Neuron Session Alignment in Calcium Imaging Data

Yangyi Lu

28 November 2017

Abstract

Background. In vivo calcium imaging through microendoscopic lenses enables imaging of previously inaccessible neuronal populations deep within the brains of freely moving animals. When analyzing long term neuronal activities to understand neuronal dynamics over time, neuron images collected in multiple sessions need to be aligned, which means neurons need to be tracked along all time sessions. However, there exist many inconsistencies in the images of different sessions, due to things like rotations and shifts of the camera, and natural changes in the brain, which leads to the alignment problem nontrivial. Currently the aligning is done by hand, which is very time consuming.

Aim. Our goal is to design an algorithm that can align neurons in different time sessions in order to help neuroscience researchers analyze long term neuron activities.

Data. The dataset we use is calcium imaging data from a pharmacological experiment in which 11 male Drd1a-tdTomato mice received IP injections of either SKF38393 (D1 specific agonist) or saline vehicle in a 2 day crossover experimental design. Calcium data was acquired with an Inscopix nVista HD microscope with an acquisition rate of 20Hz. We apply CNMF-E (Zhou et al., 2016) and PCA/ICA (Mukamel et al., 2009) to get neuron contours for every neuron movie as our input data.

Method. We apply shape context method (Belongie et al. (2001)) to encode neurons to feature vectors and focus on matching neurons across sessions by specifying a cost function to measure the dissimilarities between pair of neurons. We detect and analyze the dissimilarity based on distances between pairs of neurons and neuron shapes. Next, we form the problem into linear assignment problem to determine the final matching results.

Results. Our proposed algorithm can align neurons across sessions accurately up to 90%.

Conclusion. Shape context can extract very effective features for our matching problem and the optimization problem set up in our proposed method can help us match neurons accurately. We can provide all potential matching pairs sorted by the confidence that we have for the matching, so that researchers can choose the hard line by themselves that indicates up to where, they want to believe the matching results.

Keywords: Neuron Alignment, Calcium Imaging

DAP Committee members:

Robert E. Kass ⟨*kass@stat.cmu.edu*⟩ (Machine Learning & Statistics Department);
Jordan Rodu ⟨*jordan.rodu@gmail.com*⟩ (Statistics Department);
Jared S.Murray ⟨*jmurray.1022@gmail.com*⟩ (Statistics Department);

1 Introduction

Monitoring the long run activity of large-scale neuronal ensembles during complex behavioral states is fundamental to neuroscience research. In vivo calcium imaging through microendoscopic lenses enables imaging of previously inaccessible neuronal populations deep within the brains of freely moving animals, so microendoscopy has lots of potential applications across neuroscience field. In order to do long run analysis of neuronal activities based on microendoscopic data, we need to get neuron contours from microendoscopic data and track neurons in different time sessions.

However, neuron images are noisy so that it is hard to get clear neuron contours and even after getting all the neuron contours, the contours still cannot be used directly to identify the same neurons in different sessions. When data is collected in multiple sessions, there exist many inconsistencies in the images, due to things like rotations and shifts of the camera, and natural changes in the brain. There exists some technical limitations that the camera has to be taken out of the brain everyday in order to be maintained in a good situation and cannot be put exactly at the same position everyday. Also, neurons showing up in one session may not show up in another. Neurons may not fire in another day and if the distance from the camera to neurons changes, the range of neurons that can be covered in the image will also change. Thus, totally understanding the activities of neuron behaving is really tough (Zhou et al., 2016). Currently neurons in different sessions are matched by putting neuron cloud images on top of each other in PhotoShop and moving the images manually for larger covering area, which indicates proper matchings. But when there are neuron images from hundreds of days, it is unfeasible for human beings to track the neurons one by one by hand.

We proposed an algorithm using following techniques to automated align neurons in different sessions (c.f. Section 3):

- Extract neuron contours from noisy neuron images using constrained nonnegative matrix factorization for microendoscopic data (CNMF-E) (Zhou et al., 2016) to get neuron contours.
- Apply shape context (Belongie et al., 2001) technique to encode neurons into feature space and develop a similarity score for neurons in different time sessions.
- Based on features and similarity scores for neurons, we formalize the problem into a linear assignment problem and apply Hungarian algorithm to get matching pairs.

Experiments on both simulated and real data to demonstrate the effectiveness of proposed method (c.f. Section 4 and Section 5).

1.1 Related Works

There exist many methods for extracting cellular signals from microendoscopic data, such as semi-manual ROI analysis (Barbera et al., 2016; Klaus et al., 2017) and PCA/ICA analysis (Mukamel et al., 2009). However, both approaches have well known flaws. For instance, ROI analysis does not effectively demix signals of spatially overlapping neurons, and drawing ROIs is laborious for large population recordings (Zhou et al., 2016). As for PCA/ICA analysis, it is a linear demixing method and therefore typically fails when the neural components exhibit strong spatial overlaps, which is exactly the case in microendoscopic setting (Pnevmatikakis et al., 2016).

Fortunately, one recent work - constrained nonnegative matrix factorization (CNMF) was proposed to simultaneously denoise, deconvolve, and demix calcium imaging data (Pnevmatikakis et al., 2016). Another new work constrained nonnegative matrix factorization for microendoscopic data (CNMF-E) improved this by making it adapted to microendoscopic data which has a much more complex background structure (Zhou et al., 2016). It can denoise the raw image well and get significant more accurate cellular signals from the denoised image for microendoscopic data. Thus, we apply CNMF-E to get clear neuron contours from raw dataset.

In order to get features describing the relative distance relation of neurons, we look into strategies of shape description. A number of different strategies have been tried, e.g. nearest-neighbor techniques after extracting principal components (Turk and Pentland, 1991; Murase and Nayar, 1995), convolutional neural networks (LeCun et al., 1998), and support vector machines (Mukamel et al., 2009; Burges and Schölkopf, 1997). Impressive performance has been demonstrated on datasets such as digits and faces. However, in our problem, a vector of pixel brightness values does not help, what we want is a feature that can represent "shape". Belongie et al. (2001) developed an approach called shape context that can satisfy our requirement. It describes the coarse arrangement of the rest of the point with respect to the point. This descriptor will be different for different points on a single points cloud P; however corresponding (homologous) points on similar points cloud P and P' will tend to have similar shape contexts. It can be applied to our problem since it can describe the spatial "distribution" (i.e. features) of neurons in neuron clouds.

By using shape context method to extract useful features, we formalize the matching problem into a linear assignment problem by developing a neuron shape similarity score. The Hungarian method (Kuhn, 1955) is a combinatorial optimization algorithm that solves the assignment problem (assign n tasks for n workers) in polynomial time. Later, an adaptive version of Hungarian method (Bourgeois and Lassalle, 1971) that can solve the unbalanece assignment problem (assign m tasks for n workers) come out. We apply this adaptive version of Hungarian to solve our assignment problem.

1.2 Problem Statement

Now we state our problem. Figure 1 shows neuron clouds in two different sessions. Our goal is to align neurons in these two clouds. Human can do this task by considering the neuron shapes and relative positions to identify the same neurons appears in the two time sessions and we want our algorithm help human on this task. In Figure 1, the matchings are $1-1$, $2-2$, $3-3$, $4-4$, $5-5$ and neuron 6 in left should be recognized as a missing neuron in the right plot.

1.3 Organization

In Section 2, we describe our data. In Section 1.2, we state our problem. In Section 3, we introduce our algorithm. In Section 4, we test our algorithm with simulated data. In Section 5, we test our algorithm on real data and we conclude in Section 6.

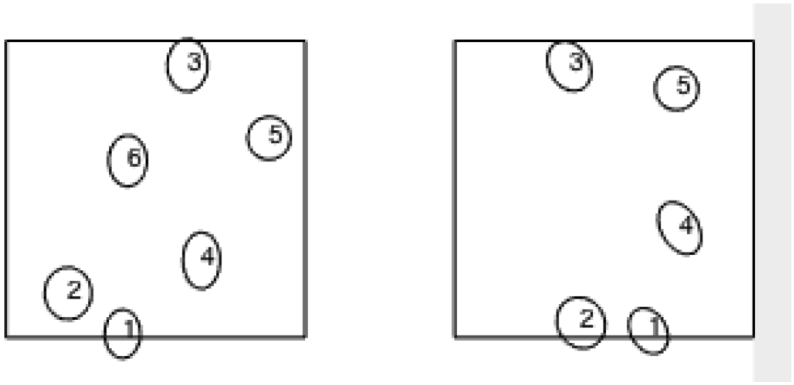


Figure 1: Two neuron clouds of two time sessions. All ellipses represent neurons here, numbers on top of the ellipses are neuron indices within each neuron cloud. In this example, neuron 6 in the first cloud is missing in the second neuron cloud and all neurons change their positions by rotation and translation and neuron shapes have tiny changes as well.

