Tensor Product Kernels for Multi-scale Control of Somatosensory Stimulation

Jose C. Principe, Ph.D.

Distinguished Professor

University of Florida

Computational NeuroEngineering Laboratory (CNEL)

Department of Electrical & Computer Engineering

University of Florida

www.cnel.ufl.edu

Acknowledgments

- Dr. Joe Francis, SUNY and his group

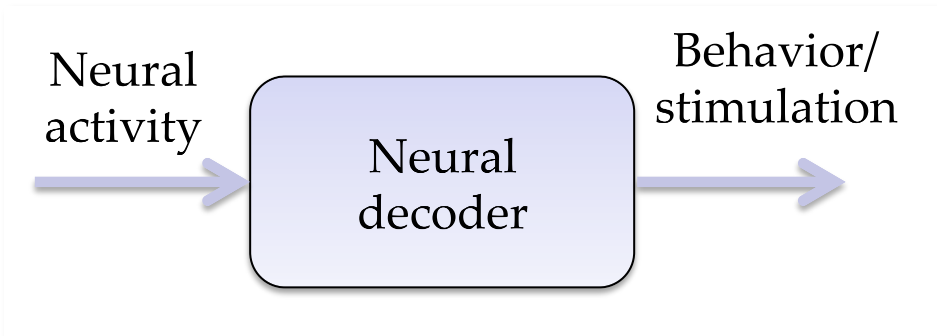
  Dr. Lin Li
  Dr. Austin Brockmeier
  Kan Li

- DARPA Repair N66001-10-C-2008

Neural Decoding

Brain

- Highly **distributed and dynamic** system
- **Largely unknown** functional structure
- **Partial observable** system (input and output)



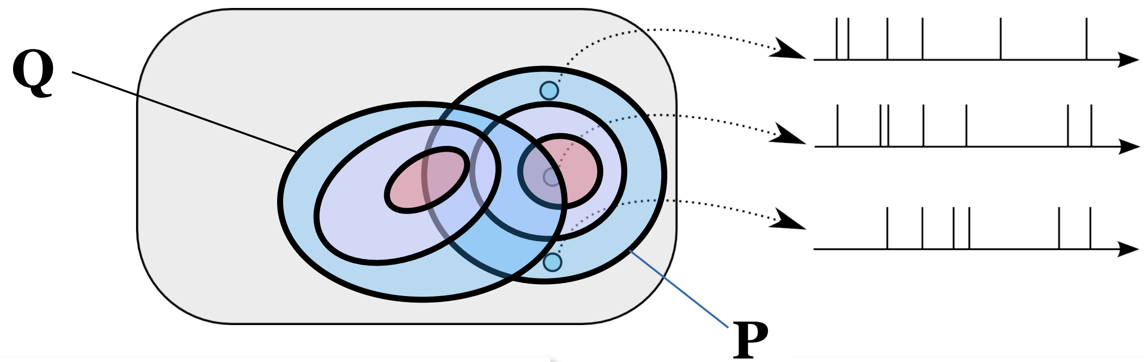
- **Goal:** Decoding information from neural activity
- **Tools:** Kernel based machine learning

Outline

- Kernel based Framework for **Spike Trains**
- Kernel based Framework for **Multi-scale Neural Activity**
- Conclusions

Spike trains

- Neurons communicate through electrical pulses, called spikes
- The activity of an individual neuron is described by **a sequence of events** occurring in time. $s_i = \{t_n \in T : n = 1, \ldots, n\}$
- A spike train can be viewed as a realization of a point process,



No algebraic structure in the space of event time sets

No functional machine learning algorithm can be applied directly

A point process is fully described by the conditional intensity function

$$\lambda(t \mid H_t) = \lim_{\Delta t \to 0} \frac{\Pr\{\text{event in}[t, t+\Delta t) \mid H_t\}}{\Delta t}$$

(with Poisson assumption, $\lambda\ (t \mid H_t)$ can be simplified to $\lambda\ (t)$, which can be estimated from one realization.)

Requirements for signal processing with spike trains

	topology	metric	linear structure	norm	inner product	
Metric space	O	O				k-Nearest Neighbor algorithm
Banach space	O	O	O	O		k-means algorithm
Hilbert space	O	O	O	O	O	Support Vector Machine, Least squares, PCA, CCA, …
Point processes?	?	?	?	?	?	

Most signal processing algorithms operate in Hilbert space

How to map spike trains to Hilbert spaces?

- **Existing Approach**

Discretized representation

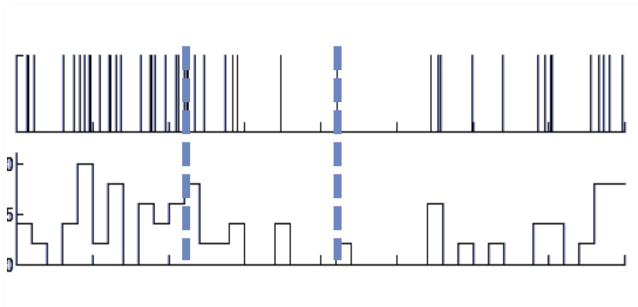
(most popular approach)

$$r(n) = \frac{1}{\Delta}\int_{(n-1)\Delta}^{n\Delta} s(t)\,\mathrm{d}t$$

Drawbacks:

- large window: lose fine time information

- small window: exacerbate the data sparseness and increase computation cost



$\underbrace{0.023\ 0.045\ \ 0.076}$

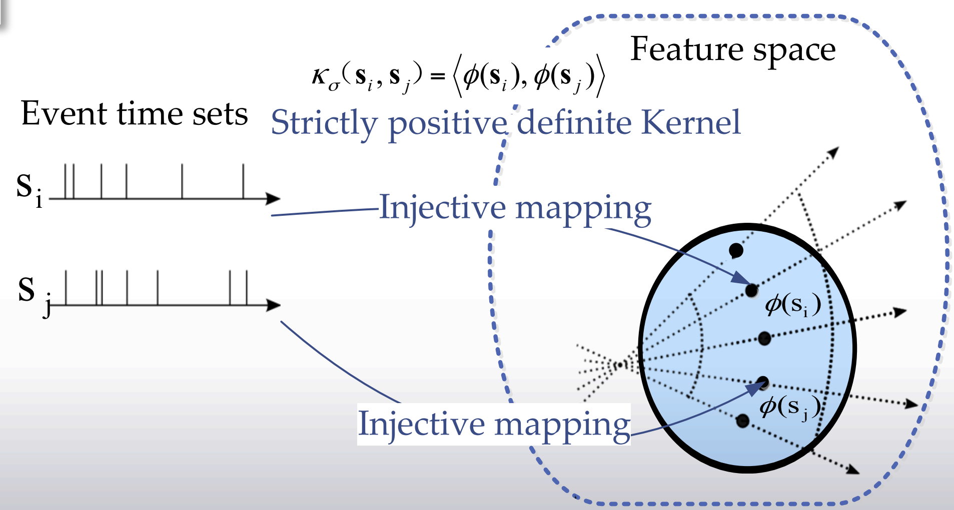
3 dimensions

$\underbrace{000000000000001000000000010.........00000000}$

200 dimensions

- **Our idea**
  - **Kernel in event sets**

    (Operator on event times)



Functional Representation of Spike Trains

Cross-intensity kernels

- Given two point processes p_i, p_j, define the inner product between their intensity functions

$$I(p_i, p_j) = \left\langle \lambda_{p_i}(t \mid H_t^i), \lambda_{p_j}(t \mid H_t^j) \right\rangle_{L_2(T)}$$
$$= E[\int_T \lambda_{p_i}(t \mid H_t^i) \lambda_{p_j}(t \mid H_t^j) dt]$$

- This yields a family of cross-intensity (CI) kernels, in terms of the model imposed on the point process history, H_t.

