LPBS - Laboratory of the Physics of Biological Systems

C.Elegans Neuron Tracking

Semester Project

Oriane Peter

Professor:
Sahand Jamal Rahi

Assistant:
Matthias Minder

Spring 2020 — École Polytechnique Fédérale de Lausanne

1 Introduction

Calcium imaging is used to record neuronal activity and is a fundamental tool when seeking to understand behavior in animals. In Caenorhabditis elegans (C.Elegans), Calcium imaging made whole brain recording possible and allows the study of neural dynamics at cellular resolution.

One of the first steps in the analysis of those recordings is the tracking of the neurons across frames. Even when immobilized, C.Elegans brains undergo consequent motion, including non-rigid deformation, making this tracking non-trival. Tracking used to be done completely manually, as in the paper by Kata et al.2015[1]. Subsequent work improved tracking by manually annotating only a subset of reference frames and using them to classify the rest[2]. However manual annotation is expensive, both in time and resources, and a fully automatic tracking method would be preferable.

Nguyen, Linder, Plummer, Shaevitz and Leifer, 2017 presented an automated approach to segmenting and tracking neurons in calcium imaging of unrestrained C.Elegans[3]. In this method the neurons are first aligned along the principle axis of the worm and then tracked using a point set registration via a Gaussian Mixture Model (GMM).

In this report, I will present an adaptation of this method for the recording performed at the EPFL Laboratory of the Physics of Biological Systems. The first step of the method requires a low magnification dark field image of the full worm to allow straightening of the recording. Lacking this data, I only focused on GMM point set registration, expecting that straightening the worm would not be a necessary step when processing immobilized worm recording. The second step can be done in three parts: Feature Engineering, Classification and Error Correction. In this report, I will go over each of them and then showcase the tracking performance of the method.

2 Methods

A Git repository is available here with the implementation of the method. The method takes as an input the results of the segmentation as performed in the current pipeline. Here is a rough outline of how the method performs:

1. Find a set of features to describe each segment.The features should be similar for the same segment across frames and allow to differentiate a segment from another.
2. Cluster together the segments that have similar features.
3. Make sure that a cluster correspond to one segment tracked across frames. This means making sure that no two segments from the same frame are clustered together.

Each step is detailed bellow and all images are taken from the analysis of the ”20200210_162333_SJR3.1.1_w1_s1_IAA” recording.

2.1 Features Engineering

Nguyen et al.[3] proposed to characterize neurons based on how they match some references. The idea is to define a set of reference frames and, for each segment, build a ”neuron registration vector”. A neuron registration vector is a vector containing the match of a neuron on each reference. Figure 2.1 is the illustration by Nguyen et al.[3] of this registration vector construction.

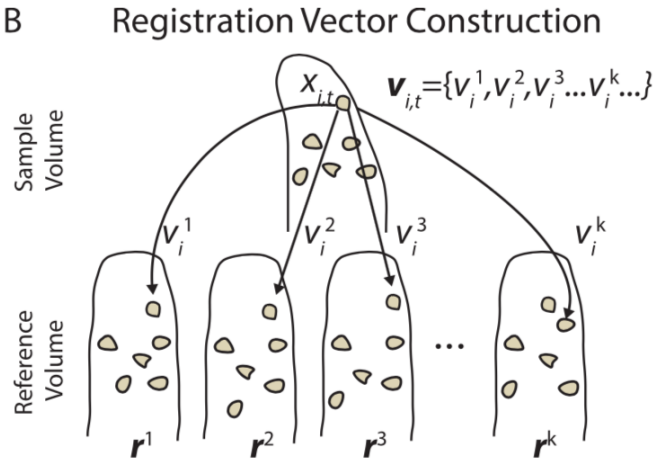

Figure 2.1: Neuron registration vectors are constructed by assigning a feature vector $v_{i,t}$ to each neuron $x_{i,t}$ in a sample volume x_t by performing the registration between the sample volume and a set of 300 reference volumes, each denoted by r^k. Each registration of the neuron results in a neuron match, v_i^k and the set of matches becomes the feature vector $v_{i,t}$.

In the original paper 300 reference volumes are used. In my method, I decided to use only 10 since the results were not improved by adding more references but the time cost rose significantly. However, using less than 10 references decreased the quality of the tracking.