2 Data

The data we use is calcium imaging data. It is from a pharmacological experiment in which 11 male Drd1a-tdTomato mice received IP injections of either SKF38393 (D1 specific agonist) or saline vehicle in a 2 day crossover experimental design. Mice were approximately 4 months old at the time of the experiment. Mice received injections 500nl of AAV9.hsyn.GCaMP6m virus into the ventro-medial striatum followed by implantation of a 0.5x6mm GRIN lens. After 4 weeks of recovery and expression time, mice were implanted with a microscope baseplate. The mice were subsequently habituated to the open field chambers and microscopes for 3 days prior to the experiment. They randomly received injections of either 10 m/kg SKF38393 or saline on day 1. 48 hours later, the injections were reversed on Day 2. Calcium data was acquired with an Inscopix nVista HD microscope with an acquisition rate of 20Hz. Mice were recorded in the open field chamber for 10 minutes prior to injection and 30 minutes post injection.

From the raw data, we apply constrained nonnegative matrix factorization for microendoscopic data (CNMF-E) (Zhou et al. (2016)) that helps extract clear neuron boundaries and give us exact coordinates of the points on the boundaries for each neuron as our input.

Below figures show how our raw data looks like and how does CNMFE get rid of noisy background and then figure out neuron boundaries. The green square in raw data shows one neuron example which is very unclear. Besides, we use simulation data. We generate several datasets, in which there displays a certain amount of neuron shapes.

3 Proposed Approach

Our algorithm has two stages and here we describe them at a high level. First, we consider matching neurons by relative distance relations. Based on relative distances between neurons, we map neurons in the second cloud to the first cloud in a rough way. When the noise level is not large, the neuron in the second session should be mapped to a position that is close to its corresponding neuron in the first session. Next, we refine our matching using the distances between neuron centers and the shape similarities to decide whether to match two neurons or not. The two stages can be summarized below:

Name	Description	Domain
n_1, n_2	number of neurons in the two neuron clouds	$\mathbb{N}_+$
δ_{det}	threshold for the determinant of regression matrix	$\mathbb{R}_+$
δ_{dist}	threshold for neuron center distances	$\mathbb{R}_+$
δ_{shape}	threshold for neuron shape dissimilarity score	$\mathbb{R}_+$
w_{dist}, w_{shape}	weights for distance and shape dissimilarity score	$\mathbb{R}_+$
d_θ	angle for cutting the space into sectors	$\mathbb{R}_+$
N_R	# regions within each cut sector	$\mathbb{N}_+$
$\{C_i^{(k)}\}_{i=1}^{n_k}, k = 1, 2$	coordinates of neuron centers for two neuron clouds	$\mathbb{R}^{2\times n_k}$
$\{\widetilde{C^{(2)}}\}_{i=1}^{n_2}$	coordinates of neuron centers after mapping (2^{nd} cloud)	$\mathbb{R}^{2\times n_2}$
m_{ki}	# points on neuron contour (k^{th} cloud, j^{th} neuron)	$\mathbb{N}_+$
$\{Coor_i^{(k)}\}_{i=1}^{n_1}, k = 1, 2$	coordinates of points on neuron contours	$\mathbb{R}^{2\times m_{2i}\times n_k}$
$\{\widetilde{Coor^{(2)}}\}_{i=1}^{n_2}$	coordinates of points on neuron contours after mapping (2^{nd})	$\mathbb{R}^{2\times m_{ki}\times n_k}$
MP	matching pairs (pairs listed by row)	$\mathbb{N}_+^{2\times p}$
DDS	distance dissimilarity score	$\mathbb{R}_+$
SDS	shape dissimilarity score	$\mathbb{R}_+$
DS	total dissimilarity score	$\mathbb{R}_+$
SS	total shape similarity score	$\mathbb{R}_+$

Table 1: Variables used in the neuron alignment model and algorithm; $\mathbb{R}$: real numbers; $\mathbb{R}_+$: positive real numbers; $\mathbb{N}_+$: positive integers.

- Rough matching based on neuron center relative distances using shape context and linear regression, computing the coordinates of neuron centers $\widetilde{C^{(2)}}$ and neuron points on the contours $\widetilde{Coor^{(2)}}$ using Algorithm 2.
- Final matching according to distance and shape relations. Get matching pair results MP and corresponding dissimilarity score DS using Algorithm 3.

3.1 Part 1: Rough Matching

Our method is based on the algorithm proposed by Belongie et al. (2006) where they used shape context to encode shapes that allows for measuring shape similarity and the recovering of point correspondences. We show how shape context technique works in our problem by introducing how we compute the shape context and convert it to feature vector for each point. Algorithm 2 lists the pseudocodes.

1. **Get the coordinates of neuron centers**
   We use simple notations for neuron cloud here. Extract coordinates of all neuron centers from neuron clouds of two sessions denoted by:
   Neuron cloud 1:
   $$(x_1^{(1)}, y_1^{(1)}), \cdots, (x_n^{(1)}, y_n^{(1)})$$
   Neuron cloud 2:
   $$(x_1^{(2)}, y_1^{(2)}), \cdots, (x_m^{(2)}, y_m^{(2)})$$
   Here, neuron cloud 1 contains n neurons and neuron cloud 2 contains m neurons.

2. **Create feature vector for each neuron center**
   For simplicity, we explain how to compute feature vector for $(x_1^{(1)}, y_1^{(1)})$, all other points

can build their own feature vectors in the same way. Suppose $(x_1^{(1)}, y_1^{(1)})$ is the coordinate of neuron center with index 1 in Figure 2 and all other neurons are from the same time session, which means they show up simultaneously in one session.

We start to construct feature vector for $(x_1^{(1)}, y_1^{(1)})$. Firstly, we construct a local coordinate system for $(x_1^{(1)}, y_1^{(1)})$ by setting $(x_1^{(1)}, y_1^{(1)})$ as the original point, finding the point (x_0, y_0) which is the closest to $(x_1^{(1)}, y_1^{(1)})$, connecting $(x_1^{(1)}, y_1^{(1)})$ and (x_0, y_0) then setting the line as the x-axis with the the direction from $(x_1^{(1)}, y_1^{(1)})$ to (x_0, y_0) as the positive direction, setting a line pass through the original point and perpendicular to the x-axis anti-clockwisely as y-axis. Since we want the feature vector to be more precise, secondly we continue to cut the space from 4 parts to a larger number of parts. In Figure 2, we cut the whole space into 8 sectors by setting the angle to partite the space to be $d_\theta = \frac{\pi}{4}$. In order to make feature vector more precise, we also partite the radius (the radius is determined by the largest distance between $(x_1^{(1)}, y_1^{(1)})$ and another neuron center), in Figure 2, we cut the radius by 2 ($N_R = 2$). Feature vector is created anti-clockwisely and from inner to outer, the values in the vector are number of neurons falling in the corresponding regions. If certain neurons are exactly positioned on region's boundary, we can put them in either side of the sectors as long as they are consistent. In Figure 2, the feature vector for neuron 1 should be:

$$F_1^{(1)} = (0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 2, 0, 0, 0, 0, 0)$$

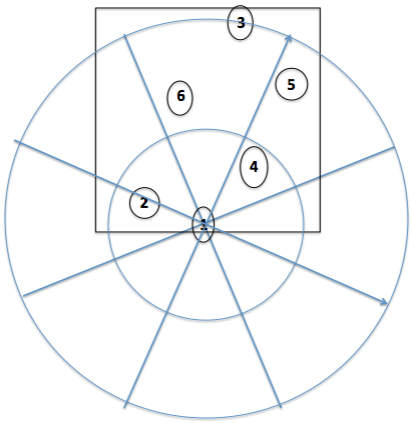


Figure 2: This example shows the feature net constructed for neuron 1, which is centered on neuron 1 and contains several regions based on cutting angles and radius. Here, angle is $\pi/4$ and radius is cut to 2 parts.

3. **Combine all feature vectors by row into a feature matrix**

$$F^{(1)} = \begin{bmatrix} F_1^{(1)} \\ \vdots \\ F_n^{(1)} \end{bmatrix} \text{ and } F^{(2)} = \begin{bmatrix} F_1^{(2)} \\ \vdots \\ F_m^{(2)} \end{bmatrix}$$

Algorithm 1 lists the pseudocodes.