Paiva A., Park I., Príncipe J. and DeMarse T., "A Reproducing Kernel Hilbert Space framework for Spike Train Signal Processing", Neural Computation, vol 21, #3, 424-449, 2009

Kernels for spike trains

- Set of event times: $s(t) = \sum_{n=1}^{N} \delta(t - t_n)$
- Functional representation: $\hat{\lambda}_s(t) = s(t) \otimes h(t) = \sum_{n-1}^{N} h(t - t_n)$

- **Cross intensity (CI) kernel:** $\kappa_C(s_i, s_j) = \left\langle \hat{\lambda}_{s_i} - \hat{\lambda}_{s_j} \right\rangle = \int_T \hat{\lambda}_{s_i}(t)\hat{\lambda}_{s_j}(t)\,\mathrm{d}t$
  - Inner product

- **Nonlinear cross intensity (NCI) kernel:** $\kappa_N(s_i, s_j) = \int_T e^{\frac{\left(\hat{\lambda}_{s_i}(t) - \hat{\lambda}_{s_j}(t)\right)^2}{2\sigma^2}}\,\mathrm{d}t$
  - Correntropy
  - Positive definite kernel

- **Schoenberg kernel :**
  - Positive definite kernel
  - Sensitive to nonlinear couplings in time structure

$$\kappa_s(s_i, s_j) = \exp\left(-\frac{\left\| \hat{\lambda}_{s_i}(t) - \hat{\lambda}_{s_j}(t) \right\|^2}{2\sigma^2}\right),$$

$$\text{where } \left\| \hat{\lambda}_{s_i}(t) - \hat{\lambda}_{s_j}(t) \right\|^2 = \left\langle \hat{\lambda}_{s_i}, \hat{\lambda}_{s_i} \right\rangle + \left\langle \hat{\lambda}_{s_j}, \hat{\lambda}_{s_j} \right\rangle - 2\left\langle \hat{\lambda}_{s_i}, \hat{\lambda}_{s_j} \right\rangle$$

Park I., Seth S., Rao M., Principe J., “Strictly positive definite kernels for point process divergences” Neural Computation, Vol 24 Issue 8, 2223-2250, August 2012
Park I., Seth S., Paiva A., Li L., Principe J., “Kernel methods on spike train space for neuroscience: a tutorial” , IEEE SP Magazine, 149-160, June 2013..

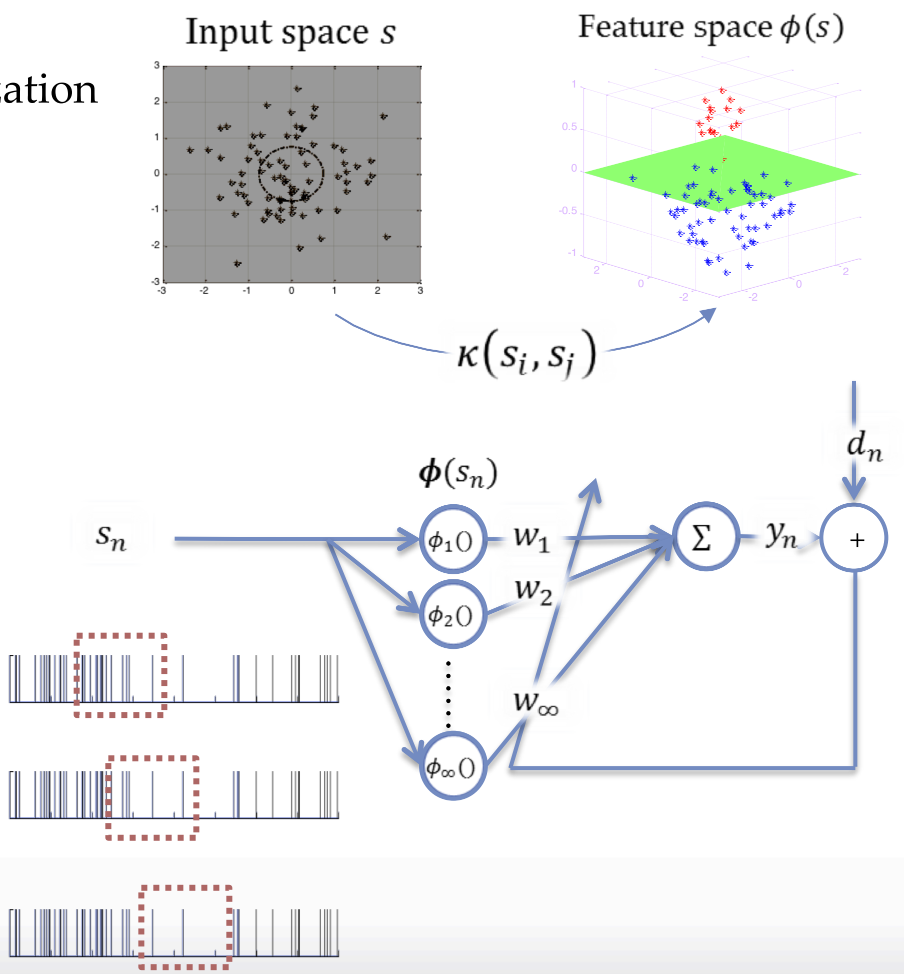
Kernel-based Regression on Spike trains

- Advantages
  - Nonlinear mapping →linear optimization
  - No local minima
- Kernel based methods
  - RBF: Radial basis function
  - SVR: support vector regression
  - KLMS: kernel least mean square
  - KRLS: kernel recursive least square
- Kernel Least Mean Square (KLMS)

  Online adaptation

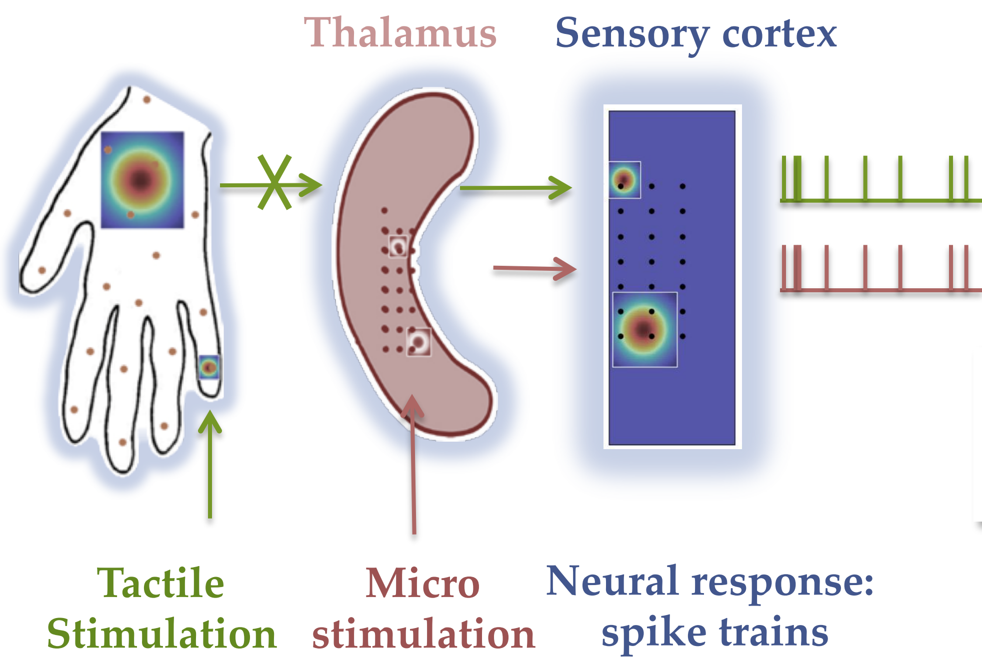
  Low computation time

$$\Omega(0) = 0$$
$$\Omega(n+1) = \Omega(n) + \eta e(\mathrm{n})\phi(s_n)$$
$$\Omega(n) = \eta \sum_{i=1}^{n-1} e(i)\phi(s_i)$$
$$y(n) = \langle \phi(s_n), \Omega(n) \rangle = \eta \sum_{i=1}^{n-1} e(i)\phi(s_i)$$



