Lab Report

Christine Grabosch

March 2020

Contents

1 Introduction

This project was about aligning neuron activity from different trials to identify place cells with higher precision.
Place cells in the Hippocampus are instruments for spatial representation and self-location. Together with entorhinal grid cells, place cells form the basis for representation of places, routes and associated experiences as well as memory. Since both of these cell types are representing internal computations, investigating these cells could improve the understanding of cortical network dynamics [1].
Place cell identification therefore plays a vital role in Hippocampus and learning research. It can be deduced how animals represent the environment and react to reward [1]. Since the recordings are done by injecting electrodes into the region of interest and measuring single neurons in the head-fixed mice, followed by spike sorting, inaccuracies could be present in the data. Another problem is that a mouse does not necessarily run with the same speed in every trial, so the positional data could have a slight deviation. To account for this, Alex Williams et al. developed simple warping methods that are understandable and interpretable while still performing well, compared to other more complex warping methods [2]. In this report I will explain how the warping algorithm works and show results of applying warping on place cell data, as well as evaluating what influences different regularizations have on the algorithm.
The functions that are used to align the spikes could be evaluated with respect to the experiments, e.g. do the knots of a piecewise linear function correspond with stopping of the mouse or could the slope of the function correspond with the speed of the mouse?

2 Methods/ Implementation

In this chapter I will briefly introduce the dataset that was provided to the algorithm, the concept of warping and how it is implemented in the Python library that was used.

2.1 Data

The dataset consists of recordings from a head-fixed mouse running on a wheel which turned a carousel in front of the mouse. Based on this rotation, the mouse was presented with varying inputs placed on the carousel. In the resulting dataset, place cells could be identified.
For every neuron, every trial and every position, the dataset includes the number of spikes per position normalized by time spent in that position. The dataset requires a list of spike-sorted recordings, either with dimensions neurons x positions x trials or as continuous data, and a separate list of neuron IDs. Additionally, lists with more information can be provided, for example the mean firing rate or SSI, to select a subset of neurons for the analysis.

2.2 Warping

The algorithm used in this project is an adaption of time-warping [2], but here it is applied to spatial instead of temporal data.
A widespread approach to analyzing spikes is using over-trial average. If N neurons are measured at P positions over K trials, trial-average is defined as

$$\bar{X} = \frac{1}{K} \sum_{k=1}^{K} X_k$$

with X_k being an NxP matrix, describing the measurements for all neurons in trial k.
Even though it is frequently used, if the data are misaligned across trials, the trial-average can

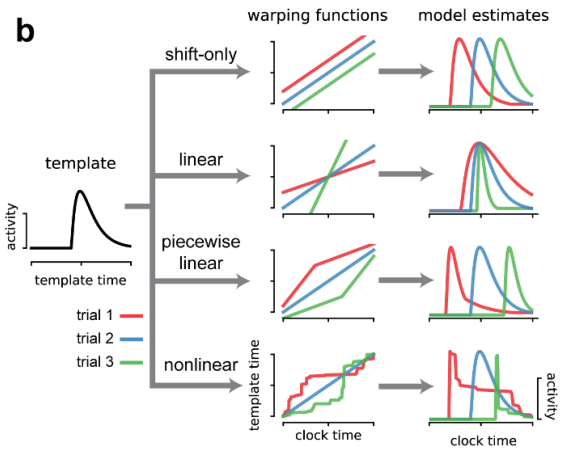

Figure 1: Illustration of different warping functions. From [2].

result in misleading results. To account for this, many approaches were developed to align the data with algorithms or manually. One of these approaches is called warping. It is a data-driven algorithm that uses statistics to align the data.
The algorithm works by fitting a *response template* that is warped, meaning shifted and stretched, for each trial to match the data. This response template $\tilde{X}$ is again an NxP matrix that represents the average activity across the trials after correcting for deformations. The position axis of the template is then transformed for every trial k by a *warping function* $\omega_k(p)$. It maps each spatial bin to a new index. Warping functions need to be monotonically increasing and therefore invertible, so the inverted functions can be used to align the data. The different versions of warping function are illustrated in figure 1.
Shift warping works with warping functions that are constrained to be linear with a slope of 1, so only the intercept is modeled for each trial. Linear warping has intercept as well as slope as free parameters and piecewise-linear warping implements the option of having knots where the slope of the function can change. Nonlinear warping functions apply non-uniform changes to the data. Thus, shift warping accounts for differences in latency, linear warping can additionally account for stretches or compressions and piecewise-linear warping and nonlinear warping add even more complexity.
The warping functions are chosen to minimize the mean squared error given by

$$MSE = \frac{1}{NPK} \sum_{n=1}^{N} \sum_{t=1}^{P} \sum_{k=1}^{K} (\tilde{X}_{n\omega_k(p)} - x_{npk})^2$$

To avoid overfitting, a smoothness regularization term is included to encourage spatially smooth model estimates. In addition to that, a penalty is introduced that limits the warp-magnitude. To identify the optimal model with the best parameters for a given dataset, nested-crossvalidation is implemented.

2.3 Optimization

The template and warping functions are optimized to minimize an objective function F:

$$F(\tilde{X}, \omega_1, \omega_2, ..., \omega_K) = \sum_{k=1}^{K} f_k(\tilde{X}, \omega_k) + \rho_1(\tilde{X})$$

with per-trial loss f_k and regularization term ρ_1, penalizing roughness and size of template. Since least-square loss functions are used, it follows that

$$f_k(\tilde{X}, \omega_k) = ||W_k\tilde{X} - X_k||_F^2 + \rho_2(\omega_k)$$

In this formula, ρ_2 is a regularization term penalizing warping magnitude and $||\cdot||_F^2$ is the Frobenius norm, representing the sum of squared residuals.
To minimize F, block coordinate descent is used, updating the template and the warping functions by subdividing the optimization in subproblems for every variable respectively. This enables efficient computations, since every parameter is independent from all others, only depending on the raw data X_k for trial k and the current warping template $\tilde{X}$.