4. **Get matching pairs**

Algorithm 1 Computing the Feature Matrix (FM)

Input:
Parameters: d_θ, N_R.
Data: $\{C_i\}_{i=1}^{n}$.
Output: Feature matrix F.

for $i = 1, \cdots, n$ **do**
  $j \leftarrow$ index of the closest neuron with i^{th} neuron considering the distances between neuron centers.
  Construct a local coordinate system for C_i. Set C_i as original point, connect C_i and C_j as x-axis and set y-axis perpendicular to x-axis. Cut the coordinate system space to $2\pi/d_\theta$ sectors centered around C_i and cut the radius to N_R, form $2\pi N_R/d_\theta$ regions.
  $\{B_j\}_{j=1}^{2\pi N_R/d_\theta} \leftarrow$ cut regions for C_i
  for $j = 1, \cdots, 2\pi N_R/d_\theta$ **do**
    $F_{ij} \leftarrow \#\{$neuron center points$\}$ fall into B_j
  end for
end for
$F \leftarrow [F_1; \cdots; F_n]$

After getting two corresponding feature matrices $F^{(1)}$ and $F^{(2)}$ for the two neuron clouds, this step is to figure out mapping function. We first normalize the feature matrices by row. If two normalized feature vectors are similar, then their inner product should be close to 1, and otherwise close to 0.

We would like our mapping to be more accurate, so we first find out neuron pairs that are most likely to be matched by an optimization problem.

$$\begin{aligned} \max_{C} & \sum_{i,j} F_{ij} C_{ij} \mathbb{1}_{\{F_{ij} \geq \tau\}} \\ s.t. & \sum_{i} C_{ij} \leq 1 \ for \ j = 1, \cdots, J \\ & \sum_{j} C_{ij} \leq 1 \ for \ i = 1, \cdots, I \\ & C_{ij} \in \{0, 1\} \end{aligned} \tag{1}$$

where $F = F^{(1)} F^{(2)T}$. $C_{ij} = 1$ indicates neuron i in cloud 1 and neuron j in cloud 2 should be matched, otherwise not.

5. **Further Refinement by Linear Regression**
   We further refine and test our matching using a linear regression technique. We assume linear transformation for neuron clouds across sessions. Apply linear regression and solve it by least squared:

$$\begin{bmatrix} x^{(1)} \\ y^{(1)} \end{bmatrix} = A \begin{bmatrix} x^{(2)} \\ y^{(2)} \end{bmatrix} + b \tag{2}$$

   based on reliable matching pairs from the solution of the above optimization problem and get mapping function from regression. If $|det(A) - 1| > \delta_{det}$, we conclude the datasets are too noise for our algorithm to align, otherwise, we apply this linear transformation mapping to all neurons in the second cloud.

Algorithm 2 lists the pseudocodes for all steps of rough mapping.

Algorithm 2 Roughly Mapping Based on Neuron Centers Distances between Two Neuron Point Clouds (Rmap)

Input:
Parameters: $d_\theta, N_R, \delta_{det}$.
Data: $n_1, n_2, \{C_i^{(1)}\}_{i=1}^{n_1}, \{C_i^{(2)}\}_{i=1}^{n_2}, \{Coor^{(1)}\}_{i=1}^{n_1}, \{Coor^{(2)}\}_{i=1}^{n_2}$.
Output: $\{\widetilde{C^{(2)}}\}_{i=1}^{n_2}$, $\{\widetilde{Coor^{(2)}}\}_{i=1}^{n_2}$.
$F^{(1)} \leftarrow FM(C^{(1)}, d_\theta, N_R)$ Using Algo 1.
$F^{(2)} \leftarrow FM(C^{(2)}, d_\theta, N_R)$ Using Algo 1.
$F \leftarrow F^{(1)}F^{(2)T}$
Solve the linear assignment problem referring to Equation 1 with coefficient matrix F as input, Let M denotes the result: $0-1$ matrix that represents all matching pairs.
$I_1 \leftarrow \{i : M_{ij} = 1\}$
$I_2 \leftarrow \{j : M_{ij} = 1\}$
Solve the linear regression problem using least squared method: $C_{I_1}^{(1)} = AC_{I_2}^{(2)} + b$
if $|det(A) - 1| > \delta_{det}$ **then**
    return A is not close to orthogonal matrix, this case cannot be solved!
end if
for $i = 1, \cdots, n_2$ **do**
    $\widetilde{C_i^{(2)}} \leftarrow AC_i^{(2)} + b$
    $\widetilde{Coor_i^{(2)}} \leftarrow ACoor_i^{(2)} + b$
end for

3.2 Part 2: Matching neurons based on relative distances and shape similarities

In this part, we decide final matching pairs based on neuron center relative distances relationship and neuron shape similarities. Algorithm 3 lists the pseudocodes.

1. **Compute distance dissimilarity score:** DDS
   Calculate all pair of distances between neuron centers of neuron cloud 1 and neuron cloud 2 after rough mapping in last step, denote it by DDS.

2. **Compute shape dissimilarity score and total dissimilarity score:** SDS, DS
   We firstly translate two neuron contours by putting the centers of them on (0,0) and then calculate the distances between points on two boundaries as shown in the below figure. We made θ grids dense enough to make sure the mean of such distances can give a reasonable dissimilarity score for the two given shapes and define the mean as shape dissimilarity score. The final dissimilarity score contain two part:

   $$DS = w_{shape}SDS + w_{dist}DDS$$

   w_shape and w_dist are weights for SDS and DDS when computing DS that sum to 1.

3. **Construct adjacency matrix based on the neuron center distances**
   By setting up a threshold δ_{dist}, if neuron pairs are closer than δ_{dist}, the corresponding entry in adjacency matrix should be 1 and otherwise be 0. Next, we get connected groups from the bipartite adjacency matrix, called by clusters.

4. **Locally maximize overall similarity scores within each cluster**
   Instead of matching neurons with highest shape similarity score, we locally optimize the matching results within each cluster, which means we want the similarity score to be

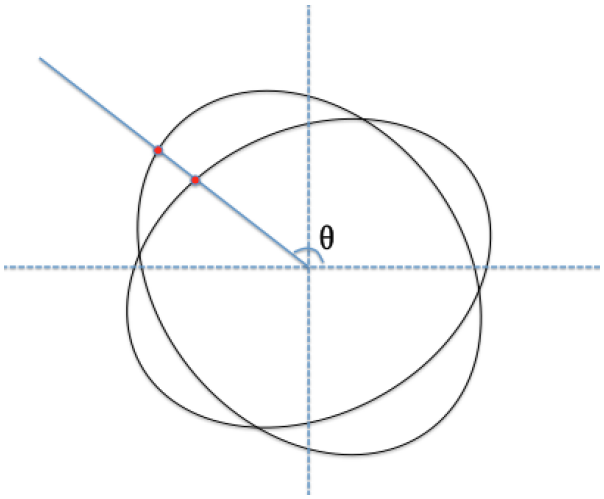


Figure 3: Method of computing shape dissimilarity score: We uniformly choose 24 angles from 0 to 2π as θ grids and average the distances between the boundaries corresponding to the angles (distance between two red dots).

maximized within clusters. The linear assignment problem to be solved for this is shown below:

$$\begin{aligned} \max_{C} & \sum_{i,j} SS_{ij} C_{ij} \mathbb{1}_{\{DS_{ij} \geq \delta_{shape}\}} \\ s.t. & \sum_{i} C_{ij} \leq 1 \ for \ j = 1, \cdots, J \\ & \sum_{j} C_{ij} \leq 1 \ for \ i = 1, \cdots, I \\ & C_{ij} \in \{0, 1\} \end{aligned} \tag{3}$$

where SS is the similarity score matrix, $SS_{ij} = -DS_{ij} + \max_{ij}(DS_{ij})$. $C_{ij} = 1$ indicates final matching result.

4 Simulation

Neuron alignment problem is hard since there are various kinds of noise happen to neurons in different time sessions. We summarize them into three categories and we adjust for them to see results in the simulation.

Firstly, neurons fire in one day may not fire in another day. Also, the distance between camera and the brain can cause problems. For example, one neuron locates on the boundary of one day's images, if the camera are set closer to the brain on the second day, the neuron near the boundary may not be included in the second day's images due to range issues. These two reasons can cause neuron missing. In our simulation, we test this noise type by dropping out some neurons from the original neuron cloud and see how does our algorithm perform.