The matching between a segment and a reference is done using the point set algorithm developed by Jian and Vermuri[4]. This algorithm takes as input two 3D point sets: an object and a reference. Its aim is to register the object on the reference by assuming both point sets are Gaussian mixture and trying to deform space by minimizing the distance between the two mixture. The input is the coordinate of the centroids of the segments of the frame to be registered and of the reference frame. The output is the 3D projection of the centroids of the object on the reference. This means that each segment has 30 features: the 3 coordinates of its projection for 10 references.

2.2 Classification

Classification is performed using KMeans on the previously found features. This is diverging from Nguyen and al.[3] They used hierarchical clustering with a distance cutoff of 0.9. This was not performing well on our data and was slower and less robust than KMeans. However, KMeans requires deciding in advance the number of clusters to be found. Since we do not have this information beforehand, the number of clusters was decided to be higher than the maximum number of segments present in any frame. This might produce splits of clusters but can be corrected by hand given an adequate user interface.

2.3 Error Correction

The classification does not assure that segments of the same time frames cannot be clustered together. Such a ”collision” event would lead to two different neurons assumed to be the same. This could be detrimental to subsequent analysis of the recording. To avoid this, when collision occurs, the segments are reassigned based on how close they are to the cluster. The *ith* closest segment to the cluster is always reassigned to the same new cluster.At the end of error correction all clusters are renumbered to avoid uselessly large number.

This method is different from the one used by Nguyen and al.[3] In their paper, they used majority voting on the clusters assignment of the segments in the registration vector of each neuron. This method was more time expensive and led to worse results that the one based on cluster distance.

3 Results

To make sure that each step of the process performs as expected, we can visualize them.

3.1 Feature Engineering

Figure 3.1 shows the result of the features engineering step.

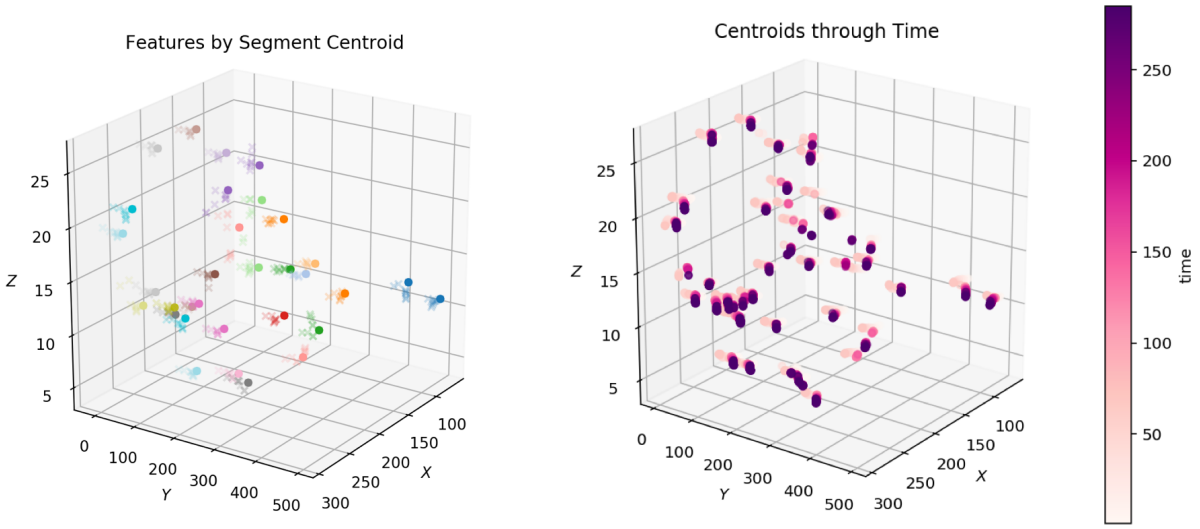

Figure 3.1: Left: Features of each segment of the first frame. The segment centroid is represented by a dot, its associated feature is represented by a cross of the same color. Right: Centroids plotted and colored through time.

We see that the centroids are mapped to coherent references i.e. not to points that are closer to another centroids. We also see that the features correspond to the emplacement of the segment in a later time frames and can therefore assume that this step is performing as expected.

3.2 Classification

Figure 3.2 shows the result of the classification step.