2.4 Implementation

To implement the previously described method, I developed a Python script to fit the warping functions on experimental data. The warping functions were provided by a Python library found on GitHub (https://github.com/ahwillia/affinewarp). The library includes shift warping and piecewise warping functions that are applied to the datasets. Apart from the data, regularization parameters are passed when fitting the model. For spike data, a data structure is available to store the data in order to work with the warping functions, but for continuous data, the algorithms can be applied without preprocessing.
My implementation includes flexible selection of the model and the respective hyperparameters, as well as the option to select a subset of neurons or a subset of trials to be taken into account when fitting the model. Furthermore, the aligned data are visualized and stored automatically. The program is fully documented and can be called from the command line, providing additional information, however, you can find a user guide with examples in chapter 5.1. The project can be found as an open source repository on GitHub (https://github.com/ChristineGra/position_warping) together with the development history.

3 Results

In this section, I will present and analyze the results of applying the warping functions to the raw data. Overall observations are that even while including neurons without a strong spatial pattern, improvements are observable in the data. However, by eliminating these neurons without spatial patterns, the alignments for the remaining neurons improves. The best improvements were achieved when only taking into account one single neurons for fitting the model.
In the plots, higher activity at a position is displayed by a brighter color (ranging from dark blue to yellow) and the respective firing rate is presented in red. On the left, the raw data and on the right next to it the transformed data are plotted.

3.1 Multiple Neurons

In figures 2 - 5, it can be observed that improvements occur in multiple neurons. However, the amount of neurons that show improvements and also the magnitude of improvements increases with increasing model complexity. For the shift model, changes can be seen especially in neurons 154, 196, 902 and 998. When applying the linear model, also neuron 511 shows improvements, in the piecewise model with 1 knot, neurons 137 and 716 are additionally improving and in the piecewise model with 2 knots, also neuron 934 shows improvements. However, since some neurons do not have strong spatial patterns, no or very little improvement can be observed in those.

3.2 Few Neurons

In contrast to the results for multiple neurons, when only taking a few neurons with strong spatial patterns into account, improvements can be observed for all four neurons, as can be seen in figures 6 - 9. Furthermore, the improvements again increase in magnitude with increasing complexity of the model.

3.3 One Neuron

When only fitting the model on a single neuron, the improvements get even more substantial. This is represented in figures 10 - 13, where for two neurons each the models were fitted separately. It can be seen that for neuron 6, no patterns were observable in the raw data, but after applying the results of the model, a pattern emerges. The same trend can be observed in neuron 511, where the spatial pattern gets more precise after applying the model. Again, with increasing complexity of the model, the impact on the spatial pattern increased.

3.4 Optimization

As we discussed in chapter 2.3, the models can be tweaked by applying different regularization factors. This is done for example to prevent overfitting. To explore which impact these regularization factors have on the shift magnitude of the data, I evaluated these hyperparameters by computing models with different hyperparameter combinations on a synthetic dataset. An overall observation is that the shifts increase with increasing complexity of the models, but decrease for the piecewise model with 2 knots, the most complex model used for the analysis.

3.4.1 Warping Magnitude

In figure 14, results of testing different warping magnitude penalties can be observed. This parameter influences the magnitude of the shift applied by the model. It can be seen that applying no penalty on the warping magnitude leads to larger shifts on average, with decreasing effects for more complex models.

3.4.2 Smoothness of Template

The impact of different smoothness penalties to prevent the model from producing jittery data, can be seen in figure 15. Apart from the overall observation about shift magnitude, it is noticeable that for the shift model, the shift decreases with higher regularization, whereas for the other models the shift increases with higher regularization.

3.4.3 L2 Norm/ Size of Template

When observing changes for applying different factors for regularizing the size of the template with the L2 norm, nearly no difference can be observed within one model. This is depicted in figure 16.