Liu W., Pokarel P., Principe J., "The Kernel LMS Algorithm", IEEE Trans. Signal Proc., V. 56, # 2 :543 - 554, 2008

Scenario: Adaptive Inverse Control



Challenge 1: system is dynamic

Challenge 2: output is a spike train (point process)

Adaptive control

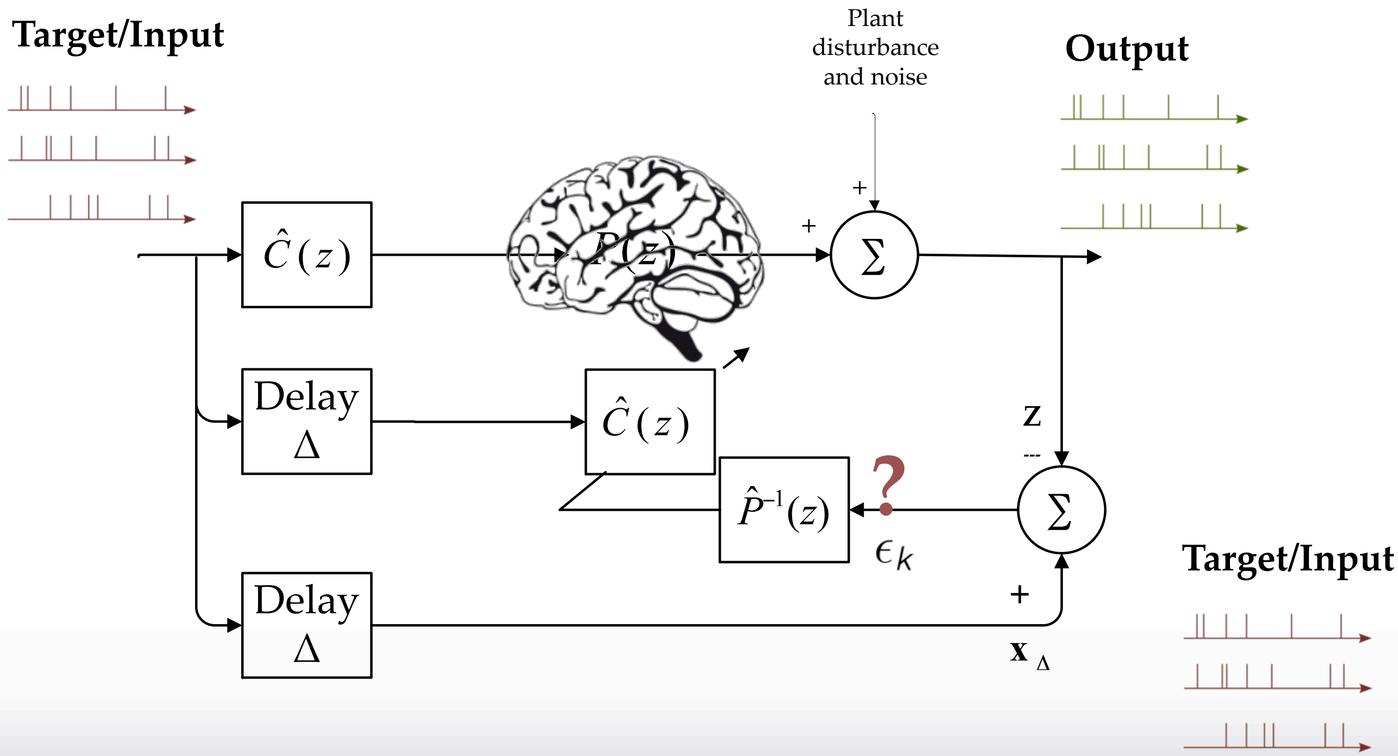
Solution[1]: Adaptive inverse control + Schoenberg kernel for point process data

[1] Li, L., Principe J., "Adaptive Inverse Control of Neural Spatiotemporal Spike Patterns with a Reproducing Kernel Hilbert Space (RKHS) Framework", IEEE T. Neural Sys. Rehab. Eng., vol 21, #4, 532-543, 2012.

Adaptive inverse control diagram

$P(z)$: plant (neural circuit)
$\hat{C}(z)$: inverse controller
$\hat{P}^{-1}(z)$: inverse model of $P(z)$
Δ: half of the window size

Challenge:
No definition of the distance between two event time sets ϵ_k



[1]Adaptive Inverse Control (Lin 2012)

Kernel-based view simplifies the problem

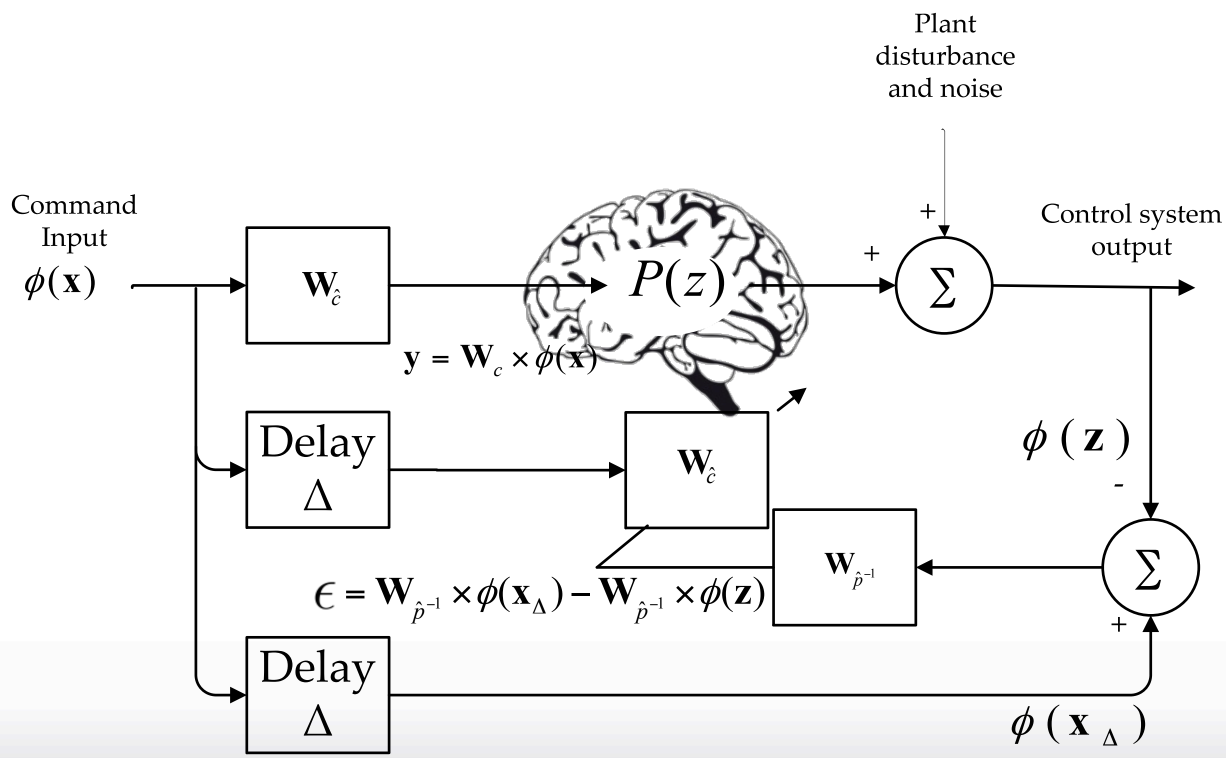
$P(z)$: plant (neural circuit)
$\hat{C}(z)$: inverse controller
$\hat{P}^{-1}(z)$: inverse model of $P(z)$
Δ: half of the window size