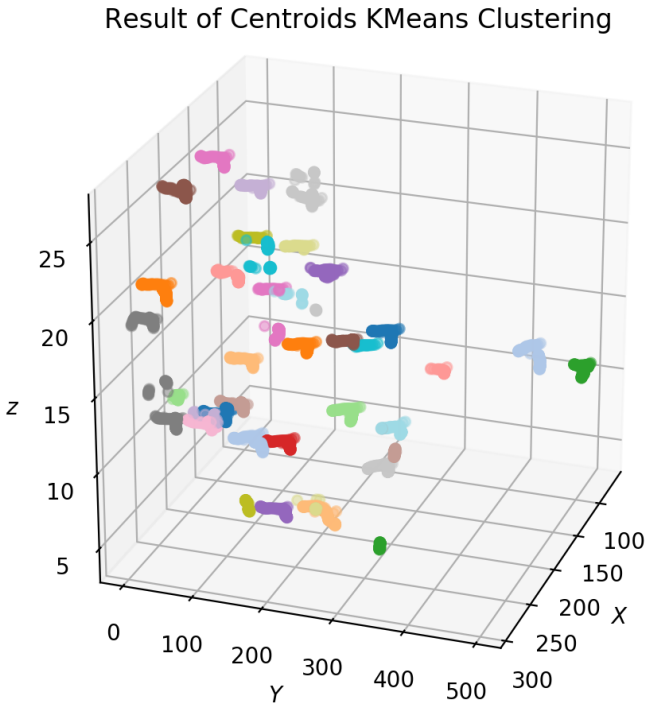

Figure 3.2: Centroids of segments for all time frames. Points that have the same color belong to the same cluster. NB: Sometimes clusters that are far appart have the same color but are in fact two distinct clusters. This is due to the shortage of plotting colors.

Visually, the segments that are clustered together do seem like they are the same but at different time points, indicating that the clustering performed what we expected.

3.3 Error Correction

Figure 3.3 shows the result of the error correction step.

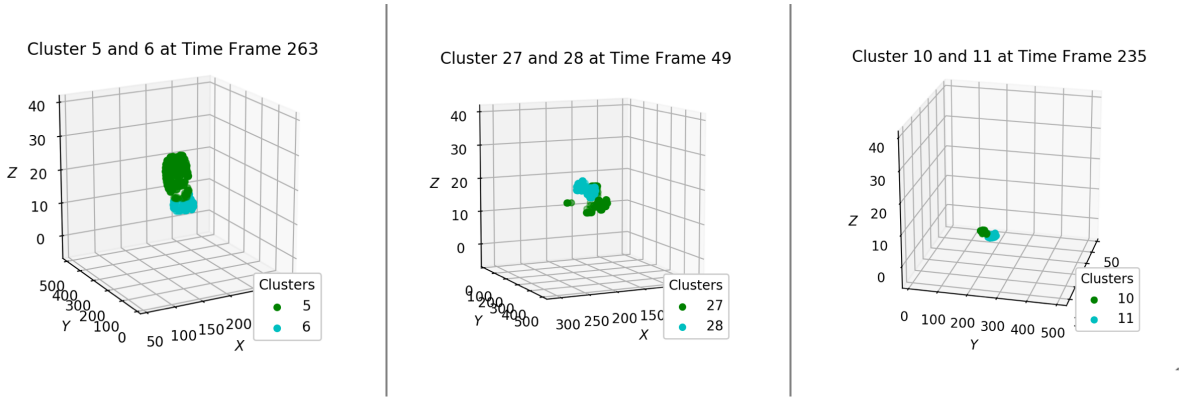

Figure 3.3: Result of error correction on three clusters with collision. Each image shows the contours of segments that were initially clustered together, colors show how the separation was performed.

It is difficult to tell whether the error comes from segmentation, where a single neuron could have been cut into two segment. The error could also be due to motion, causing one of the segment to take the position of the other on one or more reference frames, thus confusing the

classifier. If this is the case, we see that the error correction coherently splits the clusters and resolves this issue.

3.4 Assessing Performance

Since there is no ground truth to compare the results to, assessing performance is somewhat tricky. A first sanity check for the method is to compare our results to a baseline, in this case the tracking method currently used in the pipeline. Figure 3.4 shows the result of this matching.

