Manual

Structure of the repository

This repository contains data from 18 multi-electrode array recordings from mouse and marmoset retinas that were stimulated with different patterns of light. The data accompany the manuscript by Karamanlis et al: "Natural stimuli drive concerted nonlinear responses in populations of retinal ganglion cells". Each .rar file comprises one recording session and is named by recording date and an identifier for right or left eye, YYYYMMDD_MEATYPE_SPECIES_EYE_RETINALREGION.rar. For mouse recordings, the retinal region is either "half_dorsal" or "half_ventral", indicating the corresponding half of the retina that was used. For marmoset recordings we used internal specifications for indicating retinal region (e.g., n1 comes from the nasal retina, s3 from the superior). Each .rar file contains the following Matlab files:

- *expdata.mat:* contains general experiment information.
- *STIMULUS_data.mat:* contains stimulus-specific data from the corresponding session (can be more than one); *STIMULUS* here is a placeholder for the name of the particular stimulus.

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

1 General experiment information ("expdata")

The file contains the following variables:

`animal`	'mouse' or 'marmoset'.
`array`	Structure containing a type specifier (array.type) of the recording device, number of electrodes (array.nelectrodes), their diameter (array.diamelectrode), and the inter-electrode distance (array.distelectrodes) in meters.
`units`	An Nunits x 4 array containing information about each sorted unit (with Nunits being the number of sorted units, i.e., ganglion cells). Column 1: Kilosort ID, Column 2: number of the electrode (reference electrode) on the array with the highest amplitude for the unit, Column 3: unit ID of the units with the same reference electrode, Column 4: spike-sorting quality noted during manual curation (1 down to 3).
`fs`	Recording sampling rate in Hz.
`projector`	Structure containing information about the light projection system, including the screen refresh rate, size in pixels, and the pixel size on the retina in meters.
`typelabels`	Labels with names of the four ganglion cell types identified in the recordings.
`cellclus_id`	Ganglion cell type assignment, containing labels of 0 to 4, indicating the types defined in "typelabels" (0 stands for unclassified).

2 Raw spikes, stimulus details and reconstruction ("STIMULUS_data")

All stimulus-data files contain similar types of structures that hold raw spike times and stimulus frame times.

The spike responses of retinal ganglion cells to visual stimuli were extracted with a Kilosort-based algorithm from the multielectrode-array recorded data. The array `spiketimes` contains spike times of each spike-sorted unit. The first column contains the sorted timestamps (in samples, corresponding to the recording sampling rate `fs` specified in "expdata.mat") of all detected spikes, and the second column their assignment to a unit in the recording (matching the row number of the array units in "expdata.mat").

To align recorded spike times with stimulus presentation, we also recorded square pulses that indicate when a new stimulus appears on (or disappears from) the screen. These pulses last as long as a single frame of the monitor (e.g., 1/85 s for an 85-Hz refresh rate). The pulse onsets and offsets were saved (in samples) in two different variables: `fonsets` and `foffsets`. The frequency and significance of the pulses are stimulus-specific, with information given below in the explanations of the individual stimuli.

Stimulus-specific parameters were saved in the `stimPara` structure. These parameters are necessary for the reconstruction of the presented stimulus.

Five types of visual stimuli were used to characterize and analyze the responses of retinal ganglion cells. The *STIMULUS* part of the file name contains the identifier for each type.

1) Receptive fields were determined from responses to a spatio-temporal white-noise stimulus (stimulus id: *frozencheckerflicker*).
2) Responses to natural movies made from combining natural images with gaze data (stimulus id: *fixationmovie*).
3) Responses to flashed natural images (stimulus id: *imagesequence*).
4) For fitting subunit grid models to data, we presented a flashed sequence of gratings with varying spatial frequencies, orientations, and phases (stimulus id: *gratingflashes*).
5) For fitting subunit grid models with temporal dynamics, we presented gratings in a rapid succession (stimulus id: *gratingflicker*).

2.1 White noise ("frozencheckerflicker_data")

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) or white (1) with a probability of 50% each, and had a side length of `stimPara.stixelwidth` pixels. Spatial patterns were updated with a frequency of `fps/stimPara.Nblinks`, where `fps` is the monitor refresh rate. Frame pulses were always delivered with a frequency of `fps/2`. To cover screen updates for `stimPara.Nblinks=1` refresh rates, we therefore used both pulse onsets and offsets to recreate the frame timings when a new white-noise image appeared. The `spikesbin` array contains spike count data already binned at the stimulus update rate to facilitate analyses.

