Rate Map

Data

For rate maps you need position and trace data across time (simulated or real)

```
data = load('data.mat');
day = 1;
% position
p = data.position{day}; % (x-y coordinates) x time
% trace activity vectors
trace = data.trace{day}; % cells x time
```

Step 1 - Discretize

You need to decide on bin and smoothing kernel sizes. We will write code flexibly to take varying inputs. First, we need to create a discrete set of states within the environment.

```
envSize = ceil(max(p,[],2)); % extract environment dimensions
bins = 25; % can either be 1 if nxn or [n,m]  if nxm
bins = zeros(2,1) + bins';
a = envSize./bins; % this is what we need to divide position
% by to discretize
dp = floor(p./a)+1; % now position is x:1-bins,y:1-bins

check = [min(dp(:)) , max(dp(:))]; % check range is within bins
```

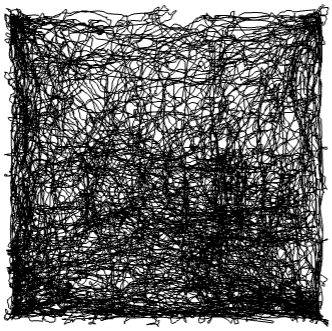
Step 2 - Compute occupancy

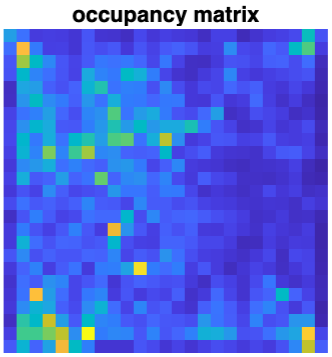
When computing rate maps, we normalize activity by state occupancy

```
% convert to linear position and count number of times in each bin
inPix = histc(sub2ind(bins',dp(1,:)',dp(2,:)'),[1:prod(bins)]);
% reshape
inPix = reshape(inPix,bins(1),[]);

% Another vectorized way to do this is to generate a matrix of
% state==time and compute the sum ^ the above way is a bit more robust to
% number of states computationally

% inPix = [1:bins(1)*bins(2)] == sub2ind(size(zeros(bins')),dp(1,:)',dp(2,:)');
% inPix = reshape(sum(inPix,1),bins(1),[]);
```

Visualize occupancy matrix - heat map of where the animal was in the environment

```
doFig([600 200])
subplot(121)
plot(p(1,:),p(2,:),'k')
axis square, axis off
axis([0, envSize(1), 0, envSize(2)])
subplot(122)
imagesc2(inPix)
set(gca,'YDir','normal')
axis square, axis off
title('occupancy matrix','fontweight','bold')
```





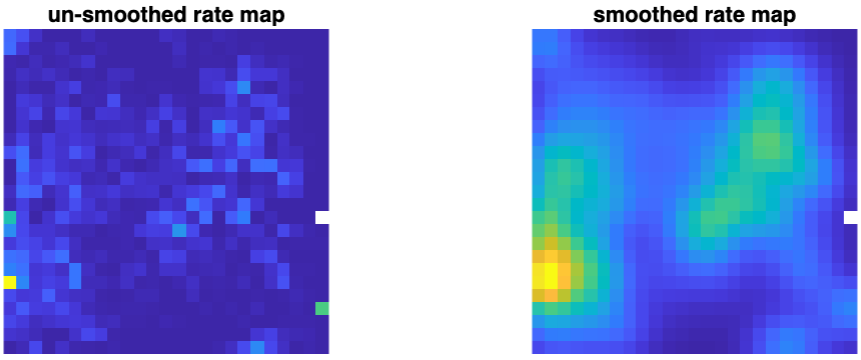
Step 3 - Iteratively compute state-wise cell mean firing rates

For each cell we compute the average firing rate in each pixel relative to dwell time and apply smoothing.

```
% Initialize maps
maps = nan([bins',numel(trace(:,1)) ]);

% I'm going to iterate through states for all cells simultaneously
% This is faster than doing individually for each cell
for i = [repelem([1:bins(1)], bins(2))', repmat([1:bins(2)], 1, bins(1))']'
    isInPix = dp(1,:)== i(1) & dp(2,:)== i(2); % when is animal in each state
    % sum firing rate
    maps(i(1), i(2), :) = sum(trace(:,isInPix),2);
end
maps = maps ./ inPix; % normalize
maps(isnan(maps)) = 0;
smoothing = 1.5; % standard deviation
smoothedMaps = gFilt(maps,bins,smoothing); % smoothed maps (gaussian filter)
% NaN out unvisited pixels
smoothedMaps = reshape(smoothedMaps,[],numel(smoothedMaps(1,1,:)));
smoothedMaps(~inPix,:) = NaN;
smoothedMaps = reshape(smoothedMaps,bins(1),bins(2),numel(smoothedMaps(1,:)));
maps(~inPix) = NaN;
unSmoothedMaps = maps;
```

Visualize rate maps

```
doFig([600 200])
subplot(121)
imagesc2(unSmoothedMaps(:,:,1)) % makes NaN's white
axis square, axis off
title('un-smoothed rate map','fontweight','bold')
subplot(122)
imagesc2(smoothedMaps(:,:,1))
axis square, axis off
title('smoothed rate map','fontweight','bold')
```



Faster

You can also skip the occupancy computation and just compute the mean firing rate in each pixel - which automatically

accounts for dwell time. I like to compute occupancy as well, in case It's ever needed

```
% maps = nan([bins',numel(trace(:,1)) ]);
% for i = [repelem([1:bins(1)], bins(2))', repmat([1:bins(2)], 1, bins(1))']'
%     isInPix = dp(1,:)== i(1) & dp(2,:)== i(2);
%     maps(i(1), i(2), :) = mean(trace(:,isInPix),2);
%     maps(isnan(rateMaps)) = 0;
% end
```