Spatial Firing Rate Maps
NEUROSC 99

Jesse Hurtado

Spring 2023

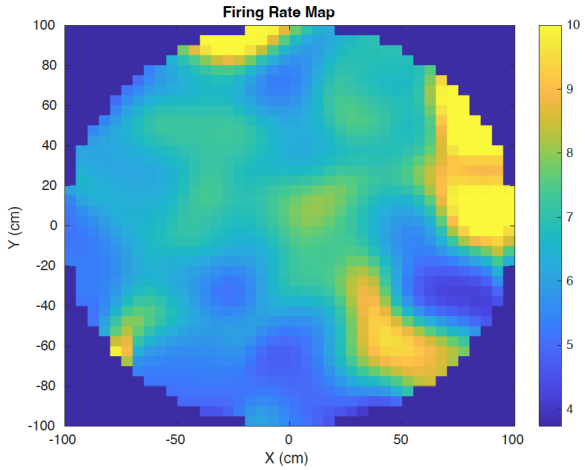
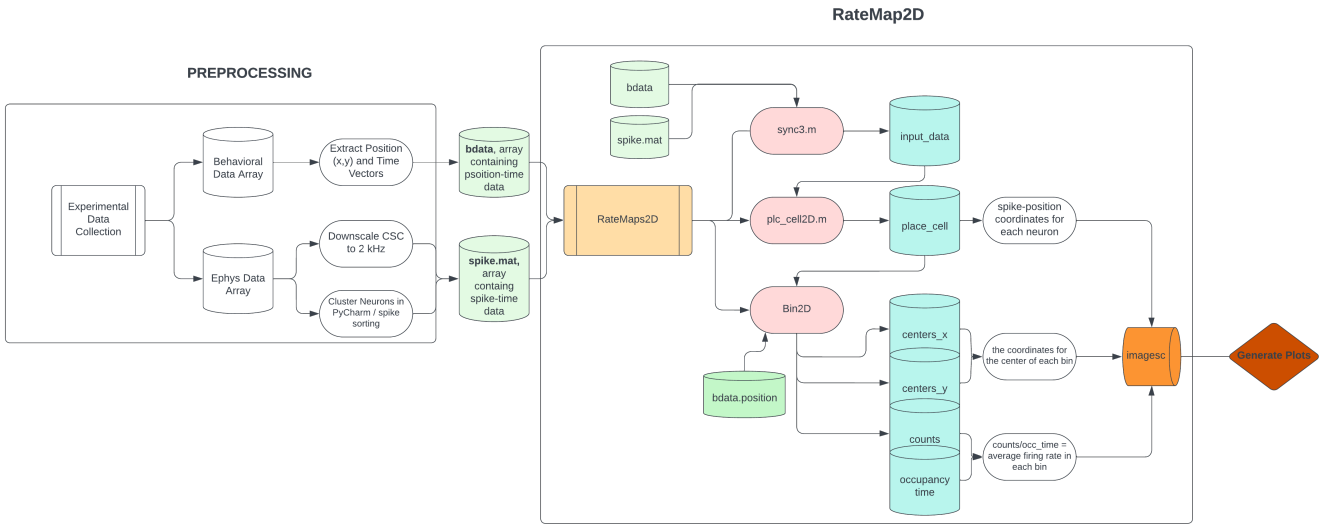


Figure 1: Plot of spatial neural activity.

0.1 Project Overview

This project aimed to create a two-dimensional histogram illustrating the firing patterns of dorsal hippocampal CA1 place cells in mice navigating a round table with a radius of 100cm. During the experiments, rats freely foraged in the designated environment while electrophysiological data were collected through implanted electrodes. Simultaneously, behavioral data, including position information, was recorded using an overhead camera. The primary objectives were to gain hands-on experience in spatial mapping of neuron firing, analyze the methodologies and tools employed for this purpose, and successfully generate a plot using authentic experimental data. The project contributed to a deeper understanding of the neural mechanisms underlying spatial navigation, providing practical insights into the methods of spatial mapping and the integration of electrophysiological and behavioral data. All the analysis done for this project was done in MATLAB.