Secondly, neurons have slightly natural movement. It means neurons' relative positions can change up to a small degree. We test this noise type by adding Gaussian noise to neuron

Algorithm 3 Final Matching considering distance and shape relations(SM)

Input:
Parameters: $\delta_{dist}, \delta_{shape}, w_{dist}, w_{shape}$
Data: $n_1, n_2, \{Coor_i^{(1)}\}_{i=1}^{n_1}, \{\widetilde{Coor_i^{(2)}}\}_{i=1}^{n_2}, \{C_i^{(1)}\}_{i=1}^{n_1}, \{C_i^{(2)}\}_{i=1}^{n_2}$
Output: MP,DS.

$Adj \leftarrow$ adjacency matrix for the bipartite graph
(V: $\{C_i^{(1)}\}_{i=1}^{n_1}, \{C_i^{(2)}\}_{i=1}^{n_2}$; E: $\{(i,j) : norm(C_i^{(1)} - C_j^{(2)}) < \delta_{dist}\}$)
$\{CC\}_{k=1}^{K} \leftarrow$ connected components of Adj (CC_k: indices of neurons within the k^{th} connected component).
for $k = 1, \cdots, K$ **do**
  $I_1 \leftarrow \{i_1 : i_1 \in CC_k, 1^{st}\ cloud\}$
  $I_2 \leftarrow \{i_1 : i_1 \in CC_k, 2^{nd}\ cloud\}$
  for $i_1 \in I_1$, $i_2 \in I_2$ **do**
    if $norm(C_{i_1}^{(1)} - \widetilde{C_{i_2}^{(2)}}) < \delta_{dist}$ **then**
      $DDS_{i_1,i_2} \leftarrow norm(C_{i_1}^{(1)} - \widetilde{C_{i_2}^{(2)}})$
    else
      $DDS_{i_1,i_2} \leftarrow Inf$
    end if
    $SDS_{i_1,i_2} \leftarrow$ shape dissimilarity score, referring to Sec 3.2.
    $DS_{i_1,i_2} \leftarrow w_{dist} DDS_{i_1,i_2} + w_{shape} SDS_{i_1,i_2}$
  end for
  $SS_{i_1 \in I_1, i_2 \in I_2} \leftarrow -DS_{i_1 \in I_1, i_2 \in I_2} + \mathbb{1} \cdot max(DS_{i_1 \in I_1, i_2 \in I_2})$ for non-Inf DS entries
  $SS_{i_1 \in I_1, i_2 \in I_2} \leftarrow 0$ for Inf DS entries.
  Solve the linear assignment problem referring to Equation 3 to get matching pairs MP_k within the cluster CC_k using coefficient matrix $SS_{i_1 \in I_1, i_2 \in I_2}$.
end for
$MP \leftarrow [MP_1; \cdots; MP_K]$

positions of the original neuron cloud and then do the whole matching process.

Finally, the algorithms that we use to extract neuron contours from the blur neuron movement movies can not give us accurate neuron shapes. The core part of extracting contour algorithm is looking for a closed contour that can cover more than 80% of the gray scale of each neuron. Besides, we cannot guarantee that the camera can be placed at the same angle everyday, if the angle changes slightly, we can imagine that a circle may turn to an ellipse so that the shapes are not the same in two days. Also, neurons sometimes rotate up to a small degree around their exact positions. We test shape noise by two ways: add additional rotations slightly, shape elongations.

4.1 Noise Categories.

In summary, by putting together all kinds of noise stuff above, we adjust the following quantities to control several noise levels.

1. **prop_overlap:** proportion of neurons that appear in both of neuron clouds of two sessions, referring to the ratio of #{correct matching pairs} and #{neurons of the larger cloud}. Here we consider the case that neurons show up differently across sessions.

2. **prop_pos:** ratio of the standard deviation of position noise (follows a Gaussian distribution) with the mean of closed neuron distances. To further clarify this, for each neuron, we can find another neuron, whose center has the closest distance with the neuron we are focusing on compared with any other neurons. For each neuron, we record such kind

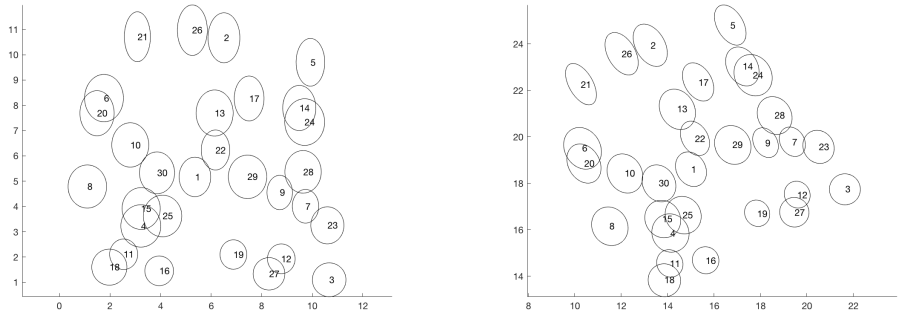
of a distance. By taking the mean of all these distances, we finally get a mean distance called "the mean of closed neuron distances". For the position noise, we sample from a Gaussian distribution $N(0, \sigma_p^2)$. So "prop_pos" is the ratio of σ_p and "the mean of closed neuron distances".

3. $\mu_{\mathbf{rot}}$: mean of the rotation angle noise of neurons. We simulate the rotation angle noise due to camera positions. For each neuron, we sample a rotation angle from Gaussian distribution $N(\mu_{rot}, (\pi/24)^2)$, positive angle refers to clockwisely rotation.

4. $\mu_{\mathbf{elong}}$: elongation noise of neuron shapes. We simulate neuron shape noise (elongations) caused by inclination of camera. In the simulation, we remain the width of each neuron and elongate the neuron shape along the y-axis by el_{coef} times, where el_{coef} is sampled from the Gaussian distribution $N(\mu_{elong}, 0.1^2)$.

4.2 Simulation Procedure and Results

We generate two neuron clouds (day 1 & day 2) including 30 neurons respectively in noiseless case and adjust for all noise types indicated in last section. In the simulation, all matching accuracies are based on tens of simulations for each single noise case, we take the mean as our final accuracy results.
Figure 4 shows 2 simulation neuron datasets in noiseless case. It shows that neurons are randomly displaying within a space range and appears slightly neuron overlapping within sessions, which is similar to real case.



(a) day 1 (b) day 2

Figure 4: Simulation: noiseless case

4.2.1 Noise cases and levels

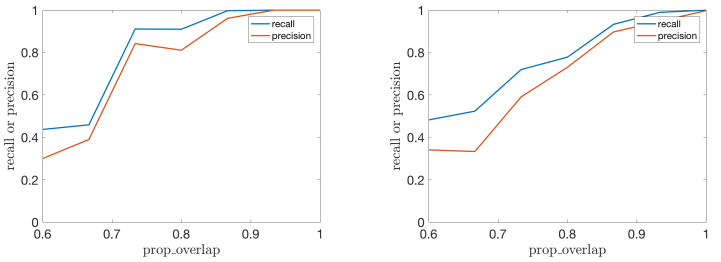
Figure 12, Figure 13, Figure 14 and Figure 15 show how we choose the range of parameters in the simulated data.

4.2.2 Adjusting for Four Noise Types

Next, we adjust for different types of noise levels to see how our algorithm performs. We plot the precision and recall with different parameters for noise in Figure 5, Figure 6, Figure 7 and Figure 8.

