Data Collection: The data presented here represent the responses of V1 populations of neurons to natural image sequences. Visual stimulation was performed as described in below. A description of surgical procedures, anesthesia, and the insertion of the arrays can be found in this paper. All recordings were performed on Old-world monkeys (*Macaca fascicularis*) in primary visual cortex (area V1). Receptive fields were between 2 and 6 degrees in eccentricity.

Visual stimulus: Natural image sequences were generated by digitally sampling commercially available videotapes in VHS/NTSC format. A Silicon Graphics R10000 Solid Impact was used to sample frames at a spatial resolution of 320 × 240 pixels and at a temporal rate of 30 Hz. Thirty different segments of approximately 30-s duration were sampled from four different movies (*Sleeper*, *Benji*, *Goldfinger* and *Sheakespere in Love*), making a total of 20 minutes of video.

The movies were compressed using Silicon Graphics' MVC2 compression scheme (proprietary) and stored on a disk. A Silicon Graphics O2 R10k played back the images during the experiment. The refresh rate of the monitor was 90 Hz and each movie image was presented for three consecutive frames. The mean luminance of the display was 56 cd/m^2. Stimulation was monocular to the dominant eye (the other eye was occluded).

The individual frames shown have been stored in the directory ringach/movie_frames. Each movie clip and segment have are assigned a different directory. Thus, the subdirectory movieNNN_MMM.images contains all the images corresponding to movie_id = NNN and segment_id = MMM. The movie_id takes one of four possible values (0, 1, 2 and 3) corresponding to the different movies. The segment_id takes one of 30 possible values (from 0 to 29) corresponding to the different segments sampled for each movie.

Within each subdirectory each frame is stored separately as a jpeg image. The format of the file name is movieNNN_MMM_KKK.jpg. This frame corresponds to movie_id = NNN, segment_id = MMM and frame number KKK. There are about 900 frames in each movie which represents 30s of stimulation.

Frame #k in the image was presented during the interval $(kT, (k+1)T)$, where T represents the refresh rate of the frames (not the monitor) and equals $T = \frac{3}{90Hz} = 33.33ms$.

Reading an image is simple done using the `imread()` function in Matlab:

```
>> img = imread('movie000_004_047.jpg');
>> imshow(img)
```

Will show the 48th frame (remember the first index is zero) in the movie #0 segment #4.

You may note some artifacts in the images. These come from the sampling itself and include effects of interlacing evident in moving objects and artifacts from the compression algorithm. The frames you obtain, however, are *exactly* the ones that were shown during the experiments.

The data: We provide the data in a single Matlab file that contains a single structure pepANA:

```
>> load  ac1_u005_007.mat

>> pepANA =

           config: [1x1 struct]
           module: [1x1 struct]
         sgiparam: [1x13 struct]
           looper: [1x1 struct]
          aborted: 0
              old: 0
    listOfResults: {1x120 cell}
    no_conditions: 120
        Condition: 120
           Repeat: 1
```

For the purpose of simplicity we can ignore many of the fields and jump directly to the basic data structure, contained in `ListOfResults`. This is a cell array. Each entry in the array corresponds to a different *condition.* A condition is a unique set of parameters that defines a visual stimulus. A single condition that be repeated more than one time in the experiment, leading different *repeats.* The total number of conditions in the experiment is to be found in `pepANA.no_conditions`. This number should equal the length of `ListOfResults`.

Let us look at the first entry in `pepANA.listOfResults`:

```
>> pepANA.listOfResults{1}

ans =

       condition: [0 0 0 0 0]
       symbols: {'movie_id'  'segment_id'}
       values: {[0]  [0]}
       noRepeats: 1
       repeat: {[1x1 struct]}
```

Different conditions in our system are normally generated by changing the values of some of the parameters of the visual stimuli. The field `symbols` contains a cell array with the

names of the variables that are being changed. The field `values` contains the actual values of these variables. Thus, in the present example, the first condition is such that the variables are assigned the values `movie_id=0` and `segment_id=0`. Further, one can see that the total number of repeats of this condition is 1, as specified in the field `noRepeats`. Finally, the field repeat is a cell array of structures which contains the actual data for each repeat of this condition. In this case there is only one repeat.

If we take a look at the data here we will see the following:

```
>> pepANA.listOfResults{1}

ans =

       timeTag: 0
       tzero: 1.2954e+005
       sync: 0
       syncCount: 1
       data: {1x25 cell}
```

Here, `timeTag` represents the ordinal number of this repeat within the entire experiment starting with zero. In this case, `timeTag=0` indicates this was the first stimulus of the entire experiment. Ignore all the other fields for now.

Finally, we get to the good stuff. The field `data` is a cell array that contains the waveforms and arrival times for the electrodes listed in `pepANA.elec_list`. Take a look at the first entry:

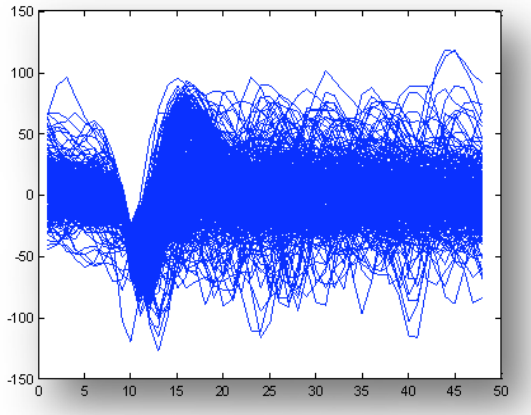
```
>> pepANA.listOfResults{1}.repeat{1}.data{1}

ans =

    [1x721 double]    [48x721 int8]
```