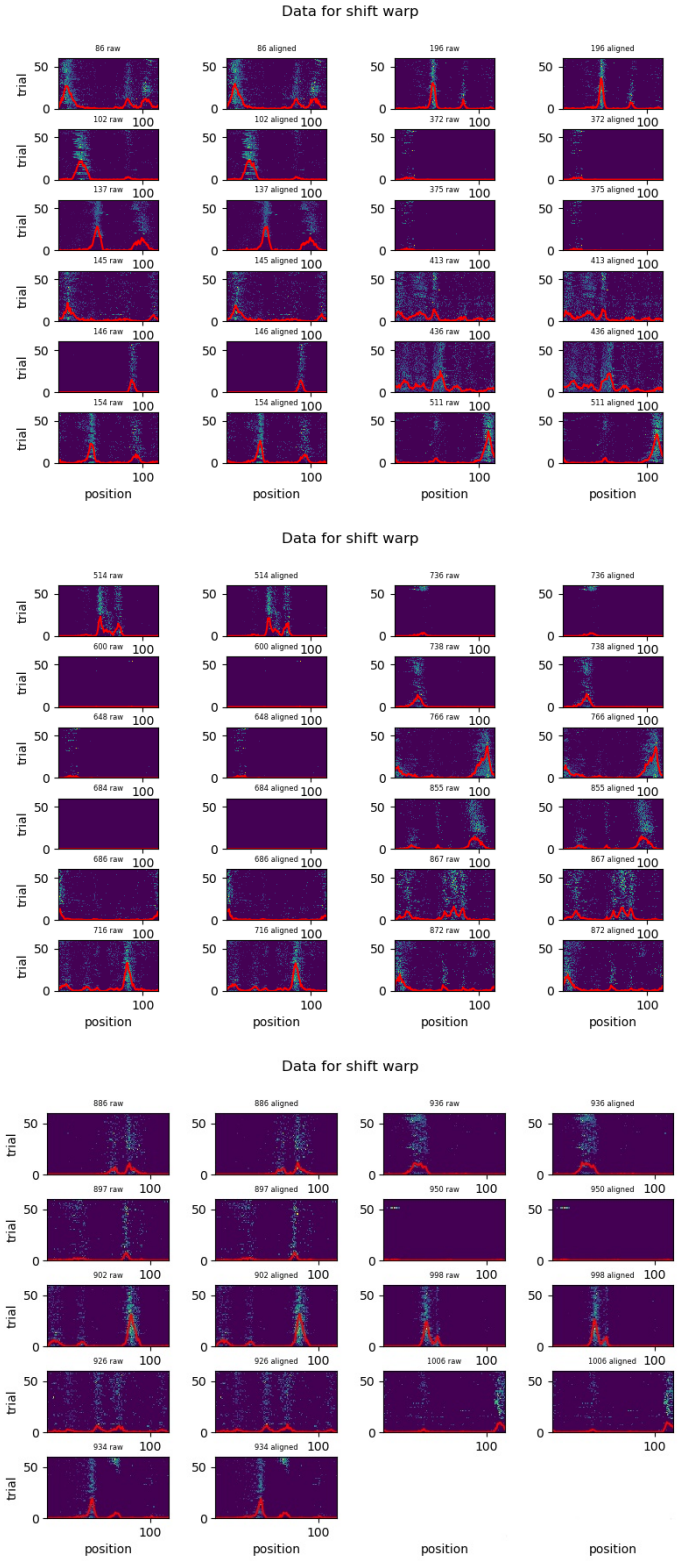

Figure 2: Shift Model - Higher activity at a position is displayed by a brighter color (ranging from dark blue to yellow) and the respective firing rate is presented in red. On the left, the raw data and on the right next to it the transformed data are plotted.

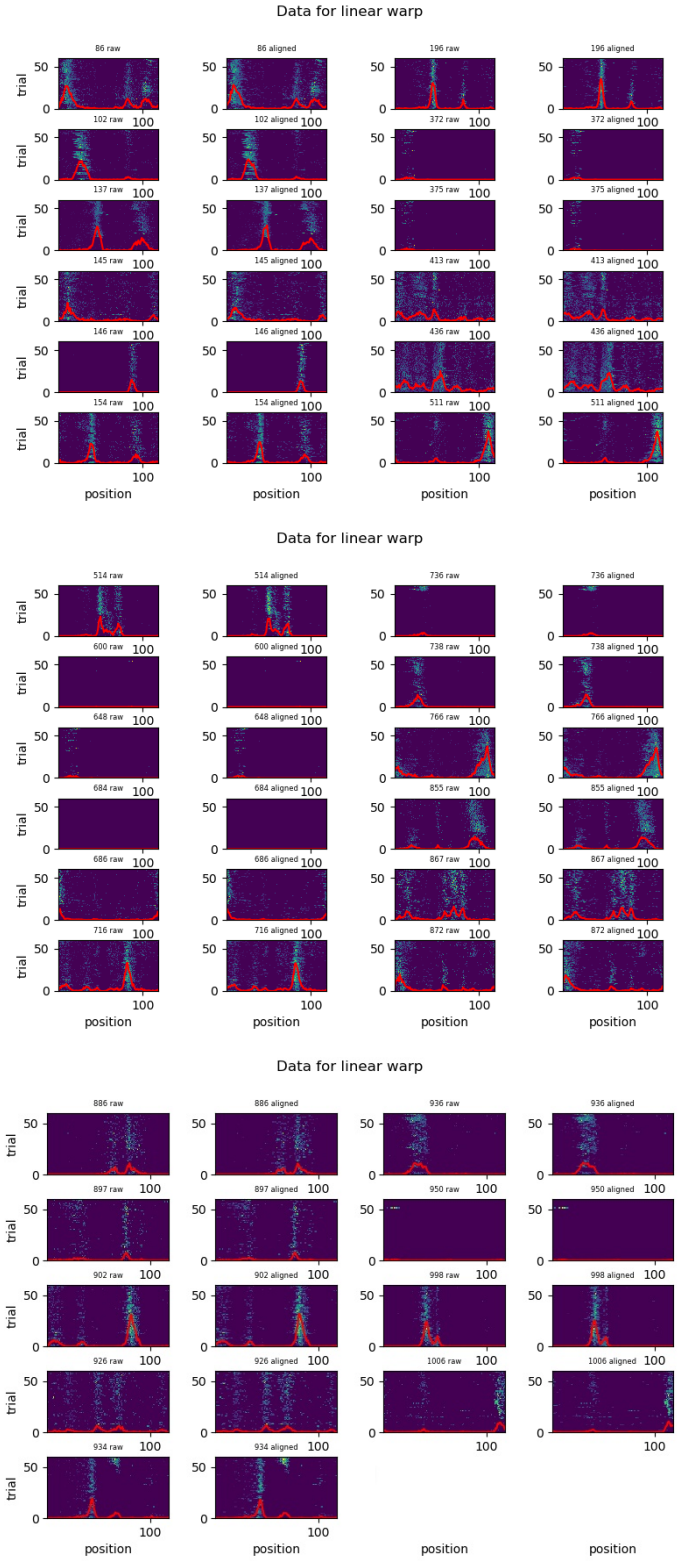

Figure 3: Linear Model - Higher activity at a position is displayed by a brighter color (ranging from dark blue to yellow) and the respective firing rate is presented in red. On the left, the raw data and on the right next to it the transformed data are plotted.