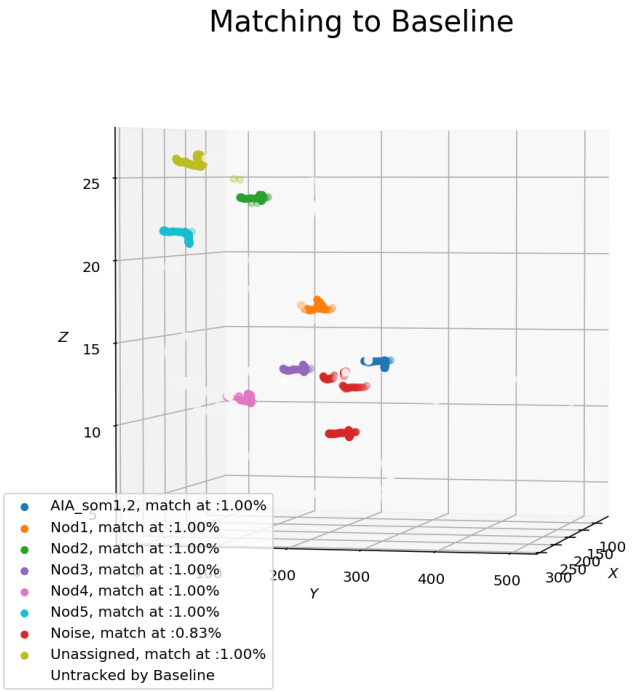

Figure 3.4: Evaluation of the match between baseline and new method. The centroids of segments are plotted across time and colored according to the assignment made by the baseline. Match is calculated as number of point clustered together by the new methods belonging to the same assignment in the baseline / total points in the cluster.

The matching metric is established by looking at the proportion of points in each cluster C_i of the new method that are also clustered together in the baseline assignment A_i. We see here that all relevant clusters are tracked similarly to the baseline. We also see that the newer methods is tracking a lot more segments than the baseline, including some arguably non negligible one. This is more evident if we visualize the tracked neurons on the nd2 files and compare both tracking, as done in Figure 3.5.

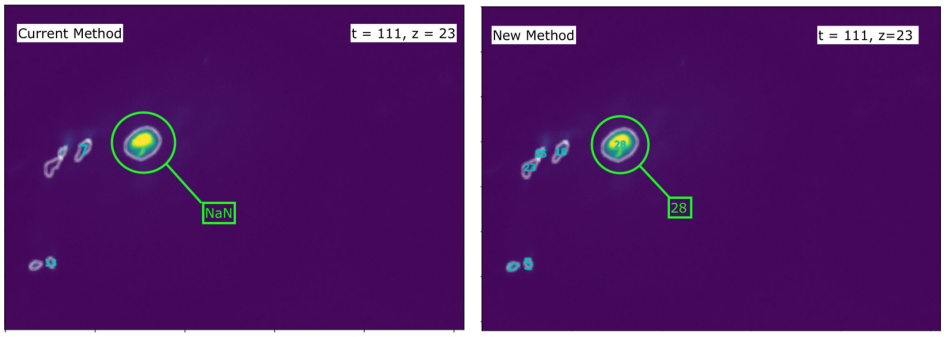

Figure 3.5: Comparison of the tracking performed by the baseline and the new method on same frame at the same level. Cluster assignment is noted in Cyan.

We see that the large bright neuron is not tracked by the current method but is successfully tracked by the new one. Further assessment should be done manually. To show an overview of the tracking, we can visualize the assignment of the method superposed to the nd2 files on Figure 3.6.

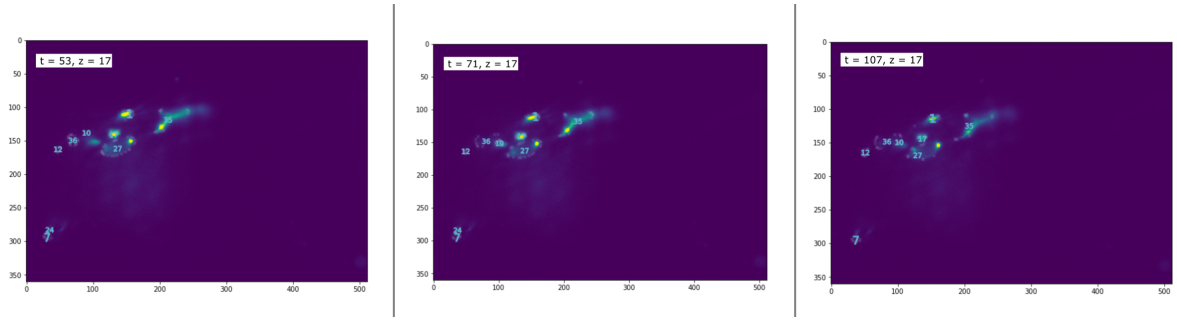

Figure 3.6: Example of tracking on three time frames at the same level, superposed on nd2 file.

