# Planar Hand Trajectory Reconstruction From Intraneuronal Recordings of a Macaque Monkey

Hugh Dickens, Giorgio Martinelli, Rahel Ohlendorf, Michal Olak
**Curious George**
Department of Bioengineering
*Imperial College London*
{hwd20, gm820, rmo20, mro20}@ic.ac.uk

***Abstract*—This study outlines methods of planar hand trajectory estimation from brain recordings of a monkey performing 500ms arm movements. Spike trains were analysed from 98 neuronal units while the monkey reached along 8 different angles. The models were evaluated using root mean squared error (RMSE) with respect to the actual trajectories. The best performing model consisted of decoding the reaching angle using linear discriminant analysis (LDA) before applying an angle-dependent linear regression (LR) model. This achieved an RMSE of 16.2cm and runtime of 14.0s.**

## I. Introduction

Brain Machine Interfaces (BMIs) have been investigated for the past half century. Recent breakthroughs have shown it is possible to decode neuronal signals for operating active prostheses with both invasive and non invasive techniques [1]. However, the main issue is their reliability in real world applications due to a current lack of robustness in control. Invasive techniques generally involve animal testing on rats or monkeys to validate the process before testing on humans [2].

This study focuses on decoding of neuronal activity of data gathered invasively from a macaque monkey brain. The dataset was recorded whilst the monkey reached in 8 different directions. The aim was to determine the 2D position of the hand by decoding the neuronal data. The objective was to create a decoding algorithm which is both accurate and fast to demonstrate the feasibility of controlling prosthetic devices.

98 neurons over 100 trials were monitored for each of the 8 reaching directions over time. The data length slightly differed from trial to trial, but in general the movement began at around 300 ms and ended 100 ms before the end of recording. The neuronal activity was represented in a binary system; 0 corresponded to an inactive neuron and 1 to a firing neuron. The results presented here are obtained by dividing the data evenly and randomly in half between training and testing.

The paper is organised as follows. In Section II, the pre-processing and data visualisation is presented. Section III contains the decoding algorithms used to reconstruct the trajectories from neuronal data. The results follow in section IV. A discussion with further work succeeds this in Section V. Finally, conclusions are drawn in Section VI.

## II. Data Pre-processing

The plots presented in Fig. 1 visualise the neuronal data from a single trial. The raster plot presented in Fig. 1 shows

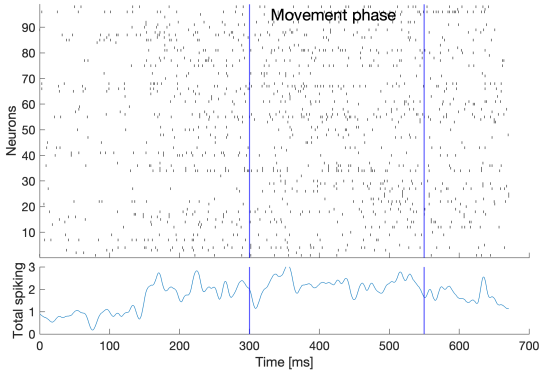


Fig. 1. Visualisation of neuronal activity during a single trial. Top: Raster plot indicating firing of each neuron. Bottom: Total firing rate summed from all neurons for each timestep, smoothed using a Gaussian function of window size 20 ms.

that the neurons vary greatly in terms of frequency and spike timing. It can be seen that for the first 150 ms the neuronal activity is sparse, followed by an increase in spike rate during the 150 ms before the movement onset. During the movement phase, the neuronal activity stays at a relatively high level and starts to decline shortly after the movement finishes.

### A. Binning

In order to perform feature extraction from the neuronal data we used the rate coding paradigm [3]. Our approach entailed a binning procedure which counted the number of spikes within a window; tuned to 20 ms. On examination of the raster plot in Fig. 1, it can be seen that the neurons fire infrequently. This coupled with sampling every 1 ms, meant that the temporal resolution of the neuronal data was high. Therefore, binning produced a meaningful feature for the decoding algorithm and decreased the runtime, by reducing the temporal resolution.