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 random-number-generator 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.

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)
```

The arrays `spaceVecX` and `spaceVecY` give the spatial coordinates (in monitor pixels) of the center of each stimulus tile.

2.2 Natural movies ("fixationmovie_data")

Movie frames were updated with the refresh rate of the monitor (`fps`). Frame pulses were always delivered with a frequency of `fps/2`. Thus, both pulse onsets and offsets are needed to recreate the frame timings.

A fixed movie sequence was repeated periodically between series of "running" movie sequences. **After** `stimPara.FrozenFixations`, a number of `stimPara.RunningFixations` was presented. After each run of the running sequence, the fixed movie sequence was repeated.

The `spikesbin` array contains spike count data binned already at the stimulus update rate. Here, these spike counts are reorganized into `frozenbin` and `runningbin` arrays, with dimensions of Ncells x Nframes x Ntrials, where Nframes is either the number of fixed or running frames.

The images used for the fixed sequence are contained in `frozenImages`, and for the running in `runningImages`. (Note that the coordinates of images are in matrix convention of rows x columns and that the rows correspond to the y coordinate on the screen, top to bottom, and the columns to the x coordinate, left to right.) In each frame, the images were shifted based on gaze data. The images presented at each frame and the corresponding shifts are saved in the `frozenfixations` and `runningfixations` arrays.

`frozenfixations` is a 3 x Nframes array, where Nframes is the number of frames in the “frozen” (repeated) movie sequence. `runningfixations` is a 3 x Nframes x Ntrials array, where Nframes is the number of frames in the “running” (non-repeated) sequence and Ntrials is the number of those presented sequences. The first dimension of both arrays contains three numbers: the image ID (1 in `frozenfixations` corresponds to the first image in `frozenImages)`, and the x- and y-position of the image that is currently on the center of the screen. These positions specify a pixel of the image (starting the count with 1,1 as the pixel in the top left corner), that is then displayed at the center of the screen, e.g., at pixel (400, 300) for a screen of 800 x- and 600 y-pixels.

To reconstruct the actual frames that were presented to the retina (`blockstimulus)`, use the following function:

```
blockstimulus = returnFixMovie(screensize, imageEnsemble, listfixations)
```

where `screensize` is the monitor size (always [600, 800] for our experiments, with number specifying the y- and x-dimensions of the screen correspondingly).

2.3 Natural images (“imagesequence_data”)

All raw data (spike times, pulse onsets and offsets, stimulus parameters) are contained in the structure `rawdata`. 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 image flash was presented between frames `stimPara.flashstart` and `stimPara.flashstop` of that duration. `stimPara.flashstart` is the first frame of the flash and `stimPara.flashstop` is the first frame of the gray screen afterwards. Frame onsets mark the beginning of each image trial (e.g., for a trial duration of 85 and a refresh rate of 85 Hz, there is a pulse every second). The images we presented are contained in the `imageEnsemble` array. 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. The exact image-screen correspondence can be traced by using the `spaceVecX` and `spaceVecY` arrays, that give the screen pixels coordinates of each image pixel. To display an example of a presented image in MATLAB, call `imagesc(spaceVecX, spaceVecY, imageEnsemble(:,:,1))`.

We collected multiple trials for each image, by consecutively presenting different pseudo-randomly permuted sequences of all images. In the beginning of each permutation sequence, a gray screen was presented. To extract the image presentation sequence, run the Matlab script “findImage Order” in “stimulus_reconstruction”:

```
presentOrder = findImageOrder(Nimages, Npresentations, stimPara.nrepeats,
stimPara.seed)
```

Here, `Npresentations` can be read by the number of frame onsets, and `Nimages` corresponds to the number of images in `imageEnsemble`. The numbers in the `presentOrder` sequence correspond to the image ids from the third dimension of `imageEnsemble`. `presentOrder` entries with 0 corresponds to blank screens at the beginning of each permutation sequence.

We already organized the data in a way that `presentOrder` contains the image ID presented in each trial (without the gray screen), and `trialCounts` contains the measured spike count during each image presentation for each unit.

2.4 Flashed gratings (“gratingflashes_data”)

The organization of flashed grating data is like the one of the natural images. All raw data (spike times, pulse onsets and offsets, stimulus parameters) are contained in the `rawdata` structure. For some datasets, the `rawdata` structure contains data from two sessions of flashed gratings within the same experiment.

A series of flashed gratings was presented to the retina. Every grating trial lasted for `stimPara.stimduration` frames, and the flash was presented between frames `stimPara.flashstart` and `stimPara.flashstop` of that duration. For this stimulus, frame pulses were recorded for both the beginning and the end of each grating presentation. Thus, the number of values of pulse onsets is double the number of grating presentations.

Each grating is specified by three parameters: spatial frequency (in px^{-1}), orientation (in radians), and spatial phase (in radians). These three parameters for each different grating are contained in the array `stiminfo`. To reconstruct images of all presented gratings (`stimmat`) and obtain `stiminfo` for any parameter combination, use:

```
[stimmat, stiminfo] = getGratingFlashStimulus(stimPara, spX, spY, bwflag)
```

where `stimPara` can be found in `rawdata`, `spX` and `spY` are also given and mark the values of spatial coordinates for which the grating is calculated, and `bwflag` marks whether the grating is square-wave instead of sinusoidal (set to `False` for all recordings). In some cases, gratings were presented over a smaller part of the screen. Grating coordinates can be turned into image pixels using the `spaceVecX` and `spaceVecY` vectors (similar to the images), given by following equations, where `ypix` is the y-dimension of the screen:

```
spaceVecX = spX + rawdata.stimPara.rmargin + 0.5
```

```
spaceVecY = ypix – (spY + rawdata.stimPara.bmargin + 0.5)
```

To extract the grating presentation sequence (similar to natural image presentation) use the following function:

```
presentOrder = orderGratingFlashes(Ngratings, Npresentations, stim-
Para.nrepeats, stimPara.seed)
```

We already organized the data in a way that `presentOrder` contains the grating ID presented in each trial (which corresponds to a row number of `stiminfo`), and `trialCounts` contains the measured spike count during each grating presentation for each unit.

2.5 Flickering gratings (“gratingflicker_data”)

We also presented sinusoidal gratings in a rapid succession. Several gratings (2000-3000) were pre-generated and then presented to the retina in a pseudorandom order. Gratings presented can again be obtained using the `getGratingFlashStimulus` function.

All raw data (spike times, pulse onsets and offsets, stimulus parameters) are contained in the `rawdata` structure. For some datasets, the `rawdata` structure contains data from two sessions with flashed gratings within the same experiment. Stimulus frames were updated with the refresh rate of the monitor (`fps`). Because frame pulses were always recorded with a frequency of `fps/2`, both pulse onsets and offsets are needed to recreate the frame timings. The `spikesbin` array contains spike count data binned already at the stimulus update rate.

A fixed grating sequence was repeated periodically between series of “running” grating sequences. After a number of `stimPara.RunningFrames`, a fixed sequence of `stimPara.FrozenFrames` was presented. After each run of the fixed sequence, the running presentations were resumed. A single run of running and frozen frames is a stimulus trial. To obtain the order of gratings presented, use:

```
currorder = orderGratingFlicker(Ngratings, seed, Npresent)
```

where `seed` is either `stimPara.secondseed` for the frozen sequence, or `stimPara.seed` for the running sequence, and `Npresent` is either `stimPara.FrozenFrames` for the frozen sequence, or `stimPara.RunningFrames * Ntrials` for the running sequence across all trials of stimulus presentation (can be calculated from the numbers of presented frames).

We already organized running data in a way that `stimorder` contains the grating ID presented in each trial. Because our temporal subunit grid models contain filters that are Nt frames long (e.g., Nt = 43), we further reorganized `stimorder` into `orderfit`, which is an Nt x Nstimuli array. There are Ntrials*(Nrunningframes-Nt+1) stimuli, which essentially describe a short sequence of gratings up to a particular time. For each stimulus, we provide the corresponding spike count (for all cells) in the `spikesfit` array. We expressed temporal filters in our models in a ten-vector basis, which we also include in the array `ktbas`.