Boundary Vector Cell Successor Representation (BVC - SR)

De Cothi, W., & Barry, C. (2020). Neurobiological successor features for spatial navigation. Hippocampus, 30(12), 1347-1355.

Hartley, T., Burgess, N., Lever, C., Cacucci, F., & O'keefe, J. (2000). Modeling place fields in terms of the cortical inputs to the hippocampus. Hippocampus, 10(4), 369-379.

Lever, C., Burton, S., Jeewajee, A., O'Keefe, J., & Burgess, N. (2009). Boundary Vector Cells in the Subiculum of the Hippocampal Formation. Journal of Neuroscience, 29(31), 9771–9777.

Code Modified and Adapted from:

https://github.com/willdecothi

Step 1 - Load Data

Download the "bvcsr" file. This contains the data we need and subfolders contain functions we will need.

This data file contains 9 days of consecutive openfield calcium imaging recordings in a mouse. Recordings are 60 minutes long @ 30 fps ~ 108000 frames.

The data is a struct that contains 6 fields. We will only be using position which is a cell that contains matrices size: 2 x time for each day. This corresponds to x-y position of the mouse through time.

```
data = load('data.mat'); % load data
day = 1; % takes path from given day
p = data.position{day}; % index cell to get path for that day
```

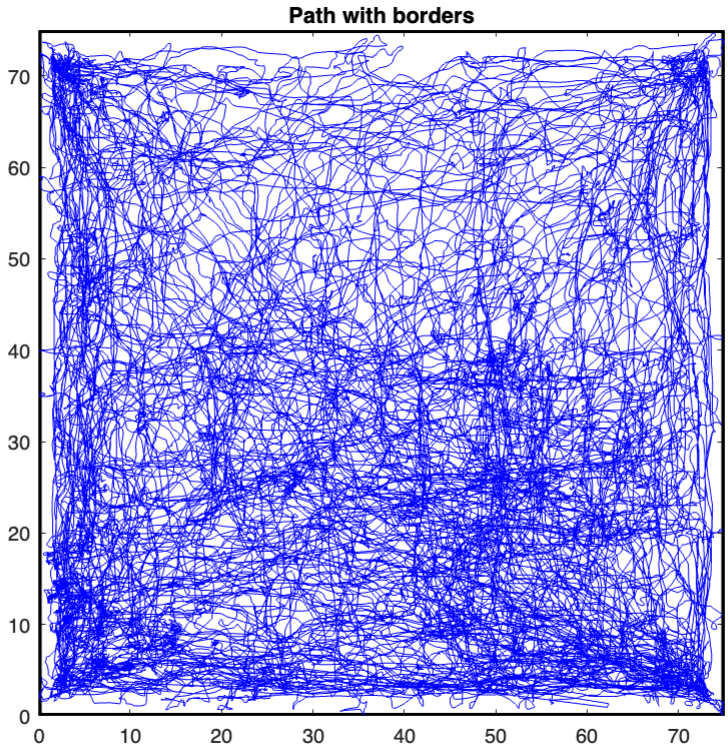
Step 2 - Set up environment

First, we need to set up the environment boundaries to compute distances and angles for boundary vector cells. Since the environment is square, you only need one size value corresponding to the max value in the path. (Should be ~75)

```
envSize = ceil(max(p,[],'all')); % Maximum size of environment
borders = [0,0,0,envSize;...
    0,envSize,envSize,envSize;...
    envSize,envSize,envSize,0;...
    envSize,0,0,0]; % creates border vectors
```

Plot the environment:

```
% plot environment
figure
plot([borders(:,1) borders(:,3)],...
    [borders(:,2) borders(:,4)],'k','linewidth',3)
hold on
plot(p(1,:),p(2,:),'b')
scale = [0,envSize]; % axes
axis(repmat(scale,1,2)) % set axes
axis square
title('Path with borders','fontweight','bold')
```



Step 3 - Simulate BVCs

First, we need to set up the parameters for our BVC Model

```
 n = 160;
 numGroups = 10; % 10 groups with linearly space distal and angular tuning
 groupSize = n./numGroups; % Cells per group
 theta = linspace(0.1,2*pi,groupSize)'; % angular tuning
 bvcDistance = envSize; % maxim distance tuning
 distance = linspace(1,bvcDistance,groupSize)'; % distance tuning
 stdev = zeros(n,1)+3; % standard deviation
 tuning = [repmat(theta,numGroups,1),repelem(distance,numGroups,1),stdev]; %
concatenate
```