Challenge:
No definition of the distance between two event time sets ϵ_k

Our kernel designed for point process data

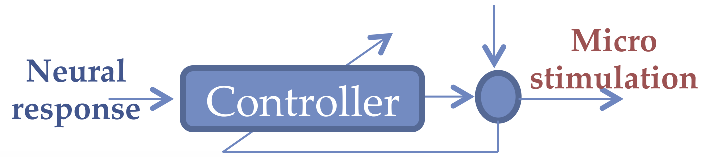
This distance is well defined in feature space

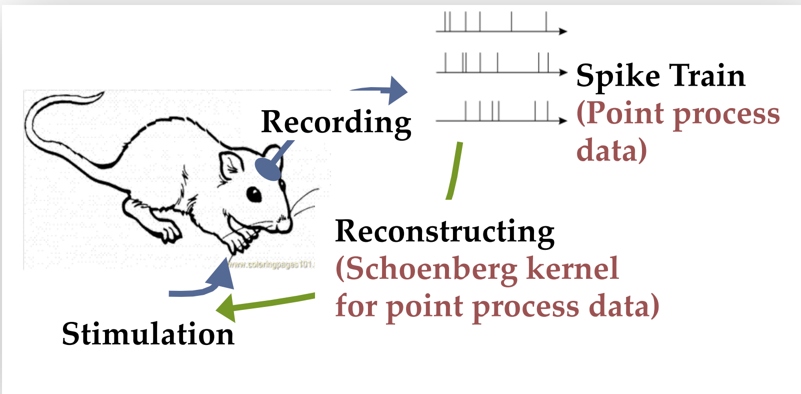
Advantage:
A linear structure in feature space→ no local minimum

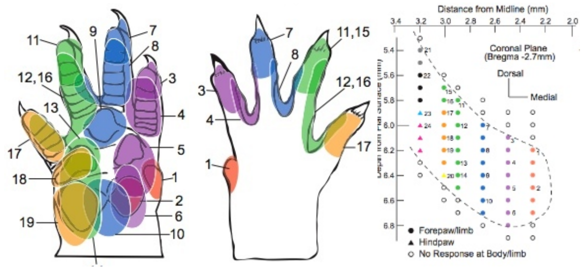


[1]Adaptive Inverse Control (Lin 2012)

Rat data

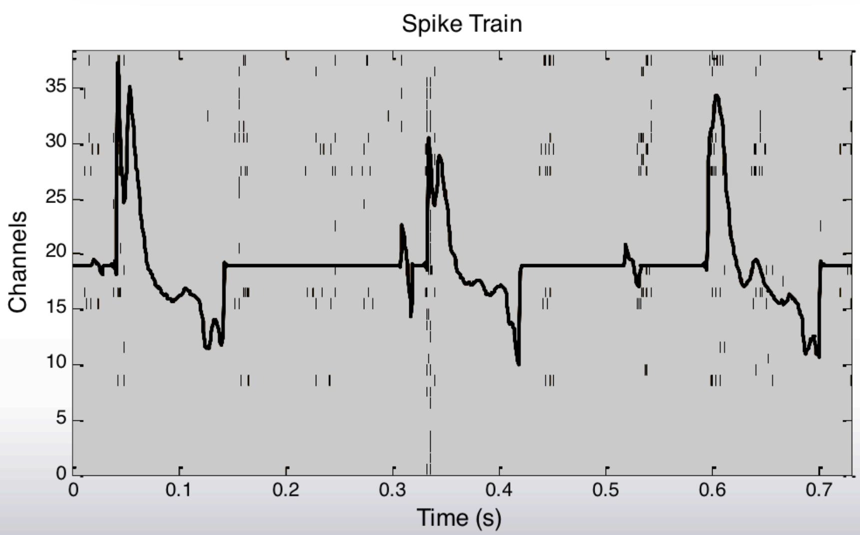




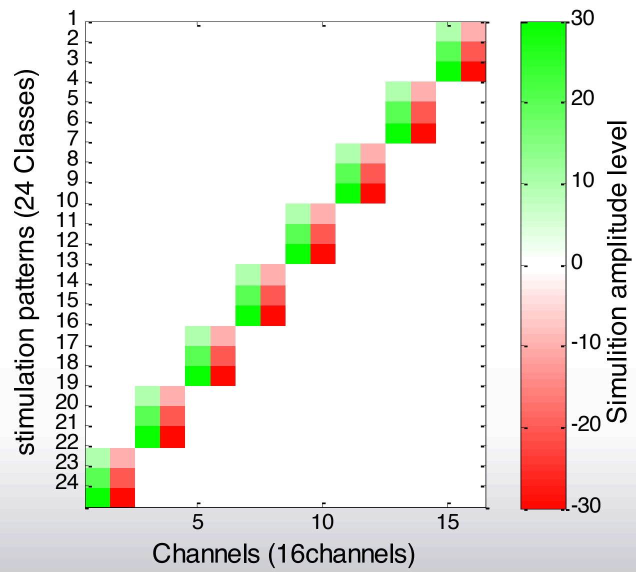


[J. T. Francis 2008]

Tactile Stimulation

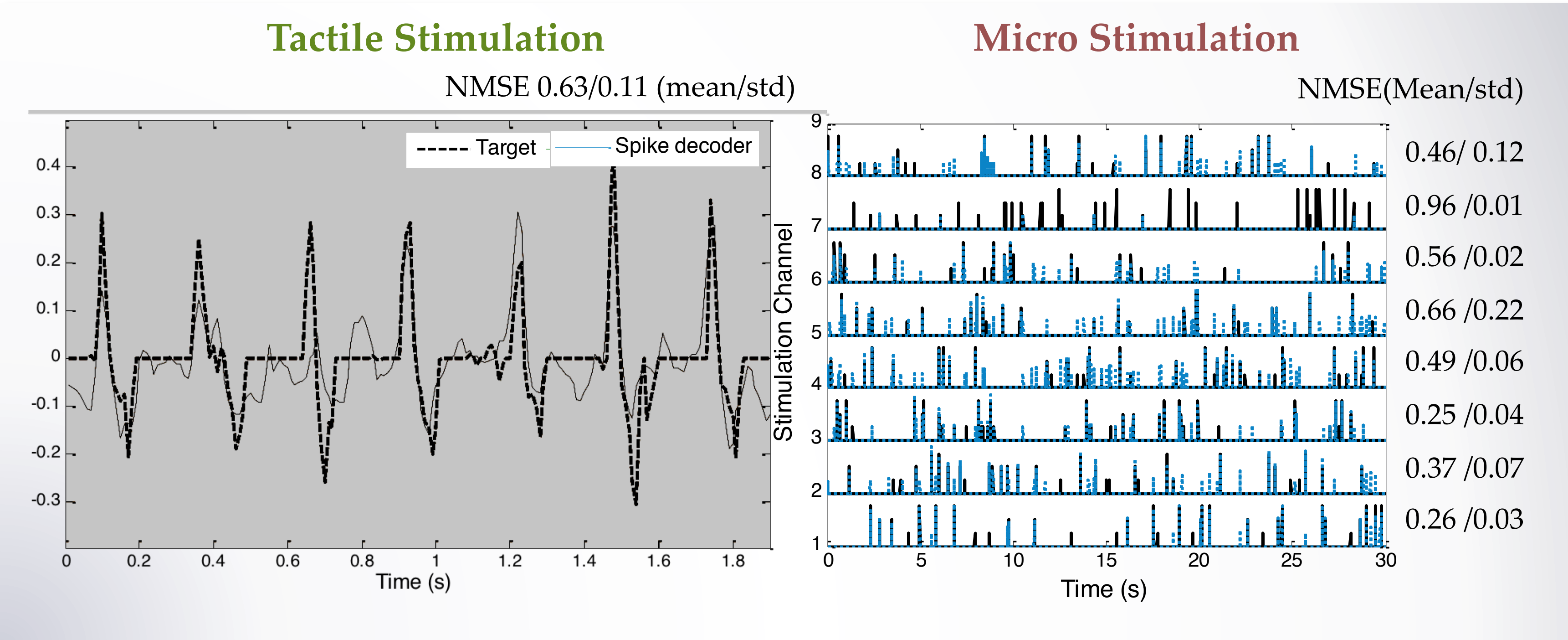


Micro stimulation



Results

- Schoenberg kernel based neural decoder is able to capture the main structure of stimulation.
- Variability of spike train causes the fluctuation of the model output.
- Burst and silence of spike train are unrelated to the stimulation.