We can also look at the result on a recording with higher motion such as recording: ”20190805_SJR3.2.2_w1_s4”, to showcase the robustness of the methods, on figure 3.7.

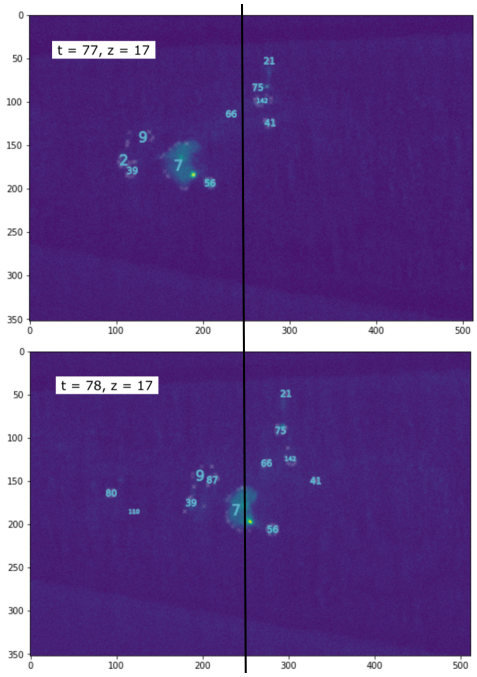

Figure 3.7: Example of tracking on two time frames at the same level in the presence of motion, superposed on nd2 file. The black line is added to help with motion perception.

The git repository contains a notebook that allows to navigate this visualization through time and levels and helps with the performance assessment.

3.5 Issues

The main issue of the method identified so far is the multi-assignments of single neurons, meaning that a single neuron is sometimes assigned to several clusters. Figure 3.8 shows an example of this. This is also from recording "20190805_SJR3.2.2_w1_s4", as more stable recording do not have this artefact.

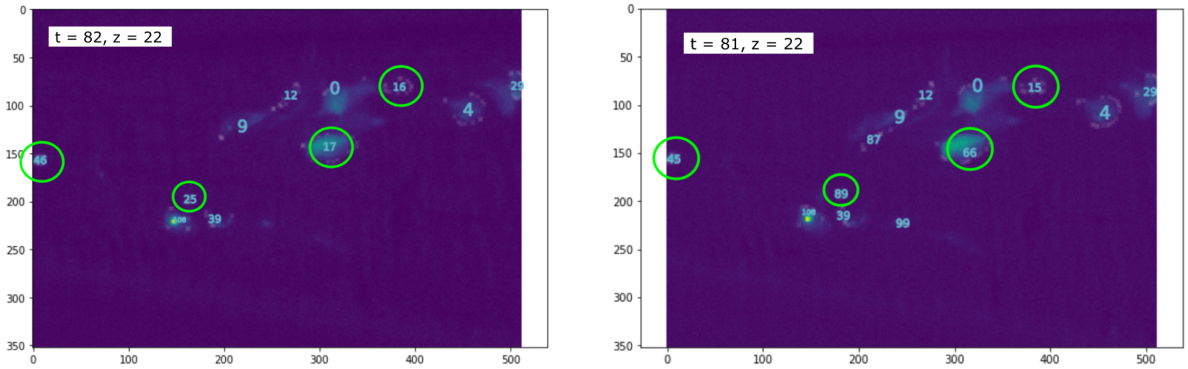

Figure 3.8: Example of multiple clusters being assigned to the same neuron.

3.6 Performance on Freely Moving Worm

The method was run on some recording of a freely moving worm ("20200228_164816_SJR4.2.2_3h_30C_w1_s1_IAA") to see if performance was maintained. A few frames of the results are shown in Figure 3.9.