0.1.1 Algorithm Map



0.2 RateMap2D.m

The RateMaps2D function generates two-dimensional rate maps for neurons based on synchronized spike and behavioral data. It takes as inputs the tetrode number, behavioral data, bin size, and smoothing parameters, with defaults for unspecified values. The function first extracts the desired data from their respective file locations, processes the data with sync3.m and plc_cell functions, calculates the number of neurons, and iterates over each neuron to perform spatial binning using the Bin2D function. The resulting binned data is stored in the variable spikes for calculating the mean firing rate in each bin by dividing the firing rate by the occupancy time in each bin, and plotting a 2D histogram of position and firing rate based on the centers of each spatial bin and the mean firing rate in that bin. The function provides a solution for examining spatial firing patterns in neural data. This is the main function that, with a minimum of 2 arguments, will compute the spatial rate map displayed in the image above. It calls upon the rest of the functions detailed in this report throughout the body of the code.

0.2.1 Arguments

```
function ratemap = RateMaps2D(TT,bdata, bin_size, smooth_kernel, dimension, circular)
if nargin < 6; circular = true; end
if nargin < 5; dimension = [200 200]; end      %cm
if nargin < 4; smooth_kernel = [7.5 7.5]; end %cm
if nargin < 3; bin_size = [5 5]; end           %cm
```

The function sets default values for parameters that are not provided. It then proceeds to generate rate maps for neurons based on the provided data. The default values for bin size, smooth kernel, dimension, and circularity of the environment can be modified if needed. The resulting rate maps are smoothed using a Gaussian kernel, and the circular option adjusts the representation of spatial data based on the environment shape.

0.2.2 Load Data

```
%% Load Data
string = sprintf('TT%d.spike.mat',TT);
load(string,'cluster_id','spike_time','session','subject','mean_rate')
[input_data, min_t] = sync3(bdata,cluster_id,spike_time);
bdata.Time(:,1) = (bdata.Time(:,1)-min_t)./1000000; % Time in seconds
input_data(:,1) = (input_data(:,1)-min_t)./1000000; % Time in seconds
neuron_num = size(cluster_id, 2);                   % Number of neurons
place_cell = plc_cell2D(input_data,neuron_num); % spike data place cells
```

- Load Spike Data: Generates a filename based on the tetrode number (TT) and loads spike-related data from a corresponding file (sprintf('TTd.spike.mat',TT)). The loaded variables include cluster_id (cluster identifiers), spike_time (spike timestamps), session, subject, and mean_rate.
- Synchronize Data: Calls the sync3 function to synchronize behavioral data (bdata) with the loaded spike data (cluster_id and spike_time). The synchronized data (input_data) and the minimum timestamp (min_t) are obtained.Min_time will be used at the end to remove low occupancy bins.
- Time Conversion: Converts time values in the behavioral data (bdata) and synchronized spike data (input_data) from microseconds to seconds. Adjusts the first column of bdata.Time and input_data accordingly.
- Number of Neurons: Determines the number of neurons (neuron_num) based on the number of columns in cluster_id. There is one cluster_id for each neuron, a vector of 1's and 0's, where 1 corresponds to a detected spike. Clust_id is used as an index for extracting the spike locations used in plc_cell, for as many neurons as there are columns in cluster_id.

- 2D Place Cell Processing: Calls the plc_cell2D function with the synchronized spike data (input_data) and the number of neurons (neuron_num). The result, place_cell, contains processed 2D place cell information for each neuron.

0.2.3 Bin Data

```
for g = 1:neuron_num
    clear counts centers_x centers_y valid
    scaling = 1;
    [counts,centers_x, centers_y] = Bin2D(place_cell{1,g}(:,2),
    [-dimension(1)/2 +dimension(1)/2], ...
        bin_size(1), place_cell{1,g}(:,3), [-dimension(2)/2 dimension(2)/2],
        bin_size(2), scaling);
    spikes = counts;
```

This loop takes each neuron's position data from place_cell, performs 2D binning using the Bin2D function, and stores the resulting binned data in the spikes variable. The centers_x and centers_y variables contain the center coordinates of the bins along the x and y axes, respectively. For occupancy time, the same function is called but with position-time data in place of position-spike data.