### B. Positional change

The positional data was binned to match the neuronal data format. The hand trajectory was encoded in a two-step process. Firstly, estimating the change in position over a bin window size for the x- and y-directions. Secondly, adding consecutive changes in position over the course of the movement to obtain the trajectory up to the current step. This approach helps to

focus on the evolution of movement and does not require information about the initial position at the decoding stage.

## III. Methods for Trajectory reconstruction

### A. Linear methods

Linear regression (LR) is used to decode neuronal data because it is easy to interpret and requires low computational power, leading to short runtimes [4].

The first approach trained a single linear regressor on the neurons' binned spike data using all 8 directions. The second approach trained a separate regressor for each reaching angle. For both methods, the output of the regressors were the corresponding changes in direction for each time period. Additionally, an individual linear regressor was used for the x- and y-directions. The MATLAB function "lsqminnorm" was used for this purpose. The function fits a linear model and at the same time reduces the norm of the model parameters. This poses an additional restraint on the model, making it possible to solve rank deficient input matrices. This is the case for our data, since some of the neurons fire infrequently within the time window used.

Decoding the reaching angle and the subsequent change in position are two fundamentally different problems. It was found that the neurons in the motor cortex are directionally dependent, meaning that different neurons fire for reaching activities in different directions [5]. This behaviour can be captured using separate algorithms for each direction, which should yield better results than a single linear regressor. Therefore, a twofold approach was chosen. Firstly, the angular direction of the movement was estimated by either k-nearest-neighbors (KNN) or linear discriminant analysis (LDA). Next, the respective LR model was used to separately estimate the x- and y-position for each angle.

### B. Direction inference - LDA/KNN

In order to decode the reaching angle before movement onset, the first 320 ms of neuronal activity was used for direction inference. This included the 300 ms of pre-movement neuronal activity. The training input was the spike count over the aforementioned window for each neuron, direction and trial. The corresponding label was the respective reaching angle. These input and output data were used for both algorithms.

KNN and LDA were chosen for the direction decoding because they are both widely used in data science and emphasise different aspects of data discrimination. KNN is powerful because it does not apply any assumptions about the form of the boundary between different classes. LDA, on the other hand, is restricted to a linear boundary, but it provides additional information about the importance of different classes. This might be especially useful in BMIs because it provides information about which neurons are indispensable for the decoding. For KNN, the MATLAB function "fitcknn" was used. The applied distance metric was the standard minkowski metric. For LDA, the MATLAB function "fitcdiscr" was used. The hyperparameters $\delta$ and $\gamma$ were optimized by automatic hyperparameter optimization, minimizing five-fold cross-validation.

### C. Deep Neural Network (DNN)

The methods presented above assumed a linear relationship between the binned neuronal spike trains and the hand trajectories. However, this is not necessarily the case for neuronal responses. Therefore, a deep learning approach was investigated to model possible non-linear characteristics.

The DNN was trained using spike train data for all trajectories. There was no classification of reaching angle before decoding the velocity due to the increased training and runtimes. The type of DNN used was a feedforward neural network to investigate the applicability of a deep learning approach. The pre-processing methods described in Section II were used.

The training data was fed into the DNN using a range of architectures. Although a full grid search was not conducted, the number of layers, activation functions, number of neurons, minibatch size, and epoch number were optimised with empirical testing. The resulting network consisted of 4 hidden layers; layer 1 = 100 neurons, layer 2 = 250 neurons, layer 3 = 100 neurons, layer 4 = 50 neurons. The layers were fully connected using tanh activation functions with 50% dropout for all hidden units in training to prevent overfitting [6]. A learning rate of 0.01, minibatch size of 30, and epoch number of 60 was used with the adam optimisation parameter. This architecture was implemented using the deep learning toolbox in MATLAB. The outputs of the DNN were the change in x- and y- positions over the binned period. The DNN had 98 input features corresponding to the number of neuronal recordings from the monkey.