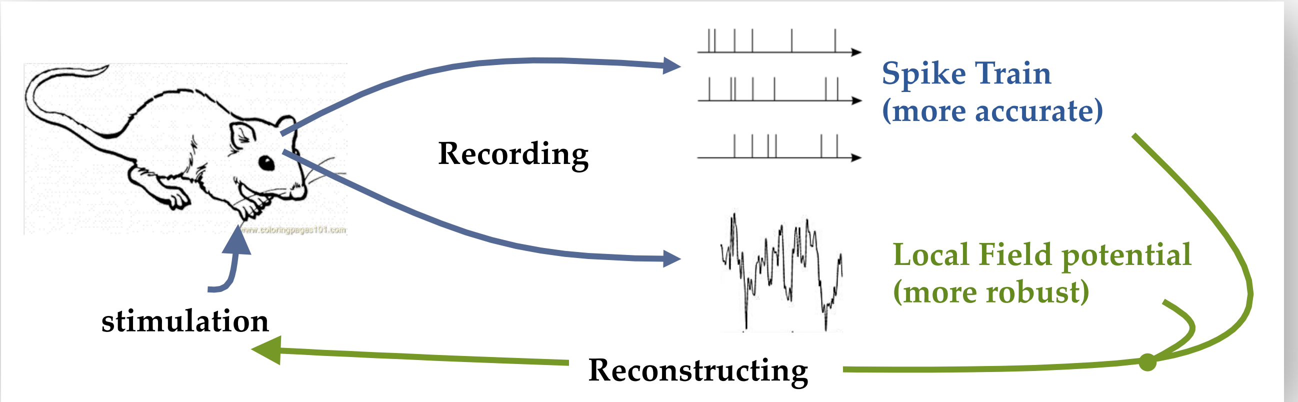
Outline

- Kernel based Framework for **Spike Trains**
- Kernel based Framework for **Multi-scale Neural Activity**
- Conclusion

Li L., Brockmeier A., Choi J., Francis J., Sanchez J., Prıncipe J., “A tensor product kernel framework for multiscale neural activity decoding and control”, Computational Intelligence and NeuroScience, vol. 2014.

Multi-scale data

- **Reasoning**
  - Data from multiple sources contain **complementary information.** Spike train, LFP, ECG, EEG



Our goal:

Effectively combine the complementary information from multiple heterogeneous data sources to enhance the modeling accuracy.

- **Challenge**
  - **Different data types** (example: point process data & amplitude data)
  - **Multiple temporal scales.**

Multi-scale data – Kernel to the rescue

- **Kernels are very flexible functions**
  - Can define a kernel for LFPs and a different kernel for spike trains
  - There are two basic ways to construct multivariate kernels:
    - Direct sum kernels
    - Tensor product kernels (preserves universality)
  - For the sum kernel the joint similarity over a set of dimensions is

$$\kappa_{\Sigma}\left(\mathbf{x}, \mathbf{x}'\right) = \sum_{i \in \mathcal{I}} \kappa_i \left(x_{(i)}, x'_{(i)}\right).$$

  - The contributions over each dimension are diluted, what can be useful when there is high variability.
  - For the tensor product, compute by the product between kernel evaluations.

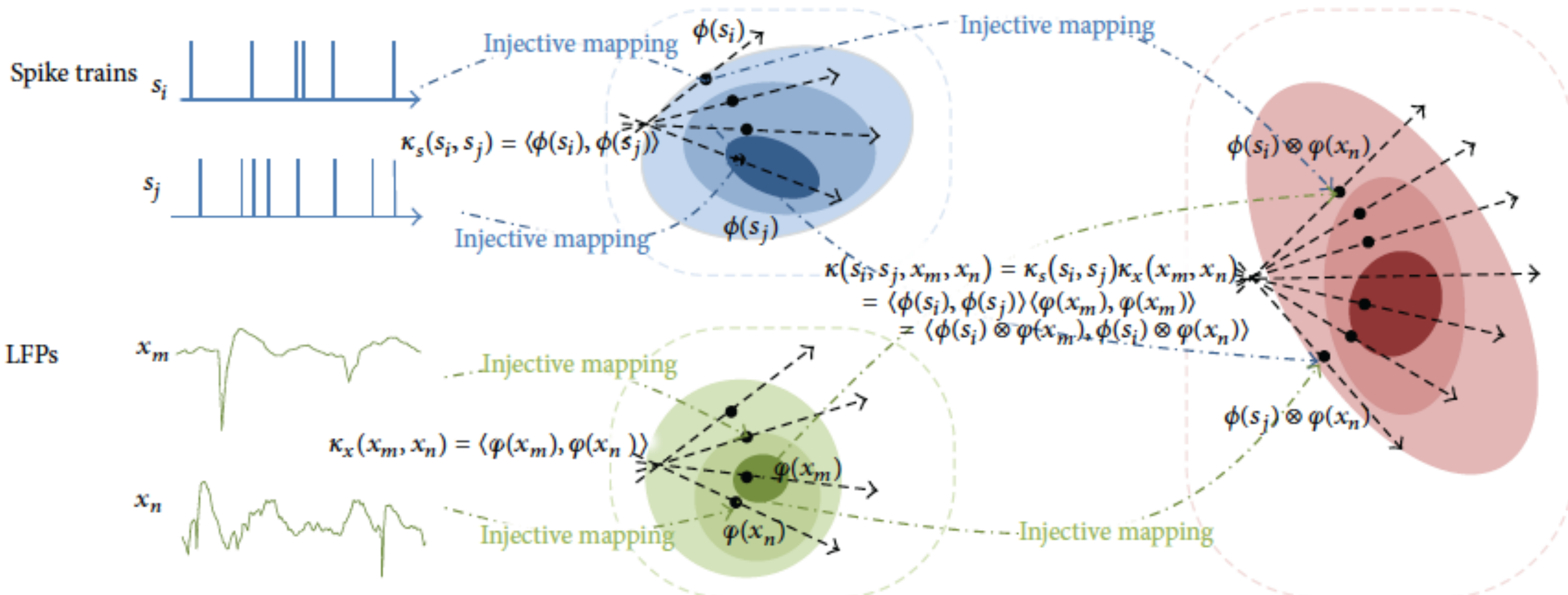
$$\kappa_{[i,j]}([x_{(i)}, x_{(j)}], [x'_{(i)}, x'_{(j)}]) = \kappa_i(x_{(i)}, x'_{(i)}) \cdot \kappa_j(x_{(j)}, x'_{(j)})$$

$$\kappa_{\Pi}(\mathbf{x}, \mathbf{x}') = \prod_{i \in \mathcal{I}} \kappa_i(x_{(i)}, x'_{(i)})$$

  - The tensor product corresponds to a stricter measure of similarity (if one dimension ~0 tensor product ~0)