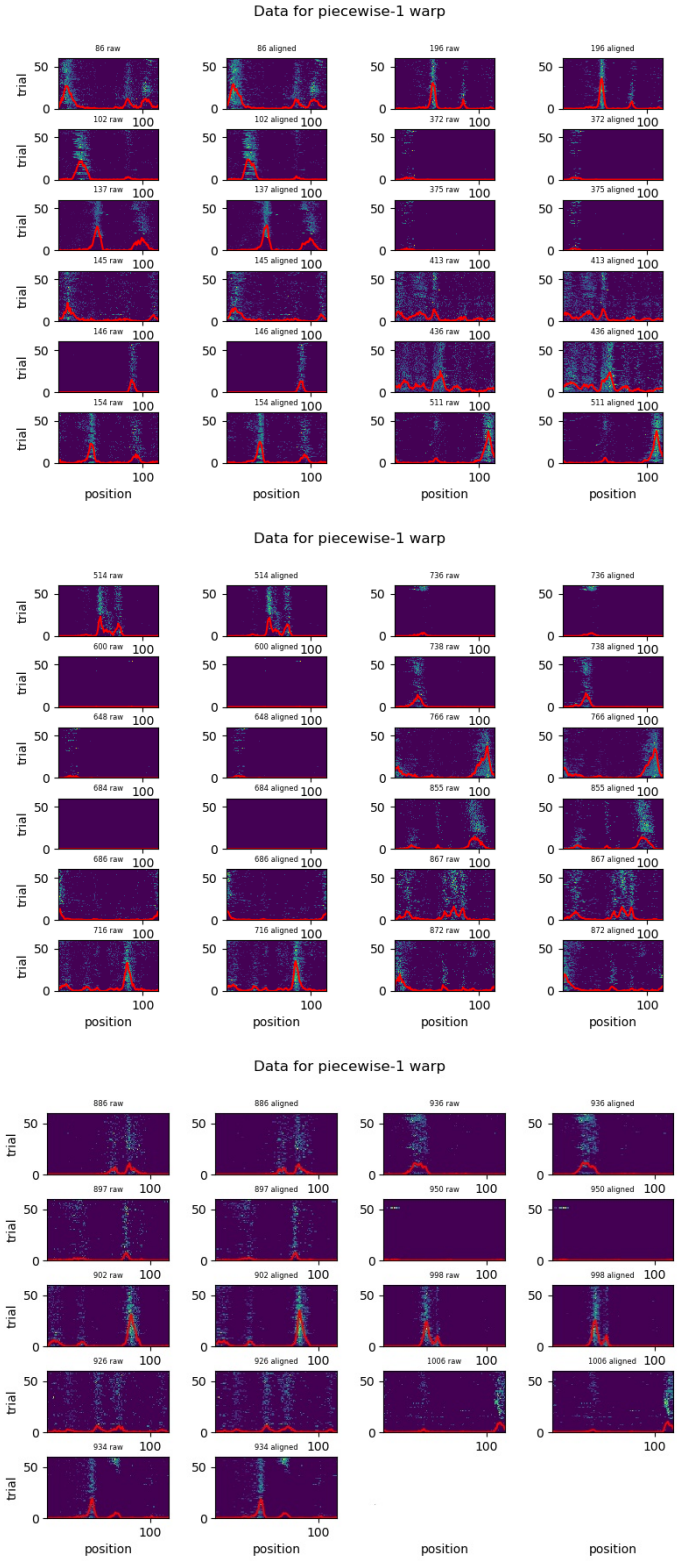

Figure 4: Piecewise 1 Model - Higher activity at a position is displayed by a brighter color (ranging from dark blue to yellow) and the respective firing rate is presented in red. On the left, the raw data and on the right next to it the transformed data are plotted.

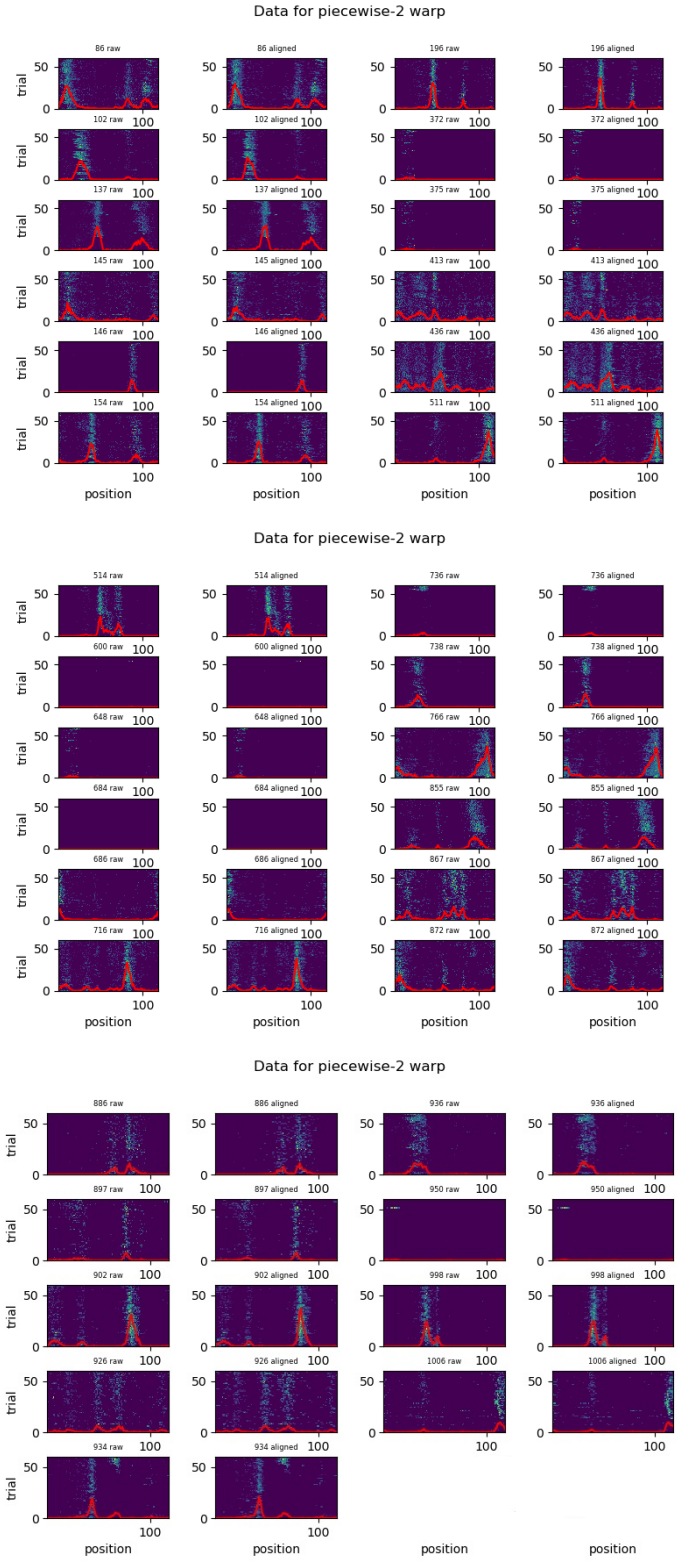

Figure 5: Piecewise 2 Model - Higher activity at a position is displayed by a brighter color (ranging from dark blue to yellow) and the respective firing rate is presented in red. On the left, the raw data and on the right next to it the transformed data are plotted.