### D. Population decoding

The methods presented thus far decode the trajectory from the neuronal data using rate coding. This assumes a relationship between the spike rate and a coded variable. Although this approach has been shown effective with motor neurons [7], population decoding provides additional information by analysing firing patterns [8]. Neurons in the motor cortex tend to fire more frequently when movement is generated in a direction corresponding to their receptive field. This relationship can be used to better estimate the movement trajectory.

Population decoding is analogous to classification. Thus, the neuronal data was binned as previously outlined in Section II, however, the position change data needed to be discretised to fit the classification paradigm. This was done by binning the maximal range of both x- and y-position change using an interval tuned to 0.05. Similarly to the procedure outlined in Section III a), the direction was first decoded using LDA and then an angle-specific KNN classifier was applied for each time step of the neuronal data to obtain the change in position.

### E. Complimentary filtering

In order to exploit different characteristics of rate and population coding methods, their outputs were combined using

a complimentary filter. The filter linearly combines two or more predictions in a preset proportion. As this approach requires training and inference of multiple algorithms, the fastest ones were chosen, namely angle-dependent LR and KNN. Filtering proportionality was tuned to 75:25. Their outputs were combined for trajectory estimation, while direction was determined using LDA.

## IV. Results

The performance of each approach presented in Section III is summarized in Table I. A score was calculated using a 20:80 split of runtime:RMSE. This split takes into account both the accuracy of the decoder as well as the time to train and test the algorithm. Both of these characteristics are vital for BMI applications. The proportions in the score reflect the subjective importance of both metrics. A low RMSE guarantees precise movements, which is the most important goal for applications, however, for online decoding applications long runtimes can be a limiting factor.

TABLE I
SUMMARY OF THE IMPLEMENTED METHODS. THE TRAINING TIME CORRESPONDS TO TRAINING 50 TRAJECTORIES, RUNTIME CORRESPONDS TO TESTING THE MODELS; INCLUDING PLOTTING THE TRAJECTORIES IN MATLAB. RMSE WAS CALCULATED AS A RUNNING SUM OF EACH 20MS BIN. THE SCORE WAS CALCULATED AS A 20:80 SPLIT OF RUNTIME AND RMSE.

| Decoding method | Time (s) | | RMSE (cm) | Score |
|---|---|---|---|---|
| | Train | Run | | |
| Vanilla LR | 0.77 | 5.27 | 45.6 | 37.5 |
| KNN + LR | 1.38 | 15.0 | 22.3 | 20.8 |
| LDA + LR | 1.48 | 14.0 | 16.2 | 15.8 |
| DNN | 9.88 | 28.8 | 30.6 | 30.2 |
| Pop. decoding: LDA+KNN | 2.95 | 102 | 27.3 | 42.2 |
| Comp. filtering: LDA+LR+KNN | 2.47 | 171 | 17.4 | 48.1 |

From Table I, the best decoder was LR with LDA achieving a score of 15.8; RMSE of 16.2cm and runtime of 14.0s. The next best method was using the similar approach of LR with KNN. These two results compared with the vanilla LR, score of 37.5, show that using a classifier for trajectory angle greatly improved performance. This is illustrated through the accurate decoding of trajectory angle displayed in Figure 2.

The linear methods with classification proved to have superior RMSE performance, followed by complementary filtering, population decoding, DNN, and finally the vanilla linear regressor.

As expected, the linear models achieved superior performance in terms of runtime; ranging from 5-15s. Following this was the DNN approach with a runtime of 28.8s. The slowest algorithms, with runtimes of over 100 seconds, were population decoding and complimentary filtering.