Multi-scale data – Kernel to the rescue

- **Explaining the tensor product Hilbert space**

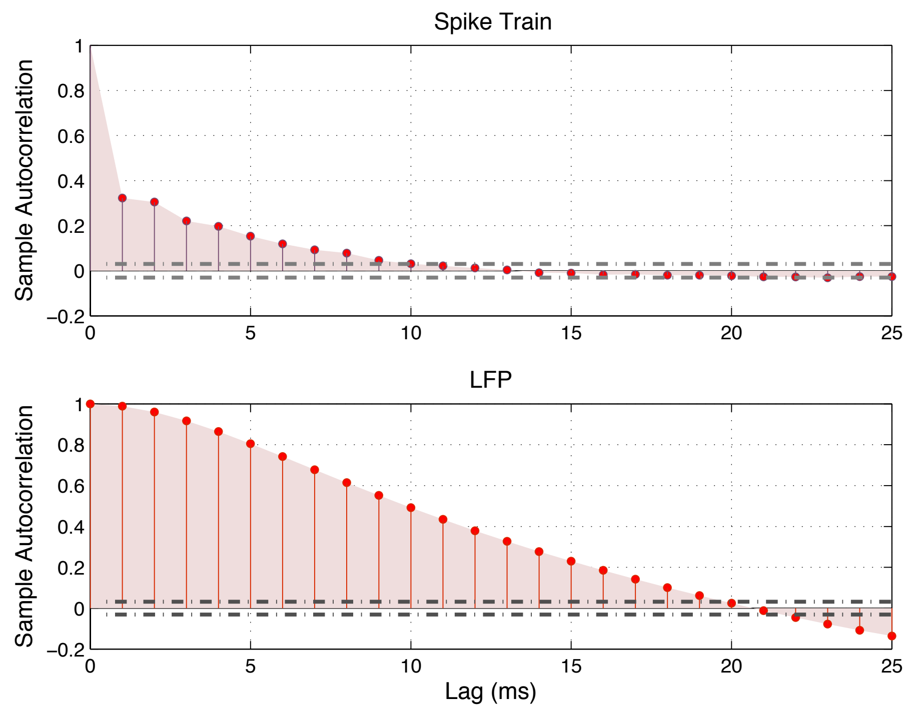


Multi-scale data – Kernel to the rescue

- **Kernel for spikes:** Schoenberg kernel
- **Kernel for LFPs:** Since LFP are time signals, will also use a Schoenberg kernel (Euclidean distance), but time scales will have to be properly defined (sample autocorrelation)

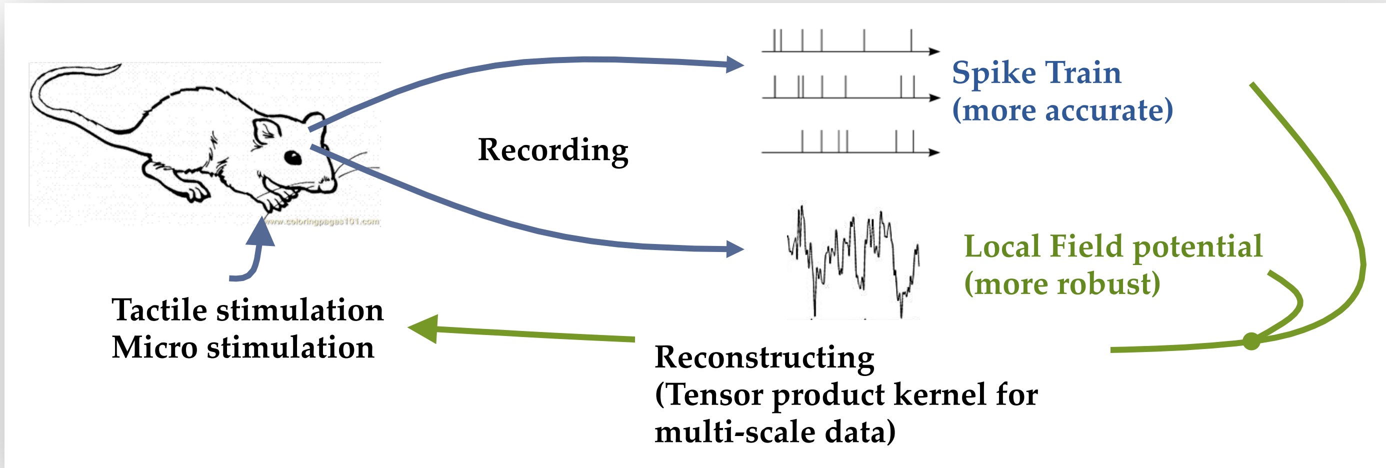
$$\kappa_x(x_i(t), x_j(t)) = \exp(-\frac{\|x_i(t) - x_j(t)\|^2}{\sigma_x^2})$$

$$\kappa_x(\mathbf{x}_i(t), \mathbf{x}_j(t)) = \sum_{n=1}^{N} \kappa_x(x_i^n(t), x_j^n(t))$$