δ_{dist}	δ_{det}	δ_{shape}	w_{shape}	w_{dist}
1	0.2	0.25	0.5	0.5

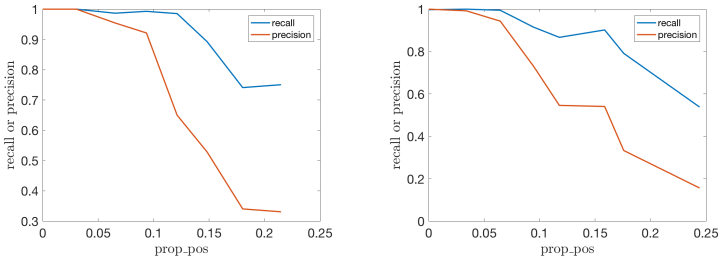
Table 2: Simulation: parameters setting for Figure 5 and Figure 6



(a) **Fixed variables:** prop_pos=0, $\mu_{rot} = 0$, $\mu_{elong} = 1$

(b) **Fixed variables:** prop_pos=0.2, $\mu_{rot} = \pi/12$, $\mu_{elong} = 1$

Figure 5: Adjust for prop_overlap, plots for prop_overlap v.s. recall & precision



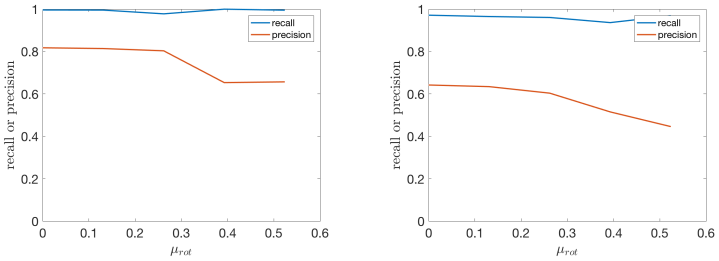
(a) **Fixed variables:** prop_overlap=0.92, $\mu_{rot} = 0$, $\mu_{elong} = 1$

(b) **Fixed variables:** prop_overlap=0.85, $\mu_{rot} = \pi/24$, $\mu_{elong} = 1.1$

Figure 6: Adjust for prop_pos, plots for prop_pos v.s. recall & precision

δ_{dist}	δ_{det}	δ_{shape}	w_{shape}	w_{dist}
0.7	0.2	0.25	0.8	0.2

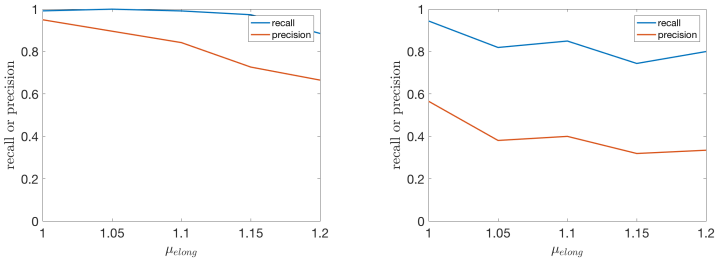
Table 3: Simulation: parameters setting for Figure 7 and Figure 8



(a) **Fixed variables:** prop_overlap=0.92, prop_pos = 0.06, $\mu_{elong} = 1$

(b) **Fixed variables:** prop_overlap=0.85, prop_pos=0.06, $\mu_{elong} = 1.1$

Figure 7: Adjust for μ_{rot}, plots for μ_{rot} v.s. recall & precision



(a) **Fixed variables:** prop_overlap=0.85, prop_pos = 0, $\mu_{rot} = 0$

(b) **Fixed variables:** prop_overlap=0.85, prop_pos=0.12, $\mu_{rot} = 0$

Figure 8: Adjust for μ_{elong}, plots for μ_{elong} v.s. recall & precision

5 Real Data - 10 mice across two sessions

For each mouse, our dataset records the neuron contours showing up on two sessions (either after injection of SKF or saline vehicle), so we can test our algorithm for these 10 matching problems.

5.1 Results

We visualize the neurons showing up on two days in Appendix B for all mice and summarize the parameters setting in Table 4, precision and recall in Table 5. We set w_{dist} smaller than w_{shape} in order to let shape information help us distinguish close neurons. Based on the spatial patterns of neuron clouds and size of neurons, we set $\delta_{dist} = 8$, $\delta_{shape} = 3$.

δ_{det}	δ_{dist}	δ_{shape}	w_{dist}	w_{shape}
0.3	8	3	0.3	0.7

Table 4: Parameters setting for the 10 mouse neuron alignment problems

mouse ID	#SKF	#Saline	#true matches	precision	recall
45 red	113	89	51	0.1351	0.0980
45 green	140	76	55	0.7500	0.9818
45 purple	63	48	28	0.8125	0.9286
46 red	111	97	77	0.8721	0.9740
46 green	130	69	30	0.5116	0.7333
46 purple	62	100	42	0.7736	0.9740
42 green	125	144	86	0.8617	0.9419
48 red	80	71	57	0.9138	0.9298
42 purple	151	141	76	0.3214	0.3554
48 green	172	116	82	0.7383	0.9634

Table 5: Precision, Recall and other descriptions for 10 mouse neuron alignment problems

5.2 Observations

Table 5 shows our algorithm works pretty well regarding to precision and recall except for mouse "45 red" and "42 purple". In order to check whether the rough mapping part is correct or not, we compare the rotation matrix when we apply our algorithm on all neurons and only on correctly matched neurons to see why in some cases our algorithm fails.
Also, we compute the mean of close neuron centers to see how much is the prop_pos noise. Take 42 green for an example, since our algorithm performs well on it, we can get more accurate noise levels.

$$mean_{close\ centers} = 3.1942$$

$$mean_{position\ noise} = 4.2651$$

Hence, $prop\_pos = 4.2651/3.1942 = 1.3353$. The prop_pos value 1.3353 is larger than what we test in simulation, shows that our algorithm can tolerant much larger position noise level. Since in real data, we have more complex shapes so that we can rely on shape differences to distinguish close neurons.

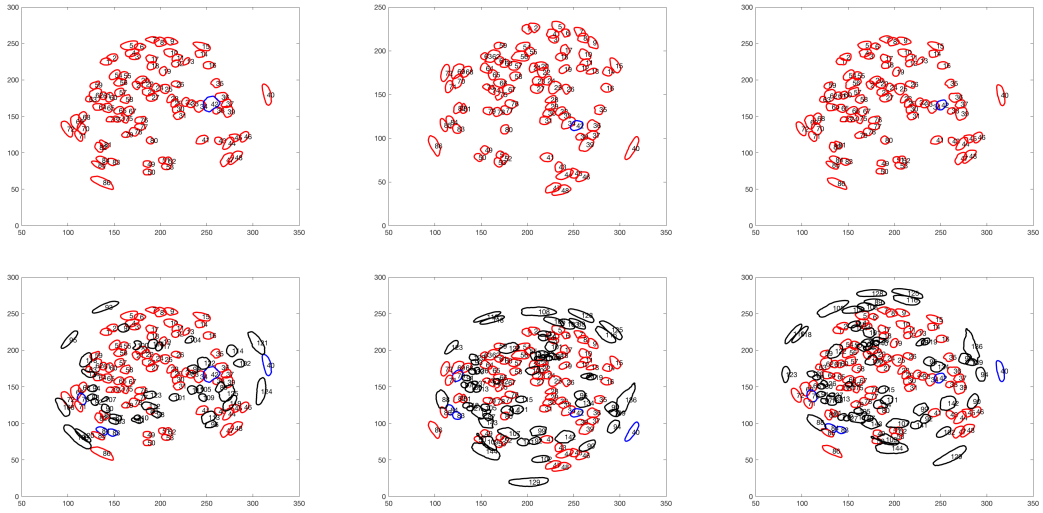


Figure 9: 42 green: **Row 1:** matching based on correctly matched neurons, **Row 2:** matching based on all neurons. **Left:** SKF injection, **Middle:** saline vehicle injection, **Right:** saline after rough map. **Red:** true matches we find, **Blue:** true matches we miss, **Black:** unmatches. The same indexes on neurons represent corresponding true matches. **Results:** determinant of rotation matrix for row 1: 1.0121, rotation: 34.88°; determinant of rotation matrix for row 2: 1.0206, rotation degree: 32.56°. **Conclusion:** the rotation matrices in rough mapping part are very similar and the global structures for the two neuron clouds are very similar, hence it is reasonable for this case to perform good.

6 Conclusion

Neuron alignment across different time sessions is very hard due to severe and blur noise in the brain and noise from the camera and the technique that we use to extract neuron contours from raw neuron activity movies. Based on the results on our simulation data, we can see that under reasonable noise level, our model can recover the neuron matching pairs highly accurately. Further, from the results of real data matching process, we can still identify which