0.2.4 Plot

```
%% Generate Plot
      if circular
        radial_cutoff_cm = dimension(1)/2;
        [a,b] = meshgrid(centers_x, centers_y);
        radial_distance = sqrt(a.^2 + b.^2);
        spikes((occ_time < min_time)) = 0;
      end
      spikes = smooth_gaussian_2d(centers_x, centers_y, spikes, smooth_kernel(1),
      smooth_kernel(2));
      rate = spikes ./ occ_time;
      rate((radial_distance > radial_cutoff_cm)) = nan;
      %rate(rate > 10) = 10;
      figure
imagesc(centers_x, centers_y, rate)
axis xy
colorbar
title(sprintf('Firing Rate Map Neuron %g));
xlabel('X (cm)')
ylabel('Y (cm)')
```

The resulting firing rates, representing the mean firing rate, are visualized using an image plot. To improve visualization, low occupancy bins are excised. The colormap effectively illustrates the distribution of mean firing rates, aiding in the interpretation of spatial firing patterns. The X and Y axes are labeled in centimeters, and a color bar provides a reference for the firing rate scale.

0.3 sync3.m

The function ensures synchronization between behavioral data (bdata) and neural spike data by interpolating the position data at spike timestamps. The resulting synchronized data (input_data) contains spike-position coordinates.

```
function[input_data, min_t]=sync3(bdata, cluster_id, spike_time)
spike_time = double(spike_time);
bdata = [bdata.Time bdata.Position];
pos_tim = bdata;
```

```
ind = abs(diff(pos_tim(:,1)))==0;
pos_tim(ind,:) = [];
posx = interp1(pos_tim(:,1), pos_tim(:,2:end), spike_time);
input_data = [spike_time posx cluster_id];
[r ~] = find(isnan(input_data));
input_data(r,:)=[];
min_t = min(input_data(1,1));
```

0.4 plc_cell2.m

Each spike.mat file contains the spiking data for multiple neurons that had been clustered and sorted prior to analysis. This function creates a new array where each cell contains the interpolated position-spike data for each neuron, the total number of neuron is stored in a variable neuron_num. The plc_cell function processes spiking data from multiple neurons, stored in the input_data matrix, and organizes it into a cell array named place_cell. Each cell in place_cell corresponds to a distinct neuron and contains the interpolated position-spike data only for the time points when spikes occurred (non-zero cluster IDs).

```
function [place_cell] = plc_cell(input_data, neuron_num)
place_cell = cell(1, neuron_num);
for j = 1:neuron_num;
    rr = input_data(:,j+3)==0;
    temp = input_data;
    temp(rr,:)=[];
    place_cell{1,j}=temp(:,1:4);
end
```

0.5 Bin2D.m

The Bin2D function bins a two-dimensional matrix based on specified bounds and bin sizes for both x and y coordinates. The data points are optionally filtered using the valid parameter. It takes as arguments the array to be binned (either position-spike or position-time), the desired bin size (will default if not changed) and the bounds for each parameter. It then calculates spike counts or occupancy time for each bin in the 2D space. The results are returned as the counts matrix, and the centers of the bins are provided as centers_x and centers_y.The function histcn is called to compute an "n-dimensional" histogram and get the center points of each bin.

```
function [counts,centers_x, centers_y] = Bin2D(measure_x, bounds_x, binsize_x,
measure_y, bounds_y, binsize_y, scaling, valid)
    if nargin  < 8; valid = true(size(measure_x)); end
    if nargin < 7; scaling = 1; end
    edges_x = bounds_x(1) : binsize_x : bounds_x(2);
    edges_y = bounds_y(1) : binsize_y : bounds_y(2);
    centers_x = 0.5 * (edges_x(1 : end - 1) + edges_x(2 : end));
    centers_y = 0.5 * (edges_y(1 : end - 1) + edges_y(2 : end));
    measure_x = measure_x(valid);
    measure_y = measure_y(valid);
    % make sure the dimensions of the data are correct
    if isrow(measure_x); measure_x = measure_x'; end
    if isrow(measure_y); measure_y = measure_y'; end
    temp_occ = histcn([measure_x measure_y], edges_x, edges_y);
    temp_occ = temp_occ(1 : length(centers_x),  1 : length(centers_y));
    counts = temp_occ * scaling;
    counts = squeeze(counts);
end
```

0.6 smooth_gaussian_2D.m

This function applies 2D Gaussian smoothing to the input matrix using a Gaussian kernel. It starts by creating the Gaussian kernel and then (optionally) upsamples the input matrix. After padding the matrix to handle boundary conditions, it performs 2D convolution using the Gaussian kernel. Finally, the padding on the matrix is removed and it performs a 2D interpolation over the matrix after the convolution to ensure that the output has the same dimensions as the input. The result updates the original counts and occupancy time vectors with the smoothed data.