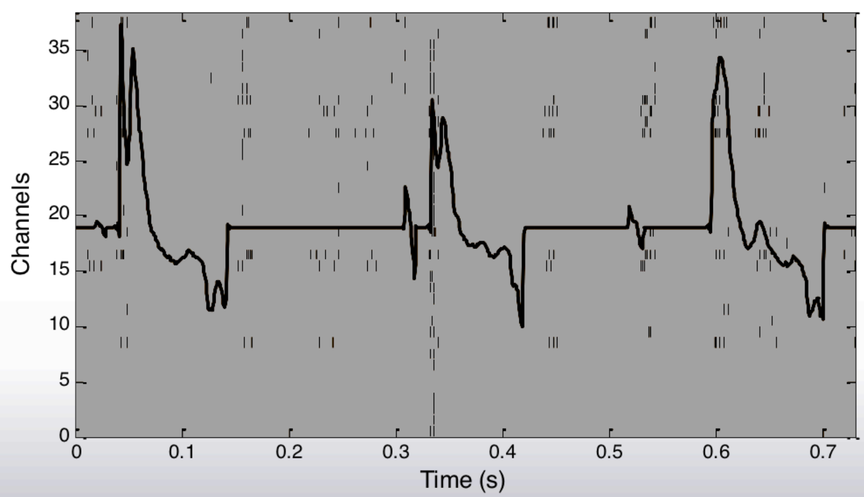


- The beauty is that the Kernel regression is the same algorithm as above

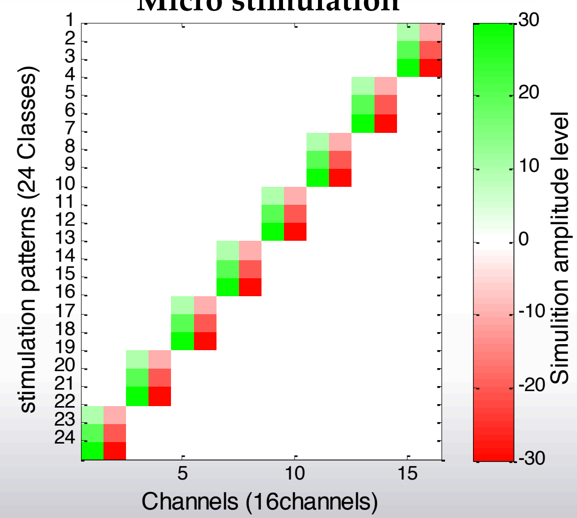
Rat data



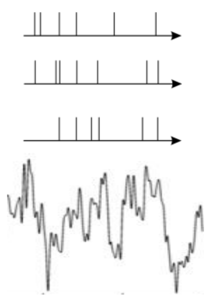
Tactile stimulation



Micro stimulation

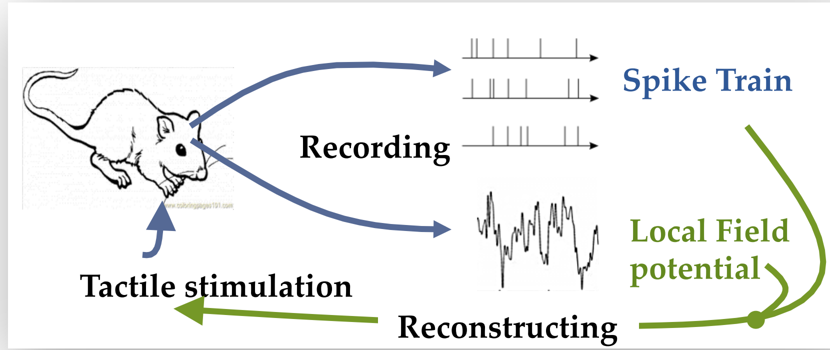


Evaluation



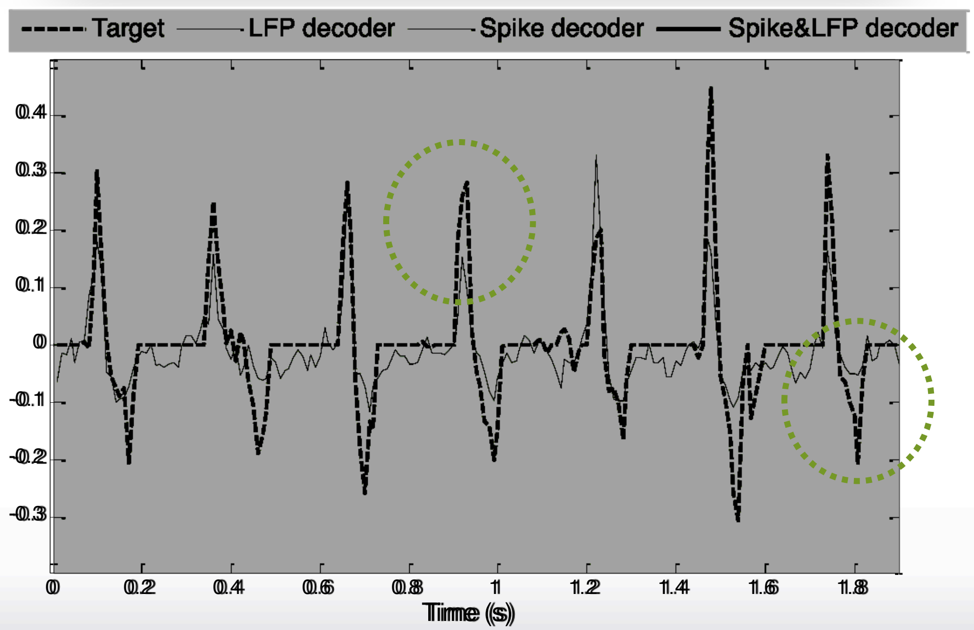
Spike Train:
more accurate

Local Field potential:
more robust



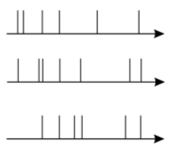
Input	NMSE (mean/ STD)
LFP	**0.55/0.03**
Spike	0.63/0.11
LFP & spike	0.48/0.05

Train data 20 s Test data 2 s
8 different trials

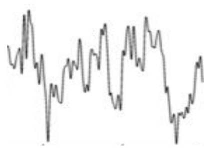


[1]Tensor product kernel (Lin 2012)

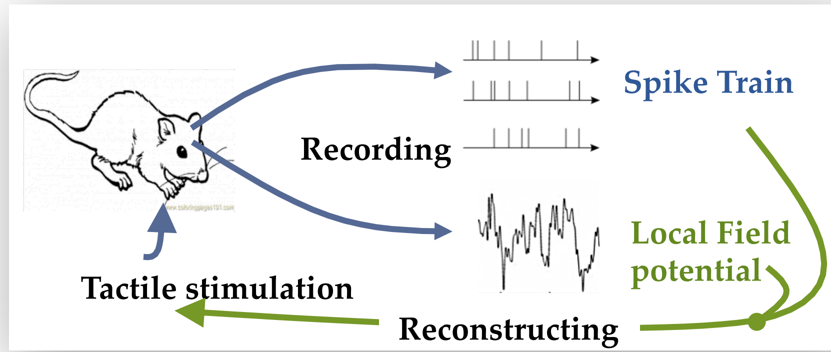
Evaluation



Spike Train:
more accurate

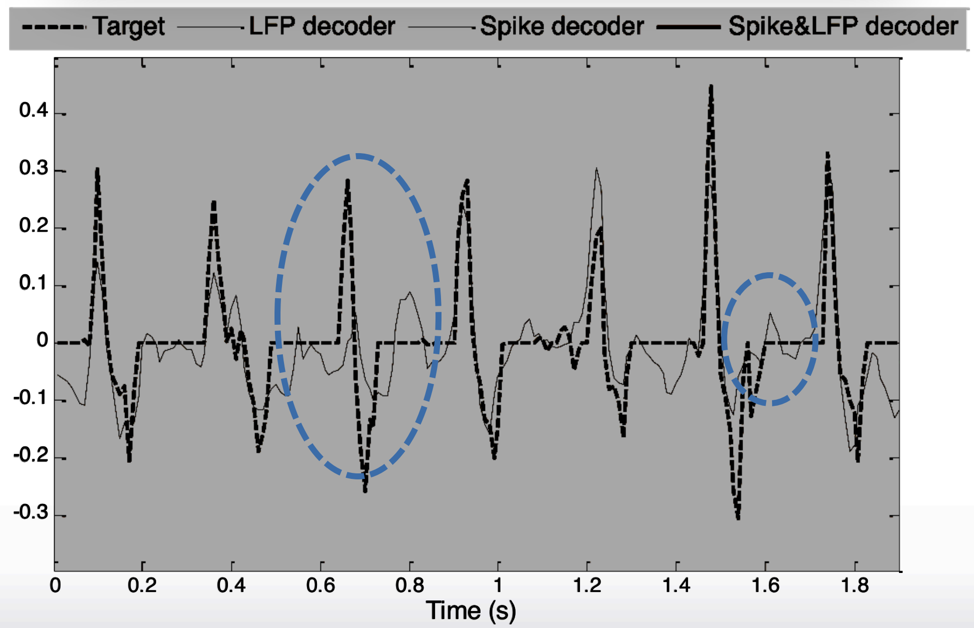


Local Field potential:
more robust



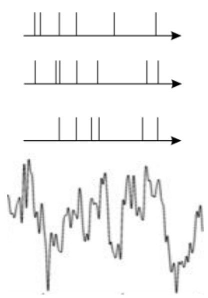
Input	NMSE (mean/ STD)
LFP	0.35/0.03
Spike	**0.63/0.11**
LFP & spike	0.28/0.05

Train data 20 s Test data 2 s
8 different trials



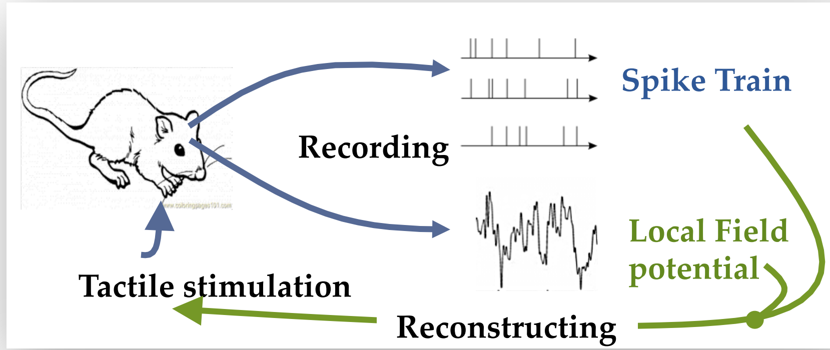
[1]Tensor product kernel (Lin 2012)