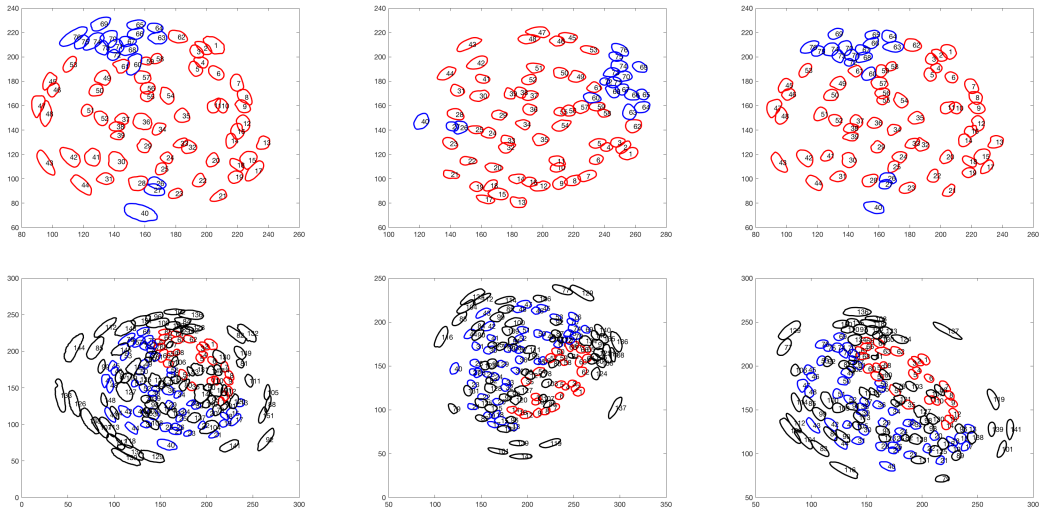


Figure 10: 42 purple: **Row 1:** matching based on correctly matched neurons, **Row 2:** matching based on all neurons. **Left:** SKF injection, **Middle:** saline vehicle injection, **Right:** saline after rough map. **Red:** true matches we find, **Blue:** true matches we miss, **Black:** unmatches. The same indexes on neurons represent corresponding true matches. **Results and Conclusion:** The rotation is roughly correct. But the regions that the camera focuses varies a lot (row 2 left: many unmatches (black) locates outside of the true matches (blue and red), so for SKF injection case, the camera is more far away. This case is hard to handle.

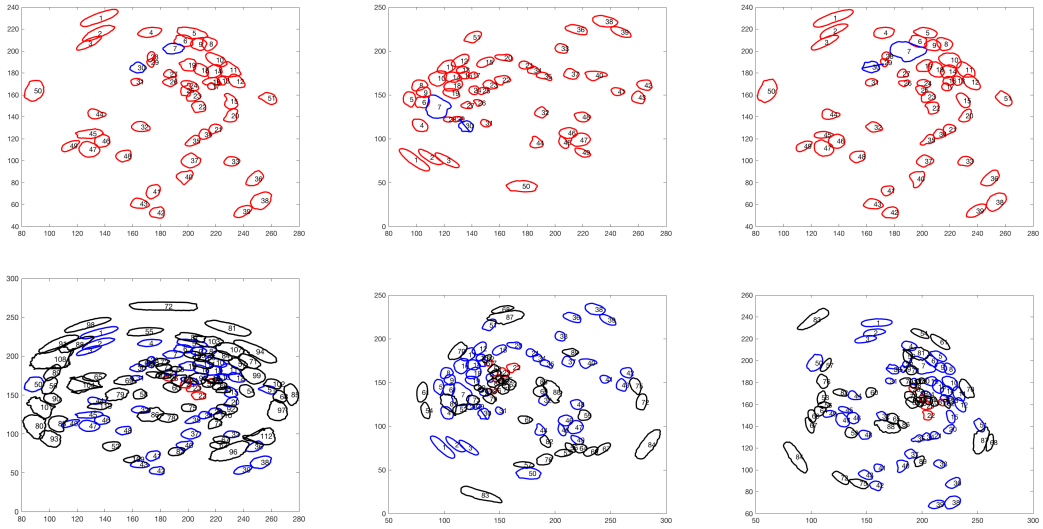


Figure 11: 45 red: **Row 1:** matching based on correctly matched neurons, **Row 2:** matching based on all neurons. **Left:** SKF injection, **Middle:** saline vehicle injection, **Right:** saline after rough map. **Red:** true matches we find, **Blue:** true matches we miss, **Black:** unmatches. The same indexes on neurons represent corresponding true matches. **Results and Conclusion:** determinant of rotation matrix for row 1: 0.9579, rotation: 103.87°; determinant of rotation matrix for row 2: 0.7073, not close to a proper rotation matrix. Also, observing row 2, the regions that the camera focuses varies a lot, camera is more far away from neurons in SKF case. When the scale of regions varies too much, the accuracy is not good.

neurons should be paired up in multiple sessions when the scale of camera focusing regions are similar. Our neuron alignment algorithm successfully speeds up hand aligning process with high accuracy.

Acknowledgement

Thanks to Prof. Rob Kass, Jordan Rodu and Jared Murray for all the instructions. Thanks to James Hyde for providing neuron data and detailed data description. Thanks to Pengcheng Zhou for useful discussions on our project and sharing the CNMF-E code with us.

References

Barbera, G., Liang, B., Zhang, L., Gerfen, C. R., Culurciello, E., Chen, R., Li, Y., and Lin, D.-T. (2016). Spatially compact neural clusters in the dorsal striatum encode locomotion relevant information. *Neuron*, 92(1):202–213.

Belongie, S., Malik, J., and Puzicha, J. (2001). Shape context: A new descriptor for shape matching and object recognition. In *Advances in neural information processing systems*, pages 831–837.

Belongie, S., Mori, G., and Malik, J. (2006). Matching with shape contexts. *Statistics and Analysis of Shapes*, 1:3–1.

Bourgeois, F. and Lassalle, J.-C. (1971). An extension of the munkres algorithm for the assignment problem to rectangular matrices. *Communications of the ACM*, 14(12):802–804.

Burges, C. J. and Schölkopf, B. (1997). Improving the accuracy and speed of support vector machines. In *Advances in neural information processing systems*, pages 375–381.

Klaus, A., Martins, G. J., Paixao, V. B., Zhou, P., Paninski, L., and Costa, R. M. (2017). The spatiotemporal organization of the striatum encodes action space. *Neuron*, 95(5):1171–1180.

Kuhn, H. W. (1955). The hungarian method for the assignment problem. *Naval Research Logistics (NRL)*, 2(1-2):83–97.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

Mukamel, E. A., Nimmerjahn, A., and Schnitzer, M. J. (2009). Automated analysis of cellular signals from large-scale calcium imaging data. *Neuron*, 63(6):747–760.

Murase, H. and Nayar, S. K. (1995). Visual learning and recognition of 3-d objects from appearance. *International journal of computer vision*, 14(1):5–24.

Pnevmatikakis, E. A., Soudry, D., Gao, Y., Machado, T. A., Merel, J., Pfau, D., Reardon, T., Mu, Y., Lacefield, C., Yang, W., et al. (2016). Simultaneous denoising, deconvolution, and demixing of calcium imaging data. *Neuron*, 89(2):285–299.

Turk, M. and Pentland, A. (1991). Eigenfaces for recognition. *Journal of cognitive neuroscience*, 3(1):71–86.

Zhou, P., Resendez, S. L., Stuber, G. D., Kass, R. E., and Paninski, L. (2016). Efficient and accurate extraction of in vivo calcium signals from microendoscopic video data. *arXiv preprint arXiv:1605.07266*.

Appendices

A Figures for determining noise levels

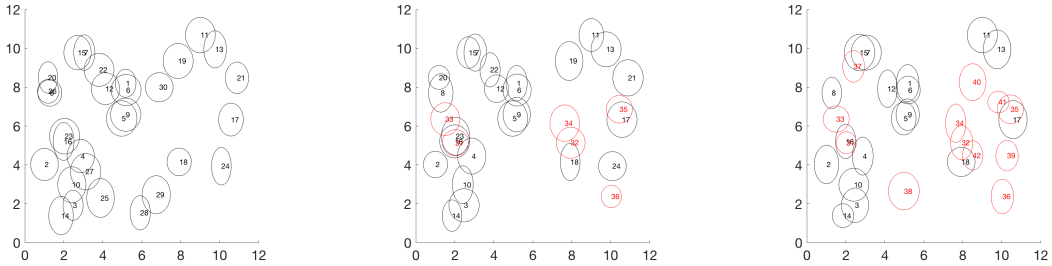


Figure 12: **Black:** Original neurons (some of them may be dropped out), **Red:** new added neurons**From left to right:** $prop\_overlap = 1 \to 0.8 \to 0.6$, no other noise. We see when prop_overlap achieves 0.6, the neuron cloud changes a lot comparing with the noiseless case. So we adjust prop_overlap only down to 0.6.

