types (`trcombis`, 32 combinations). The stimulus sequence can be recovered from the variable `stimseq`, which marks the four different grating phases with 1-4 and the two different transition types with 5 or 6 (5 is a gray transition and 6 a saccadic one).

2.7 DirectionGratingSequence

We used a sequence of drifting (square or sinusoidal) gratings to identify direction- and orientation-selective cells. Each stimulus in the sequence had distinct parameters in terms of spatial frequency (`stimpara.gratingwidthwhite` contains the different grating widths in μm on the retina), speed (`stimpara.period` contains the temporal period of each grating in frames), number of equidistant directions (`stimpara.nangles`, usually 8), repetitions (`stimpara.cycles`), presentation duration (`stimpara.duration` in frames), gray screen duration between presentations of gratings of different directions (`stimpara. preframeAngles` in frames) and type of grating (`stimpara. squareWave` is 1 when the grating was square, and 0 when it was sinusoidal). A gray screen was presented sometimes between different stimuli, lasting for `stimpara. preframeStimuli` frames. The first direction shown was always 180 degrees relative to the screen center, and the rest continued counterclockwise.

Frame times mark the beginning of all temporal periods during the presentation of a drifting grating, excluding the first period following the gray screen. Stimuli were presented sequentially, and the stimulus sequence can be reconstructed based on the following list for K different stimuli, M directions and N trials:

[stim1_direction1_trial1, stim1_direction2_trial1, …,
stim1_directionM_trial1, stim1_direction1_trial2, …,
stim1_directionM_trial2, …, stim1_directionM_trialN, …, stim2_direction1_trial1, …,
stim2_directionM_trialN, …, stimK_directionM_trialN]

References

1. The Berkeley Segmentation Dataset and Benchmark, https://www2.eecs.berkeley.edu/Research/Projects/CS/vision/bsds/
2. van Hateren's Natural Image Dataset, http://bethgelab.org/datasets/vanhateren/
3. McGill Calibrated Colour Image Database, http://tabby.vision.mcgill.ca/