Now we can simulate our BVCs

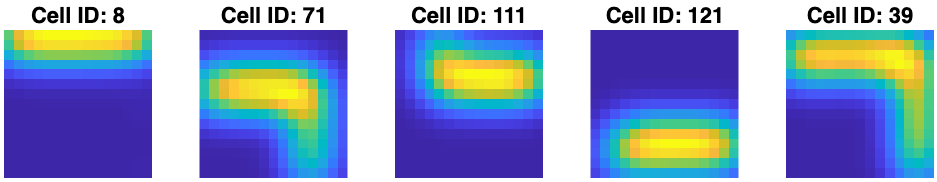
```
 bvc = NaN(n,numel(p(1,:))); % number of cells x time matrix
 center = 0; % we move the environment around the animal
 % Iterate to simulate
 tic
 for i = 1:numel(p(1,:))
     % the environment moves and is centered with the agent as the zero point
     toCenter = repmat([p(1,i) - center, p(2,i) -
center],numel(borders(:,1)),2);
     centered = borders-toCenter;
     % generate vectors for each bvc to compute distance to wall
     vectors = [repmat([center center],n,1) cos(tuning(:,1)).*(envSize*2)...
             sin(tuning(:,1)).*(envSize*2)];
     % fast line intersect to compute vector-border intersection
     [~, distances] = fastLineIntersect(vectors,centered);
     % compute minimum distance to boundary
     mind2b = nanmin(abs(distances),[],2);
     % compute probability of bvc firing
     bvc(:,i) = normpdf(mind2b,tuning(:,2),tuning(:,3));
 end
```

Generate BVC ratemaps

```
 bins = 15; % number of bins
 smoothing = 2; % standard deviation for gaussian smoothing
 % Compute rate maps
 [smoothedMaps, unSmoothedMaps, ~] =...
     getMaps(bvc, p, envSize, bins, smoothing); % function to compute rate maps
```

Visualize BVC rate maps

```
% Visualize BVC's
sample = randsample(1:n,5);
doFig([600 100])
for i = 1:numel(sample)
    subplot(1,5,i)
    imagesc(smoothedMaps(:,:,sample(i)))
    axis square
    axis off
    title(['Cell ID: ' num2str(sample(i)) ],'fontweight','bold')
end
```



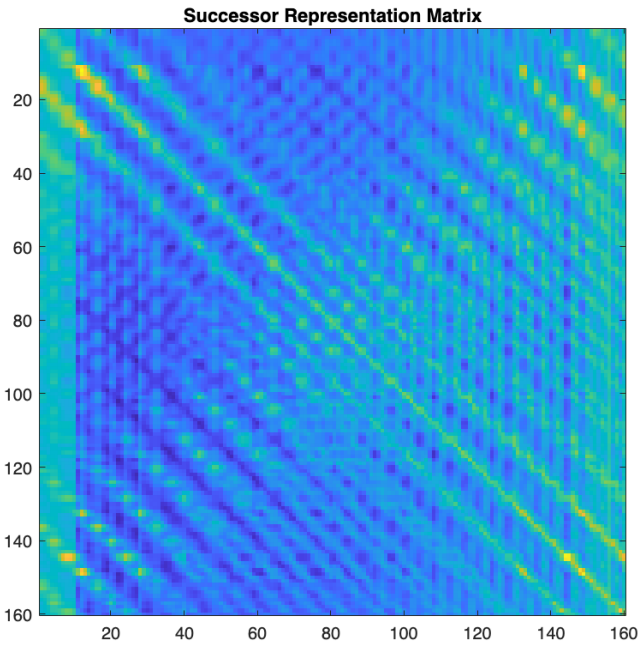
Step 4 - Compute SR Matrix

The SR (M) represents the short- and long-range transition statistics for all pairwise boundary vector cells.

```
a = envSize ./ bins;
dp = floor(p./a)+1; % discretize position

alpha = .2; % learning rate
gamma  = 0.95; %
lag = 1; %

M = ones(n); % Initialize Successor representation

% Online SR Learning Rule
for t = 1:numel(dp(1,:))-lag
    % Population vector activity vectors at each time step
    phi = smoothedMaps(dp(1,t), dp(2,t),:);
    phi_ = smoothedMaps(dp(1,t+lag), dp(2,t+lag),:);
    % Compute Sr
    M = M + alpha * (phi(:) + gamma * M * phi_(:) - M * phi(:)) * phi(:)';
end
```

Visualize Successor representation matrix

```
% Visualize SR
figure
imagesc(M)
axis square
title('Successor Representation Matrix','fontweight','bold')
```



Step 5 - Extract Successor Features

```
successorCells = nan(bins,bins,n); % corresponds to number of BVS
tic
for i = 1:n
    w = M(i,:)'; % each row of the SR functions as a set of weights

    maps = reshape(smoothedMaps,[],1) .* repelem(w,bins^2); % weight maps
    placeMap = sum(reshape(maps,size(smoothedMaps)),3); % sum across all maps

    % Filter maps to activity > 80% Max Firing Rate
    placeMap = placeMap./max(placeMap(:));
    placeMap(placeMap < (max(placeMap(:))*.8)) = 0;
    placeMap = gFilt(placeMap,bins,1.5); % smooth maps 1.5 std
    successorCells(:,:,i) = placeMap;
end
```

Examples of extracted successor features

```
sample = randsample(1:n,5);
doFig([600 100])
for i = 1:numel(sample)
    subplot(1,5,i)
    imagesc(successorCells(:,:,sample(i)))
    axis square
    axis off
    title(['Cell ID: ' num2str(sample(i)) ],'fontweight','bold')
end
```