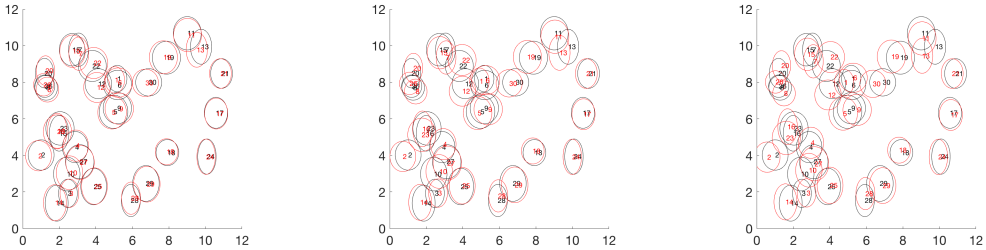


Figure 13: **Black:** noiseless, **Red:** adding different level of prop_pos, **From left to right:** $prop\_pos = 0.1 \to 0.2 \to 0.3$, no other noise. When prop_pos achieves 0.2, the neuron cloud changes a lot comparing with the noiseless case. So we adjust prop_pos only up to 0.2.

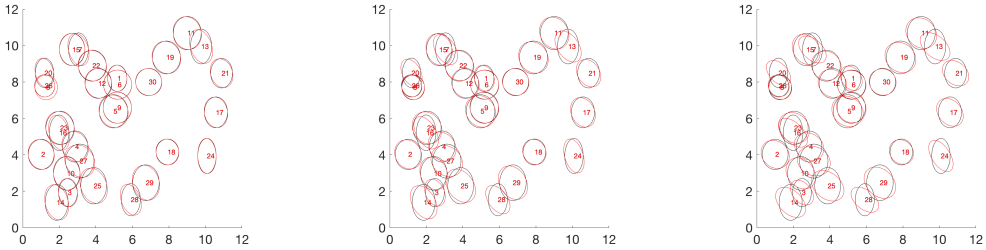


Figure 14: **From left to right:** $\mu_{rot} = \pi/12, \pi/6, \pi/4$, no other noise. When μ_{rot} achieves $\pi/6$, the neuron cloud changes drastically. So we adjust μ_{rot} only up to $\pi/6$.

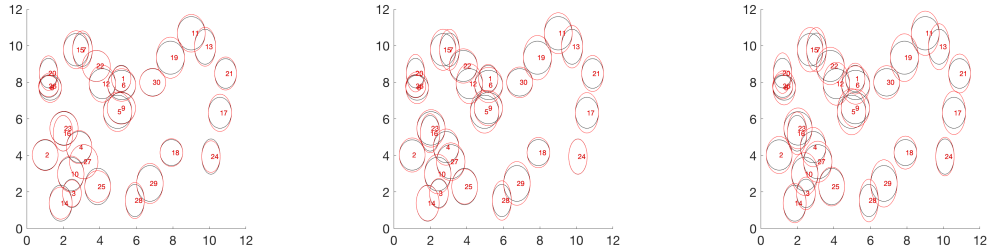


Figure 15: **From left to right:** $\mu_{elong} = 1.1, 1.2, 1.3$, no other noise. We see when μ_{elong} achieves 1.2, the neuron cloud changes a lot comparing with the noiseless case. So in our simulation, we adjust μ_{elong} only down to 1.2.

B Figures of neuron visualization of 11 mice dataset

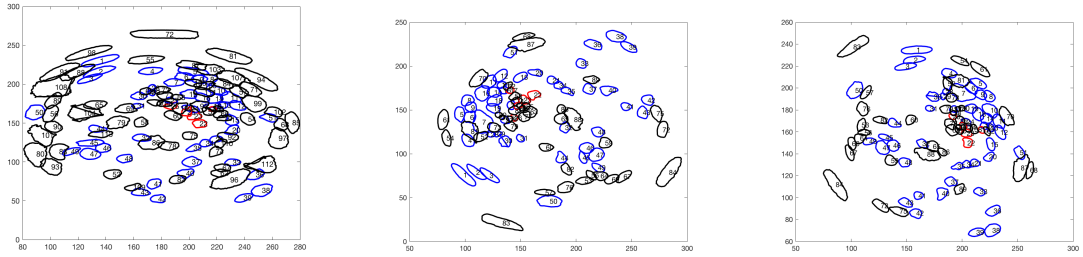


Figure 16: 45 red: **Left:** SKF injection, **Middle:** saline vehicle injection, **Right:** saline after rough map, **Red:** true matches that we find, **Blue:** true matches that we miss, **Black:** unmatches. The same indexes on neurons represent corresponding true matches.

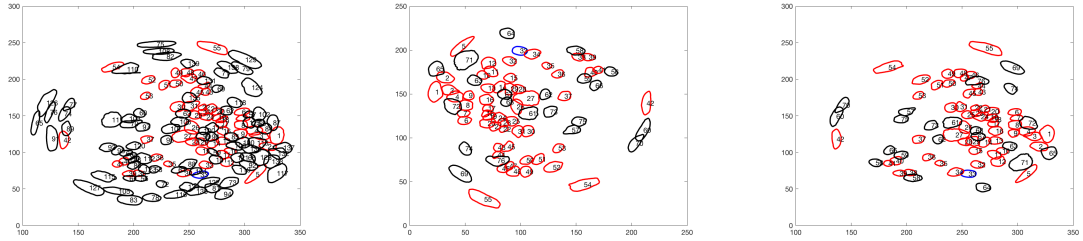


Figure 17: 45 green: **Left:** SKF injection, **Middle:** saline vehicle injection, **Right:** saline after rough map, **Red:** true matches that we find, **Blue:** true matches that we miss, **Black:** unmatches. The same indexes on neurons represent corresponding true matches.

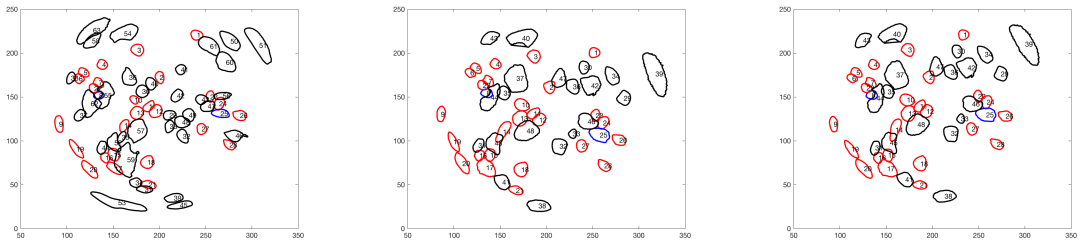


Figure 18: 45 purple: **Left:** SKF injection, **Middle:** saline vehicle injection, **Right:** saline after rough map, **Red:** true matches that we find, **Blue:** true matches that we miss, **Black:** unmatches. The same indexes on neurons represent corresponding true matches.

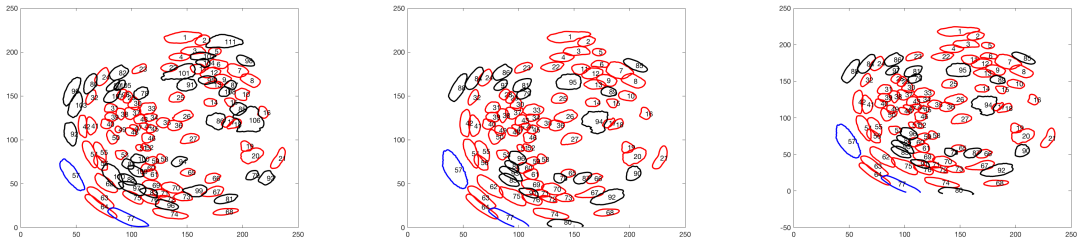


Figure 19: 46 red: **Left:** SKF injection, **Middle:** saline vehicle injection, **Right:** saline after rough map, **Red:** true matches that we find, **Blue:** true matches that we miss, **Black:** unmatches. The same indexes on neurons represent corresponding true matches.

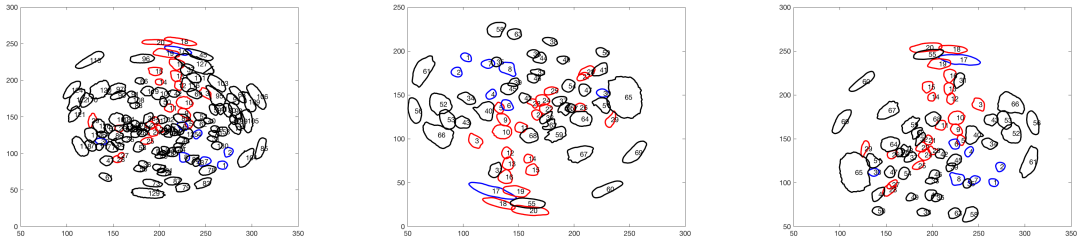


Figure 20: 46 green: **Left:** SKF injection, **Middle:** saline vehicle injection, **Right:** saline after rough map, **Red:** true matches that we find, **Blue:** true matches that we miss, **Black:** unmatches. The same indexes on neurons represent corresponding true matches.