## V. Discussion and Further Work

The first approach to decode change in position with a single LR model led to very poor results. This was expected because of the directionality of the neurons and their inherent non linearities. To combat LR's limitations, two approaches were attempted. The first was to divide the task into two steps; decode the direction, and then decode the change in position. The second approach was to use an algorithm capable of decoding nonlinear relationships such as a DNN.

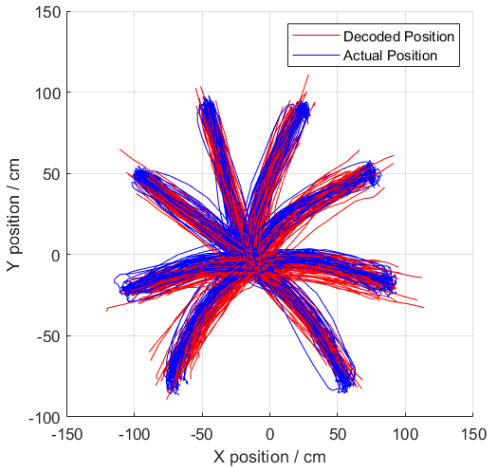


Fig. 2. Trajectory prediction of the linear regressor with LDA

For the first approach either KNN or LDA was used to determine the reaching angle. The runtime is similar for both methods since it scales similarly for a small number of data points, which is the case here. The RMSE is, however, significantly lower when using LDA. This suggests that the neuronal data can be separated linearly provided that a suitable projection is found. This knowledge can be helpful when further improving the decoder. Although the data can be separated within its natural basis by KNN, this approach is less reliable.

Decoding the reaching angle in a separate step might seem specific to this problem, however, it can be assumed that a real life problem can also be reduced to only a finite number of possible reaching angles. Our suggested approach is therefore not largely limiting the possible applications. Additionally, a linear method has the advantage of being readily accessible and interpretable, which allows it to be applied easily to a varying set of problems.

The second approach to model nonlinearities with DNN and population decoding proved to be inferior to the LR models. The DNN implemented in this study did not lead to the expected performance with respect to RMSE. This is likely due to only inputting 98 features to the DNN. A larger number of features such as a running sum of spike train data giving the model a form of memory might improve the performance. This would be implemented with a summing window which adds up the neuronal data in steps of 20ms from 20-500ms. Additionally, the DNN could be improved by creating direction dependent models, similar to the linear method implementation. The runtime of the DNN was slow compared to the other models. This was expected due to the complexity of the model and size of the network.

The population decoding performed poorly with regard to the RMSE. With its runtime being outstandingly high, it is deemed unusable in the current form. This high runtime was partially expected, as the discretisation of the positional change requires creation of a large number of classes, which inherently slows down the classification task. Nevertheless, this alternative approach still holds promise of obtaining information unavailable when only using rate decoding. It is possible that the implemented classification-based approach was not able to extract it due to its similarity to the regression approach through simple discretisation of the prediction space. A possible solution is changing the strategy from using a classifier to creating population vectors by correlating every neuron spiking rate with a particular direction. This approach should speed up the inference by eliminating the amount of classes.

The complimentary filtering approach reduced the RMSE compared to the population decoding alone. Nevertheless, as it required training and usage of as many as three algorithms, its runtime was unreasonably high. It is possible that this method is still overall beneficial, however it might be necessary to chose algorithms that have a higher chance of complimenting each other.

### A. Further Work

One approach to speed up the inference, as well as gain additional insights into the underlying data structure, is principal component analysis (PCA). A preliminary test of PCA was done in this study. The 98 binned neuronal inputs were projected into 98 principal components (PCs), the variance of which was decreasing in an exponential trend as expected. It was found that the proposed decoding solution worked almost equally well with only 40 PCs fed into it. Moreover, using the fact that PCs are linear combinations of the input features, the contributions of each neuron to all the PCs can be summed up to assess its importance. From this analysis across the 8 directions, only two thirds of the neurons provided meaningful input into the PCs. This represents two ways to use PCA for dimensionality reduction.