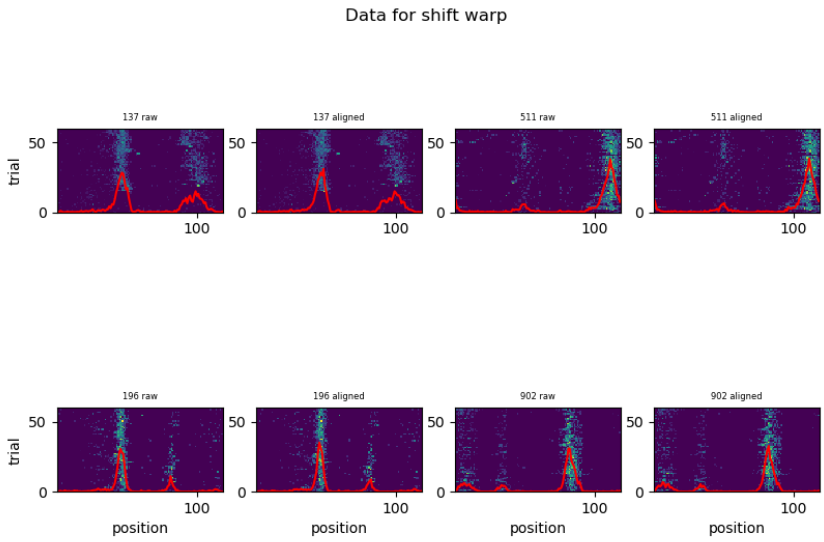

Figure 6: Shift Model - Higher activity at a position is displayed by a brighter color (ranging from dark blue to yellow) and the respective firing rate is presented in red. On the left, the raw data and on the right next to it the transformed data are plotted.

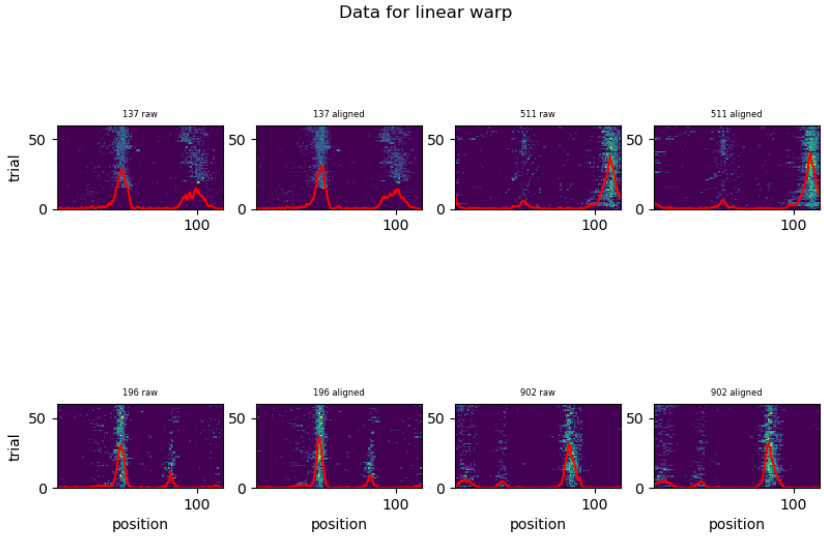

Figure 7: Linear Model - Higher activity at a position is displayed by a brighter color (ranging from dark blue to yellow) and the respective firing rate is presented in red. On the left, the raw data and on the right next to it the transformed data are plotted.

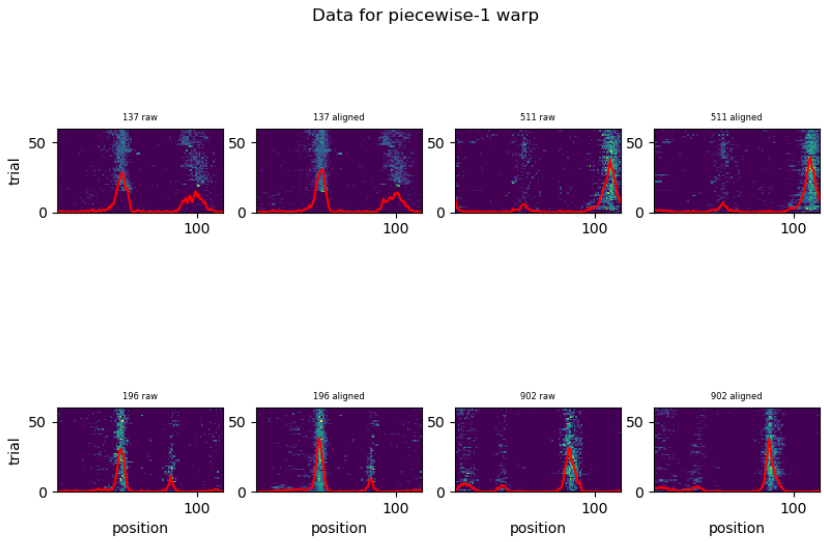

Figure 8: Piecewise 1 Model - Higher activity at a position is displayed by a brighter color (ranging from dark blue to yellow) and the respective firing rate is presented in red. On the left, the raw data and on the right next to it the transformed data are plotted.