0.6.1 mvnpdf

This line of code calculates the values of a 2D Gaussian probability density function (PDF) at each point on the grid defined by the padded coordinate matrix. The results are stored in a column vector g. Each element of g corresponds to the PDF value at the corresponding point in the grid.

0.6.2 interp2

Interpolation for 2D gridded data in meshgrid format. This function returns interpolated values of a function of two variables at specific query points using linear interpolation. The results always pass through the original sampling of the function. X and Y contain the coordinates of the sample points. It is used to compute the 2D interpolation over the convolution of the padded matrix and the Gaussian Kernel, g.

0.6.3 conv2

Performs a 2D convolution between the padded matrix "pad" and Gaussian kernel "g".

0.6.4 Gaussian Smoothing and Interpolation

The Gaussian smoothing function is based on the equation for a 2D Gaussian Distribution:

$$G(x,y) = \frac{1}{2\pi\sigma_x\sigma_y} e^{-\frac{x^2}{2\sigma_x^2} - \frac{y^2}{2\sigma_y^2}}$$

where (x,y) are the coordinates of locations where spikes were detected and

$$\sigma_x \, \sigma_y$$

are the standard deviations, respectively. The function interp1 is based on linear interpolation for 1D data and is based on the linear interpolation formula:

$$y = y_1 + \frac{(x - x_1) \cdot (y_2 - y_1)}{x_2 - x_1}$$

The function interp2 is based on billinear interpolation for 2D data and is based on the bilinear interpolation formula:

$$f(x,y) = (1-\alpha)(1-\beta)f(x-1,y-1) + \alpha(1-\beta)f(x_2,y_1) + (1-\alpha)\beta f(x_1,y_2) + \alpha\beta f(x_2,y_2)$$

where

$$\alpha = \frac{x - x_1}{x_2 - x_1}$$

and

$$\beta = \frac{y - y_1}{y_2 - y_1}$$

0.7 Parameters and Variables Quick Reference

- TT: Tetrode number.
- bdata: Behavioral data array containing position and time vectors.
- bin_size: Size of spatial bins in centimeters
- smooth_kernel: Standard deviations of the Gaussian smoothing kernel.
- dimension: Size of the spatial dimension (default is 200x200 cm).
- circular: Boolean indicating whether the spatial environment is circular.
- input_data: Synchronized data containing spike, position, and cluster_id.
- min_t: Minimum timestamp, used for removing low occupancy bins.
- cluster_id: Cluster identifiers, vectors of 1's and 0'z where a 1 indicates a spike occurred at that index value. Each clustered neuron has a cluster_id.
- spike_time: Spike timestamps, the time during the trial at which a spike was detected.
- session: Session information.
- subject: Subject information.
- mean_rate: Mean firing rate of the clustered neuron.
- neuron_num: Number of neurons, the number of columns of cluster_id.
- place_cell: Processed 2D place cell information.
- spikes: Binned and smoothed spike histogram.
- counts: Binned counts from Bin2D function.
- centers_x: X coordinates of bin centers.
- centers_y: Y coordinates of bin centers.
- scaling: Scaling factor for binning.

0.8 Conclusion

This project showcased a diverse set of computational skills to analyze and visualize neuronal activity patterns. Key concepts included the application of Gaussian smoothing to enhance the clarity of firing rate maps, interpolation techniques to synchronize behavioral and electrophysiological data, and matrix indexing for efficient data manipulation, and time series data analysis. The use of functions, such as the 2D Gaussian smoothing function and the synchronization function 'sync3', demonstrated modular and reusable code design. For loops were employed for iterating over neurons, facilitating scalable data processing. Overall, the project underscored the effective utilization of computational tools, encompassing matrix operations, functions, and control structures, to gain insights into the spatial firing characteristics of neurons behavioral tasks. All codes can be found in the repository for this project at 2DRateMaps