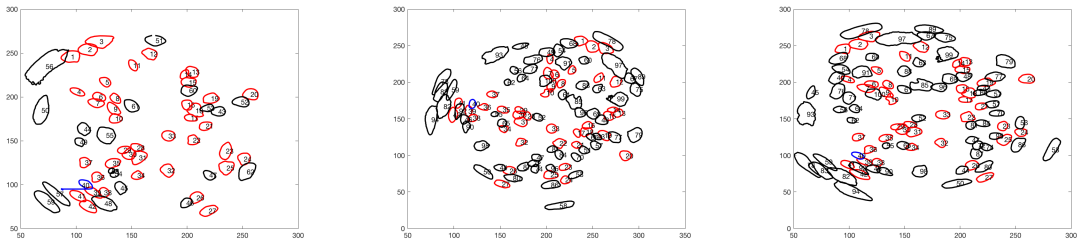


Figure 21: 46 purple: **Left:** SKF injection, **Middle:** saline vehicle injection, **Right:** saline after rough map, **Red:** true matches that we find, **Blue:** true matches that we miss, **Black:** unmatches. The same indexes on neurons represent corresponding true matches.

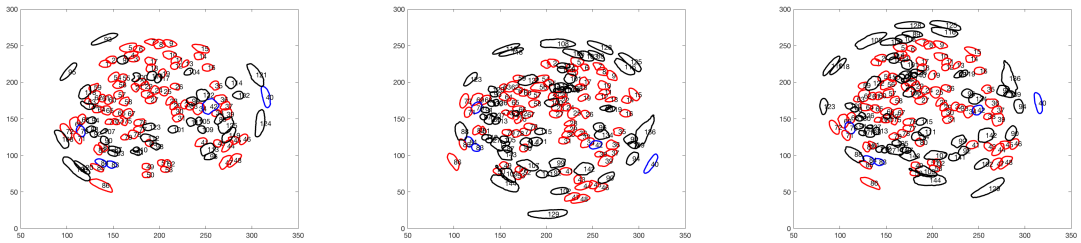


Figure 22: 42 green: **Left:** SKF injection, **Middle:** saline vehicle injection, **Right:** saline after rough map, **Red:** true matches that we find, **Blue:** true matches that we miss, **Black:** unmatches. The same indexes on neurons represent corresponding true matches.

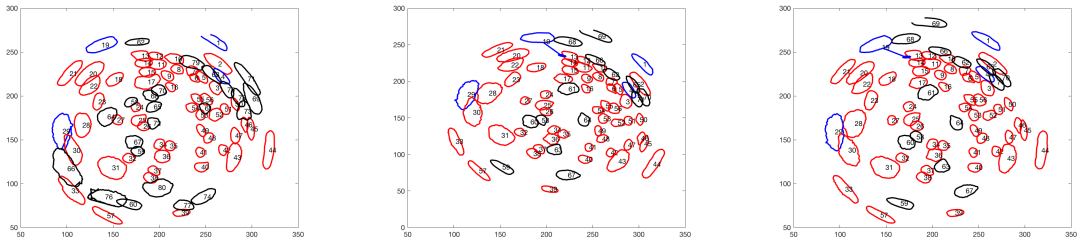


Figure 23: 48 red: **Left:** SKF injection, **Middle:** saline vehicle injection, **Right:** saline after rough map, **Red:** true matches that we find, **Blue:** true matches that we miss, **Black:** unmatches. The same indexes on neurons represent corresponding true matches.

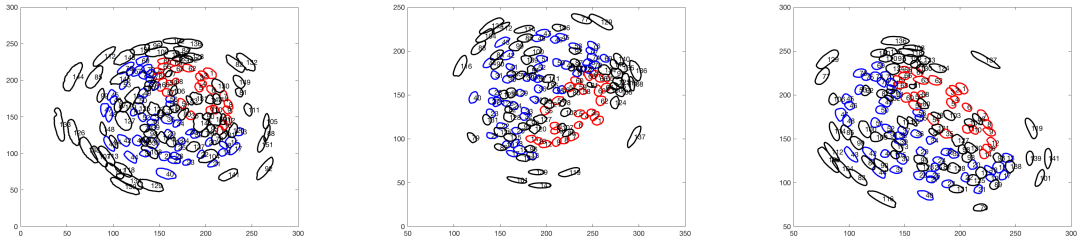


Figure 24: 42 purple: **Left:** SKF injection, **Middle:** saline vehicle injection, **Right:** saline after rough map, **Red:** true matches that we find, **Blue:** true matches that we miss, **Black:** unmatches. The same indexes on neurons represent corresponding true matches.

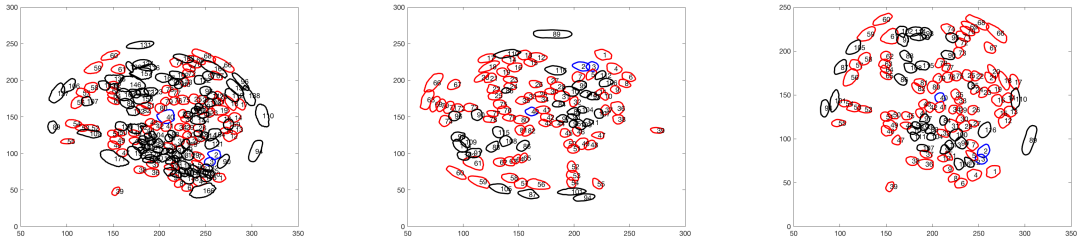


Figure 25: 48 green: **Left:** SKF injection, **Middle:** saline vehicle injection, **Right:** saline after rough map, **Red:** true matches that we find, **Blue:** true matches that we miss, **Black:** unmatches. The same indexes on neurons represent corresponding true matches.

C Comparison of true matches and our matches for 42 purple

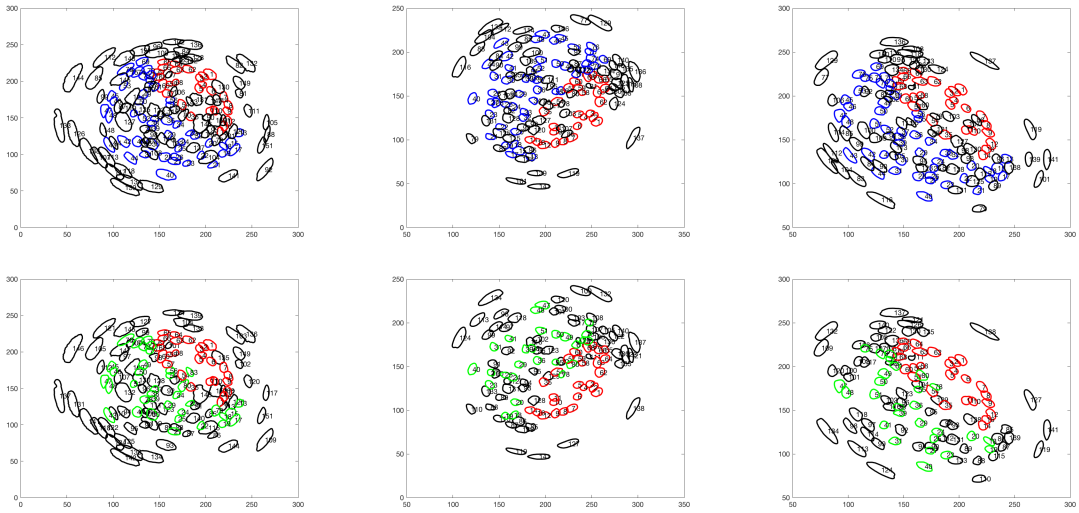


Figure 26: Our algorithm does not perform very well on "42 purple" dataset, so we clearly show neurons of "true matches" that we call or fail to call in row 1, neurons of "matches we call" that are correct or incorrect in row 2. **Row 1:** Red: true matches we find, Blue: true matches we miss, Black: unmatches according to ground truth, Numbers: same numbers corresponding to true matches. **Row 2:** Red: true matches we find, Green: matches we find but are not correct, Black: neurons not in our outputs, Numbers: same numbers corresponding to matches of our outputs (red & green). **From left to right:** SKF injection, saline vehicle injection and saline vehicle injection after rough mapping. **Observations:** We observe the camera focusing regions of SKF and saline vehicle case are different, the camera is more far away from the brain in SKF case compared with saline case, since in row 1, many unmatched neurons (black) locates on the boundary of the image, while for saline vehicle, this does not happen. Our algorithm does not work very when the focusing regions differ, many neurons in left hand side in the images fail to be matched correctly.