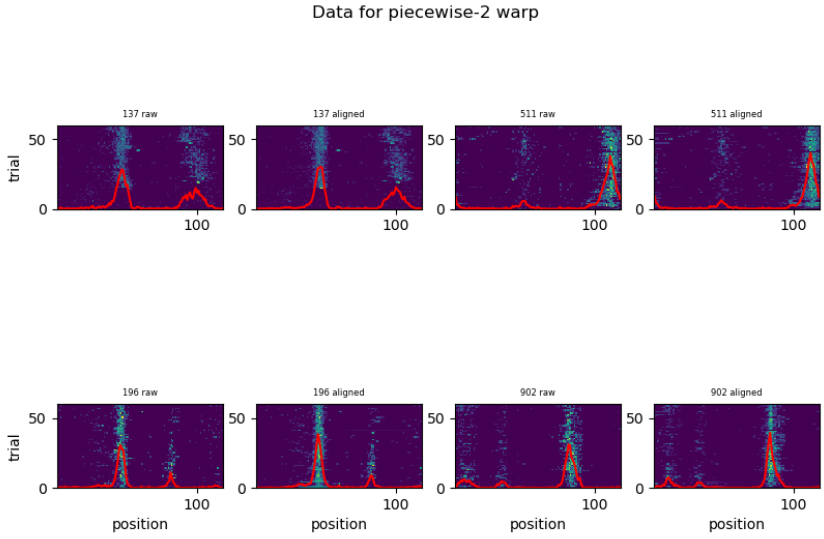

Figure 9: Piecewise 2 Model - Higher activity at a position is displayed by a brighter color (ranging from dark blue to yellow) and the respective firing rate is presented in red. On the left, the raw data and on the right next to it the transformed data are plotted.

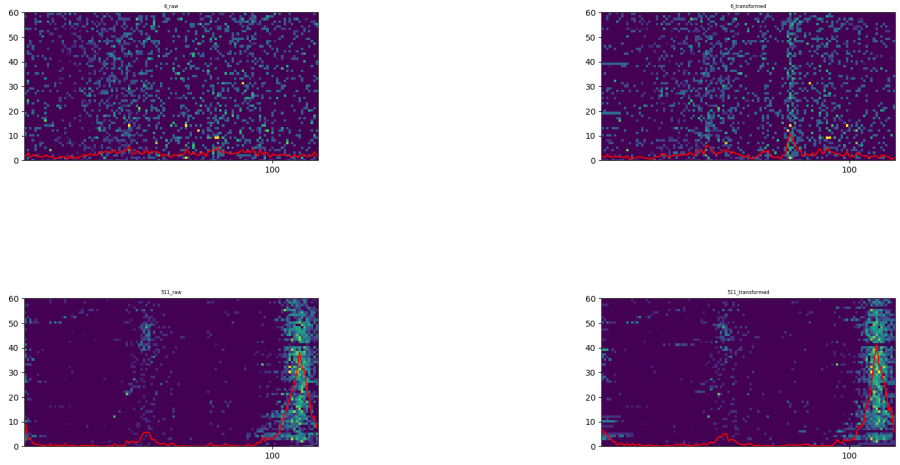

Figure 10: Shift Model: Fits for single neurons; two examples - Higher activity at a position is displayed by a brighter color (ranging from dark blue to yellow) and the respective firing rate is presented in red. On the left, the raw data and on the right next to it the transformed data are plotted.

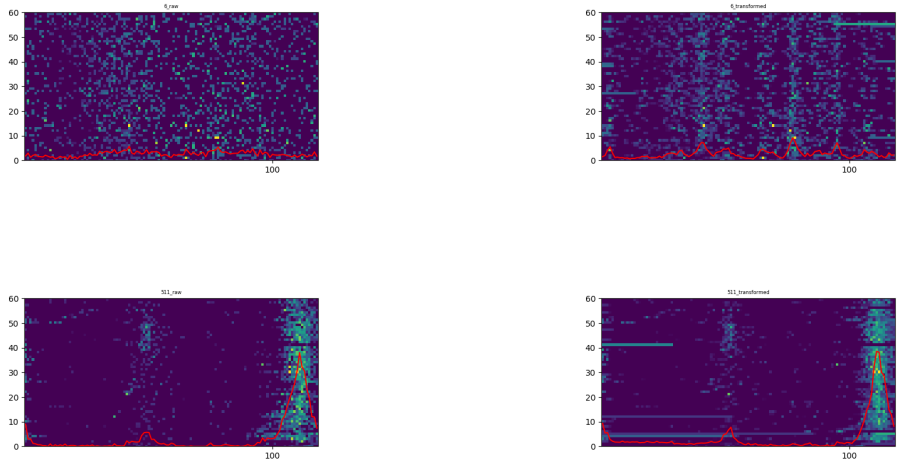

Figure 11: Linear Model: Fits for single neurons; two examples - Higher activity at a position is displayed by a brighter color (ranging from dark blue to yellow) and the respective firing rate is presented in red. On the left, the raw data and on the right next to it the transformed data are plotted.

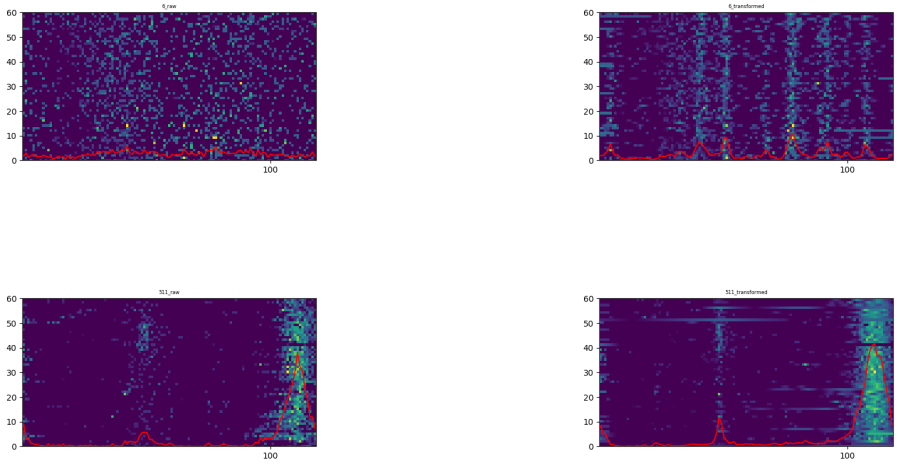

Figure 12: Piecewise 1 Model: Fits for single neurons; two examples - Higher activity at a position is displayed by a brighter color (ranging from dark blue to yellow) and the respective firing rate is presented in red. On the left, the raw data and on the right next to it the transformed data are plotted.