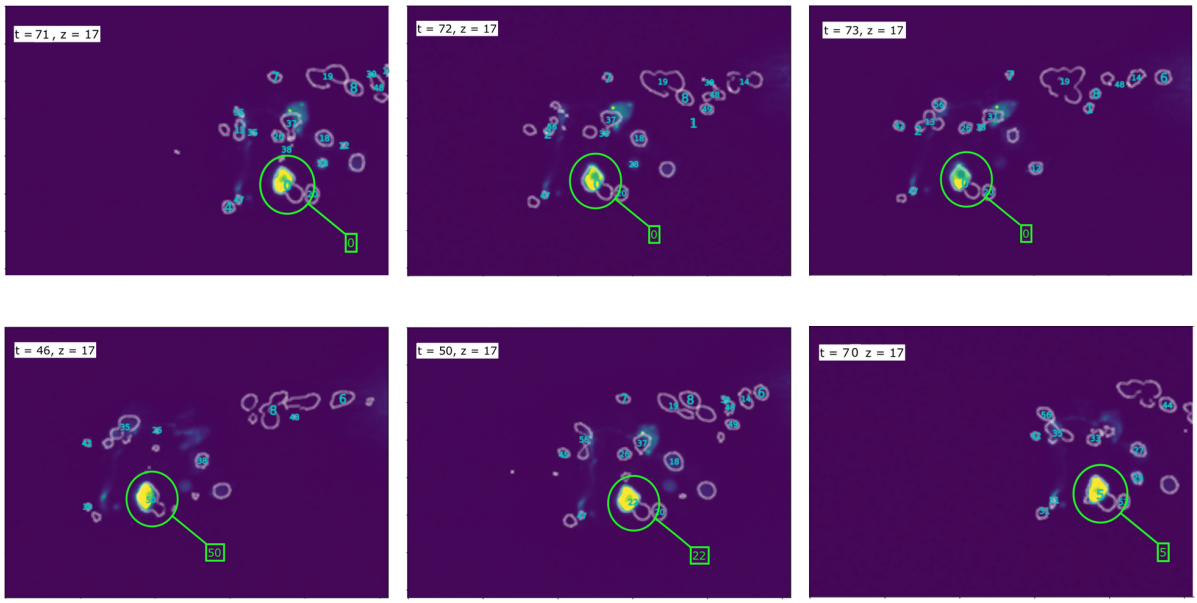

Figure 3.9: Result of method on freely moving worm recording on several frames at the same level.

4 Discussion

Overall the method allows for a robust and accurate tracking on immobilized worms. Neurons tracked by the current method are still tracked with similar performance and many other neurons are also tracked. The gain of time and resources caused by the lack of manual annotation is also a net improvement over the current method. The issue of multi-assignment can be manually solved without too much difficulty with an appropriate user-interface. So far no cluster seem to be assigned to more than one neuron. This issue would be the most harmful to the analysis of the signal and has not appeared thus far.

However, this method cannot be used as such on freely moving recordings. Figure 3.9 shows that the worm is moving too much for the algorithm to perform well. The result still shows some coherence on larger segments such as the segment 0, but this coherence is lost on smaller ones.

To be able to track neurons of freely moving worms, the first step of the method by Ngyuen et al.[3] should be implemented, namely the alignment of the worm along the principle axis. It would require a low magnification dark field image of the full worm meaning an adaptation of the recording setup.

An improvement that can be considered is to remove the assumption of uniform co-variance of the Gaussian mixtures in the point set registration algorithm by Jian and Vermuri[4]. Ngyuen et al.[3] have added the information of co-variance of the segments to increase the accuracy of the registration. Out of simplicity this has not been tried yet and might increase performance.It is reasonable to assume that this method would also work on the green channel, given a segmentation of this channel.

References

[1] Saul Kato, Harris S Kaplan, Tina Schrödel, Susanne Skora, Theodore H Lindsay, Eviatar Yemini, Shawn Lockery, and Manuel Zimmer. Global brain dynamics embed the motor command sequence of caenorhabditis elegans. *Cell*, 163(3):656–669, 2015.

[2] Vivek Venkatachalam, Ni Ji, Xian Wang, Christopher Clark, James Kameron Mitchell, Mason Klein, Christopher J Tabone, Jeremy Florman, Hongfei Ji, Joel Greenwood, et al. Pan-neuronal imaging in roaming caenorhabditis elegans. *Proceedings of the National Academy of Sciences*, 113(8):E1082–E1088, 2016.

[3] Jeffrey P Nguyen, Ashley N Linder, George S Plummer, Joshua W Shaevitz, and Andrew M Leifer. Automatically tracking neurons in a moving and deforming brain. *PLoS computational biology*, 13(5), 2017.

[4] Bing Jian and Baba C Vemuri. Robust point set registration using gaussian mixture models. *IEEE transactions on pattern analysis and machine intelligence*, 33(8):1633–1645, 2010.