Other algorithms specialised for timed series data and the learning of long-term dependencies would be applicable for this decoding task [9]. These approaches include different DNN architectures such as recurrent neural networks (RNNs) [10] with implementations such as long short term memory (LSTM) networks [9] or gated recurrent units (GRUs) [9]. These methods were not tested, but their application looks promising as shown in [11] which decoded kinematic movement from large populations of neurons. One caveat of RNN applications is that more data needs to be collected with more input features. Alternatively, Kalman filtering represents a signal processing approach with promise for time-series prediction as shown in [12]. This could be particularly effective with direction-dependent models. Finally, applying Bayesian decoding to neural spike trains yields promise as shown in [13]. This method estimates uncertainty before committing to a decision which is particularly relevant for a prosthetic application of BMIs.

## VI. Conclusions

Planar arm movements of a macaque monkey were decoded using brain signals from invasive brain recordings. Different decoding methods have been explored including linear regression, classification and a deep neural network. The trade off between accuracy and speed was best for multiple linear regression with prior direction inference using LDA, achieving an RMSE of 16.2cm and runtime of 14.0s. Enhanced data preprocessing and alternative algorithmic approaches that might improve decoding have also been discussed. Moreover, the population decoding paradigm could yield additional information given its alternative architecture.

## VII. Attributions

HD implemented the DNN, wrote the corresponding sections, and wrote the results. GM looked into different options of data pre-processing, implemented LDA and wrote the introduction. RO improved the data pre-processing, implemented LR and KNN and wrote the corresponding sections. MO did the pre-processing, looked into PCA, investigated different approaches such as population decoding and complimentary filtering and wrote the corresponding sections.

## VIII. Acknowledgments

We would like to express our gratitude to Curious George for his guidance and support. In the darkest hour, you were our brightest Macaque.

## References

[1] D. J. McFarland and J. R. Wolpaw, "Brain-computer interface operation of robotic and prosthetic devices," *Computer*, 2008.

[2] C. B. V. Gilja, C. Pandarinath, "Clinical translation of a high-performance neural prosthesis," *Nature Medicine volume*, 2015.

[3] E. R. Kandel, J. H. Schwartz, and T. M. Jessell, Eds., *Principles of Neural Science*, 3rd ed. New York: Elsevier, 1991.

[4] S. Waldert, "Invasive vs. non-invasive neuronal signals for brain-machine interfaces: Will one prevail?" *Frontiers in Neuroscience*, 2016.

[5] W.-k. Tam, T. Wu, Q. Zhao, E. Keefer, and Z. Yang, "Human motor decoding from neural signals: a review," *BMC Biomedical Engineering*.

[6] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," *CoRR*, 2012.

[7] E. D. Adrian and Y. Zotterman, "The impulses produced by sensory nerve-endings," *The Journal of Physiology*, 1926.

[8] S. Wu, S.-i. Amari, and H. Nakahara, "Population Coding and Decoding in a Neural Field: A Computational Study," *Neural Computation*, 2002.

[9] J. I. Glaser, A. S. Benjamin, R. H. Chowdhury, M. G. Perich, L. E. Miller, and K. P. Kording, "Machine learning for neural decoding," 2020.

[10] A. E. H. Yeganegi, Y. Fathi, "Decoding hind limb kinematics from neuronal activity of the dorsal horn neurons using multiple level learning algorithm." *Sci Rep*, 2018.

[11] M. L. P. H. Tseng, N. A. Urpi and M. Nicolelis, "Decoding movements from cortical ensemble activity using a long short-term memory recurrent network," *Neural Comput.*, 2019.

[12] W. Q. Malik, W. Truccolo, E. N. Brown, and L. R. Hochberg, "Efficient decoding with steady-state kalman filter in neural interface systems," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 2011.

[13] S. Koyama, U. T. Eden, E. N. Brown, and R. E. Kass, "Bayesian decoding of neural spike trains," *Annals of the Institute of Statistical Mathematics*, 2009.