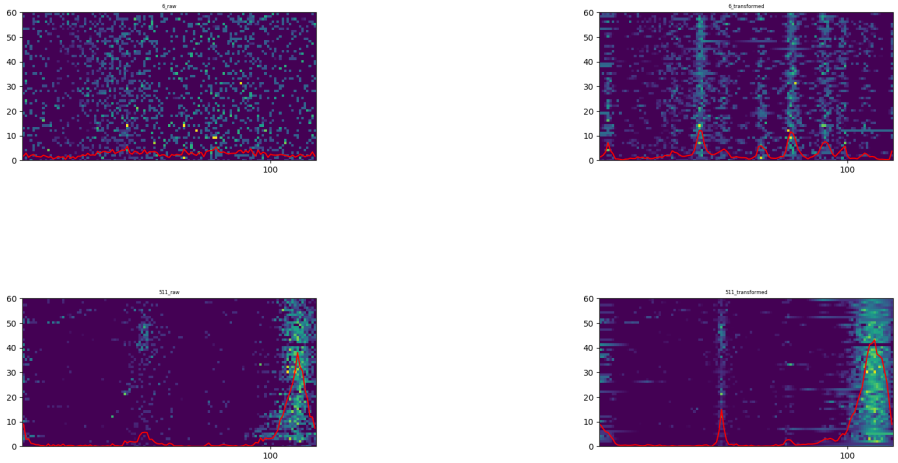

Figure 13: Piecewise 2 Model: Fits for single neurons; two examples - Higher activity at a position is displayed by a brighter color (ranging from dark blue to yellow) and the respective firing rate is presented in red. On the left, the raw data and on the right next to it the transformed data are plotted.