The first entry here contains arrival times in seconds from the beginning of the stimulus trial. The second entry contains 48 samples of waveforms snippets corresponding to each of the arrival times. In our experiments we set the thresholds for each channel at a low setting, so both putative spikes and some of the background noise have some likelihood of crossing the threshold. All the threshold crossings are detected as events and their time

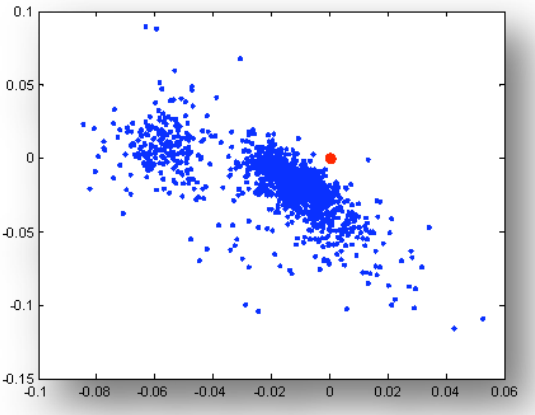
and a waveform snippet around this event (of 1.6ms long sampled at 30khz) is saved to disk (note they are the waveforms are in `int8` precision).

Lets take a look at waveforms from one electrode...

```
>> q = pepANA.listOfResults{1}.repeat{1}.data{13}{2};
>> plot(q)
>> figure(gcf)
>> plot(q,'b')
```



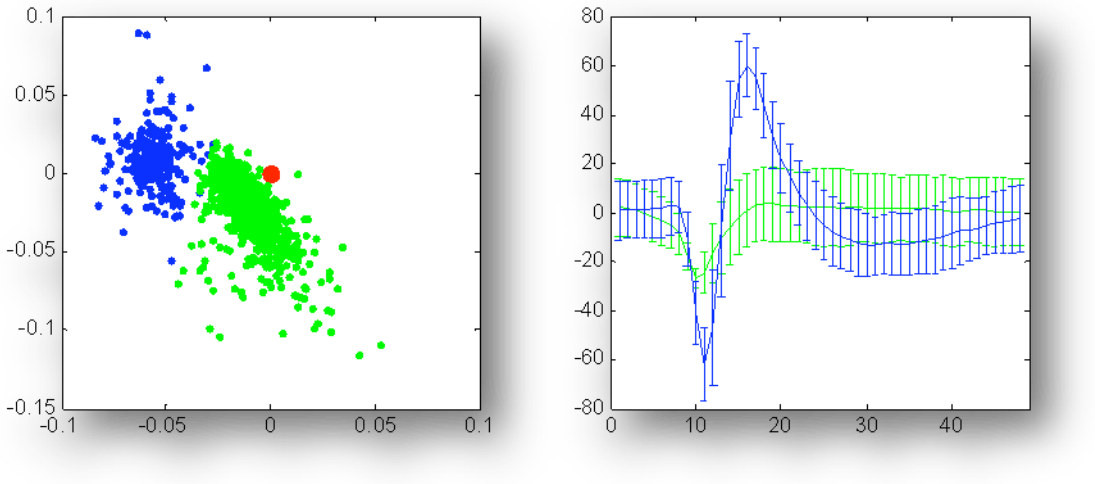
These are all the waveforms that crossed threshold superimposed one on top of each other. A simple SVD analysis reveals that there are two clouds of data points.



```
>> [u,s,v] = svd(double(q));
>> plot(v(:,1),v(:,2),'.','markersize',10); hold on; plot(0,0,'r.','markersize',25)
```

Note that there is a cloud of dots close to the origin (denoted by the red dot). This cloud of dots is simply the noise. These are signals that randomly crossed the threshold and got saved. However, one can see a clearly segregated cloud of dots that is away from the noise cloud. Sometimes, when the data are clean, a simple k-means clustering algorithm can be used to automatically classify the waveforms:

```
idx = kmeans(double(q'),2); %% cluster waveforms in 2 classes using k-means
idx1 = find(idx==1); idx2=find(idx==2);
subplot(1,2,1)
plot(v(idx1,1),v(idx1,2),'b.','markersize',10); hold on;
plot(0,0,'r.','markersize',25)plot(v(idx2,1),v(idx2,2),'g.','markersize',10);
subplot(1,2,2)
errorbar(mean(q(:,idx2)'),std(q(:,idx2)'),'g'); hold on;    % plot mean waveforms +/- 1 sd
errorbar(mean(q(:,idx1)'),std(q(:,idx1)'),'b')
xaxis([0 49])
```



Thus, all the data necessary to satisfy your curiosity about the quality of single spike isolation is available to you. This is the reason the raw data is provided instead of our group doing the spike sorting. We only screened the data to include those electrodes for which a basic PCA analysis suggested a high-likelihood that spikes could be isolated. Of

course, you should sort data across the entire experiment (not just in one repeat as done in the example above).

Natural image sequences data:

Experiments on natural image sequences loop over two quantities, movie_id and segment_id. These indicate the particular movie clip that was presented in that case and you can access the individual frames as explained above.

To find out the indices with the movie frame that appeared, for example, 100ms before a spike in one of the electrodes you can simply do:

```
T = 3/90;          % effective duration of each frame in seconds.
spk = pepANA.listOfResults{c}.repeat{1}.data{13}{1};
spk = spk - 60e-3; % compute frames present on screen 60ms before the spike;
frame_idx = floor(spk/T);
```

This basic introduction should allow you to start exploring the dataset. Enjoy! If you have any questions, please post them on the crcns.org forum for the data set (preferable) or contact me at dario@ucla.edu.