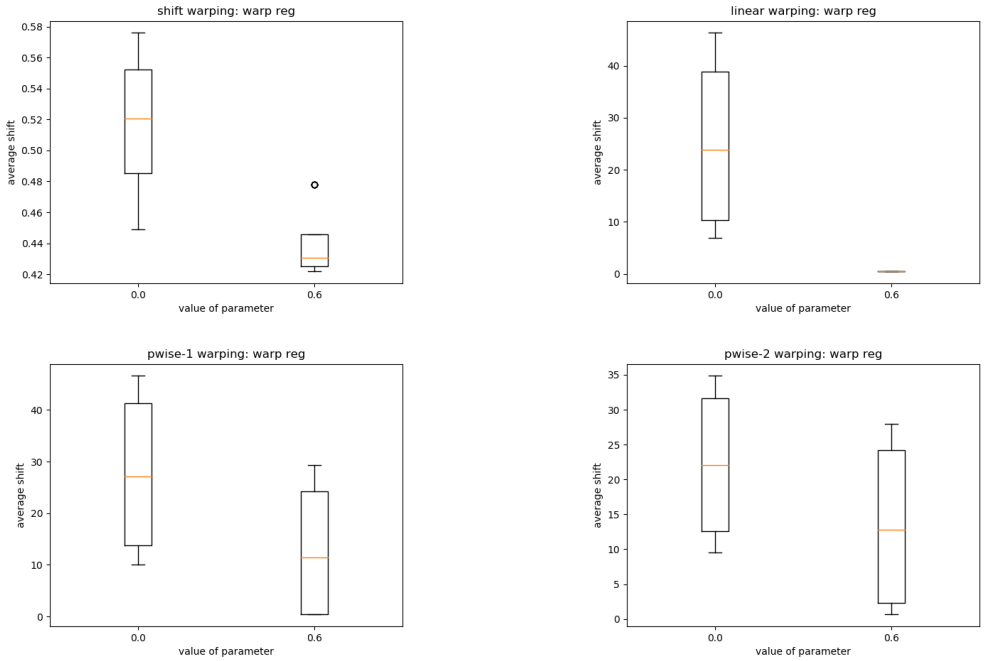

Figure 14: Impact of warping regularization on shift magnitude

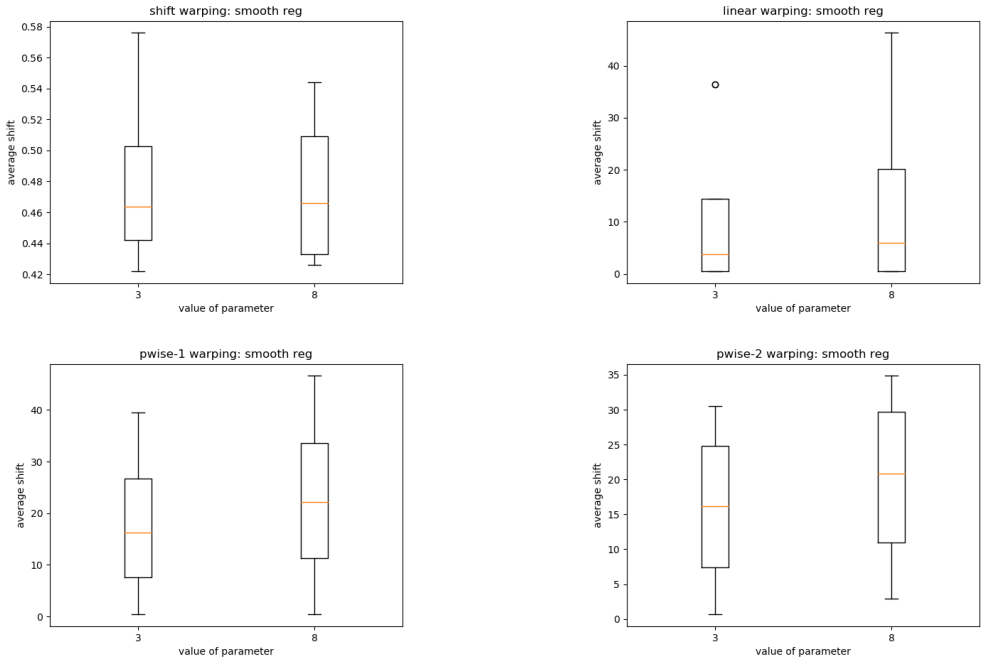

Figure 15: Impact of smooth regularization on shift magnitude

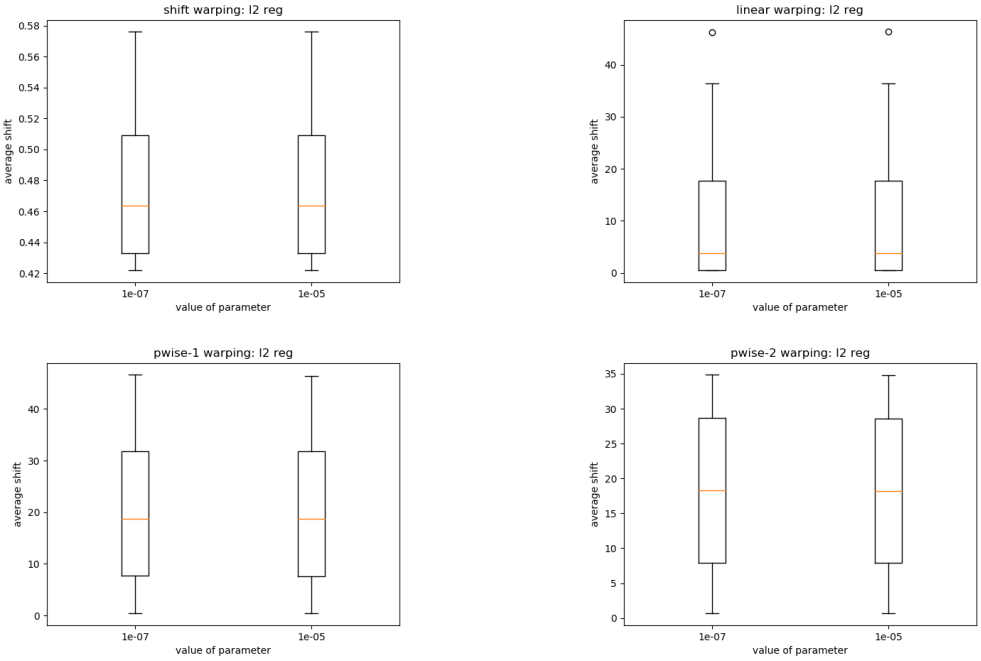

Figure 16: Impact of L2 regularization on shift magnitude

4 Conclusion

To conclude my findings, it can be said that using warping on spatial data positively influences the ability to identify place cells. The alignment effect is strongest when only one neuron is computed, and gets worse when more neurons are taken into account and even worse when noise is added by also taking into account neurons with less substantial spatial patterns. However, the seemingly optimal alignment when only modeling one single neurons is likely due to overfitting and using only one neuron for warping should therefore be used with caution.
When searching for the best hyperparameters to use with the model, my findings suggest that L2 and smooth regularization have relatively little impact on the shift magnitude compared to the warp size regularization. Yet, the effect of the warp size regularization declines when using more complex models. However, since the measurements were done on synthetic data, they might not represent the effect on real-world datasets. It is recommended to use the nested crossvalidation algorithm provided in the Python library to find the best parameters.
These findings are important especially for the analysis and interpretation of recordings in repetitive tasks with deviations between trials, e.g. mice running in a maze, but with inconstant speed and the possibility of stops during the measurement.

4.1 Open Questions

Even though the findings show the usefulness of the algorithm, there are still a lot of questions open for further research. One example would be whether the algorithm improves - or even would be more likely to overfit - when separating the dataset into spatial bins and only aligning the neurons that have a large firing rate in this respective bin. This could be a good way to improve the alignments, because the neurons with little activity in a bin might introduce noise to the fits of more active neurons.
Related to this, it could also be helpful to evaluate the impact of using piecewise linear functions with more than 2 knots, because there might be more places where the alignment changes. Another topic to look into would be whether the places of knots align with the mouse stopping or the slope of the linear functions corresponds to the speed of the animal. This way, possible relationships to the behavior could be investigated.

5 Appendix

5.1 User Guide

In order to use the same analysis that I did, follow this short user guide:

1. Prepare data so that your dataset is one .mat file containing a matrix of comntinuous data with dimensions (spike time or position x trial x neuron ID) and a list of neuron IDs
2. Install Python (optimally in a conda environment)
3. Install the affinewarp library from GitHub (https://github.com/ahwillia/affinewarp) by following the instructions in the README file (may have to adjust the requirements file with updated library names (sklearn → scikit-learn))
4. Place script from
   https://github.com/ChristineGra/position_warping/blob/master/data_exploration.py
   in your directory
5. Call script with path to your dataset and path to folder to save in as input parameter:
   Python data_exploration.py path_to_data path_save
   Optional parameters are a list of neurons to fit the model on, a specific model, a list of trials to select and hyperparameters for the models. Further information on these parameters can be found with typing
   Python data_exploration.py -h

If you want to use other data formats, different models or other visualizations, the examples on the GitHub page (https://github.com/ahwillia/affinewarp) could be a good source extend/ modify the analysis or write your own script. I highly recommend using nested crossvalidation (also implemented in the affinewarp library) when searching for the best model parameters.

References

[1] Edvard I Moser, Emilio Kropff, and May-Britt Moser. Place cells, grid cells, and the brain's spatial representation system. *Annu. Rev. Neurosci.*, 31:69–89, 2008.

[2] Alex H Williams, Ben Poole, Niru Maheswaranathan, Ashesh K Dhawale, Tucker Fisher, Christopher D Wilson, David H Brann, Eric M Trautmann, Stephen Ryu, Roman Shusterman, et al. Discovering precise temporal patterns in large-scale neural recordings through robust and interpretable time warping. *Neuron*, 105(2):246–259, 2020.