Evaluation

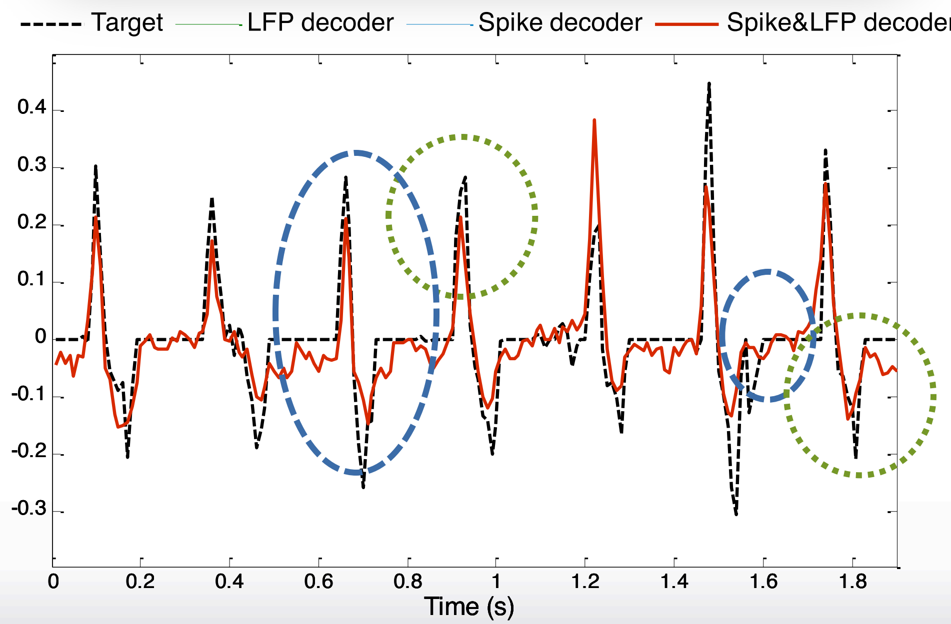


Spike Train: more accurate

Local Field potential: more robust



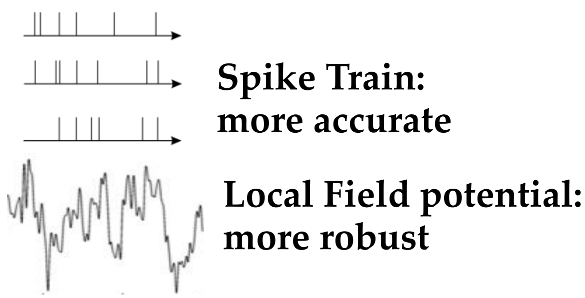
Input	NMSE (mean/ STD)
LFP	0.35/0.03
Spike	0.63/0.11
LFP & spike	**0.28/0.05**

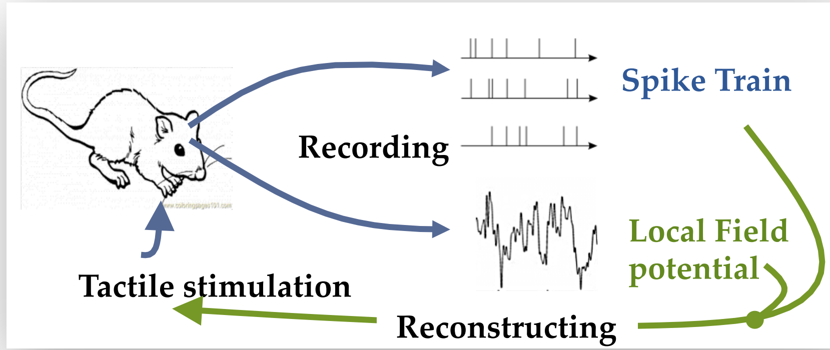


Train data 20 s Test data 2 s
8 different trials

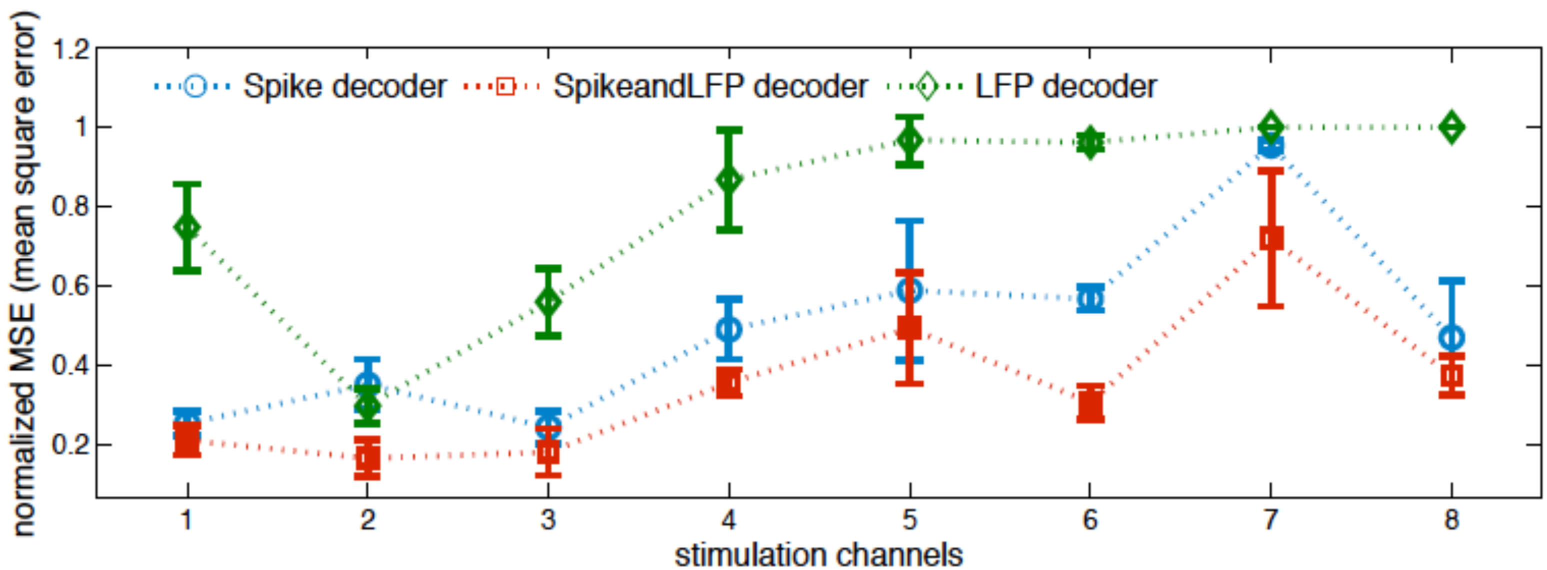
[1]Tensor product kernel (Lin 2012)

Evaluation – Reconstruction of the stim pattern





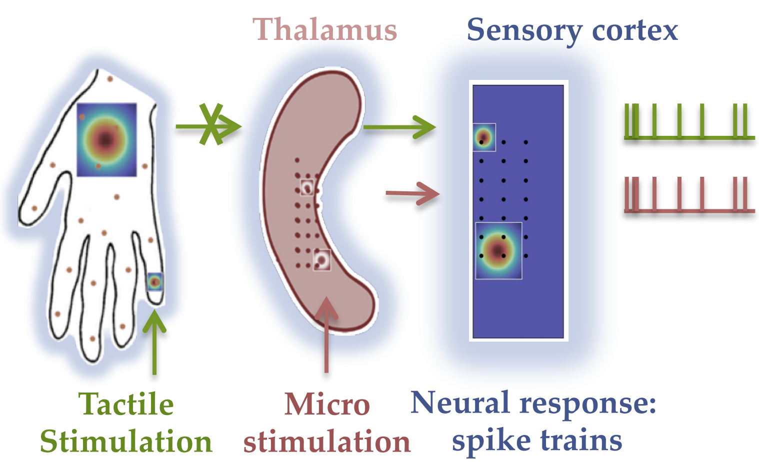
- **Stimulation events are divided in 8 channels, and the neural data is used to predict the occurrence and intensity of each stimulation**

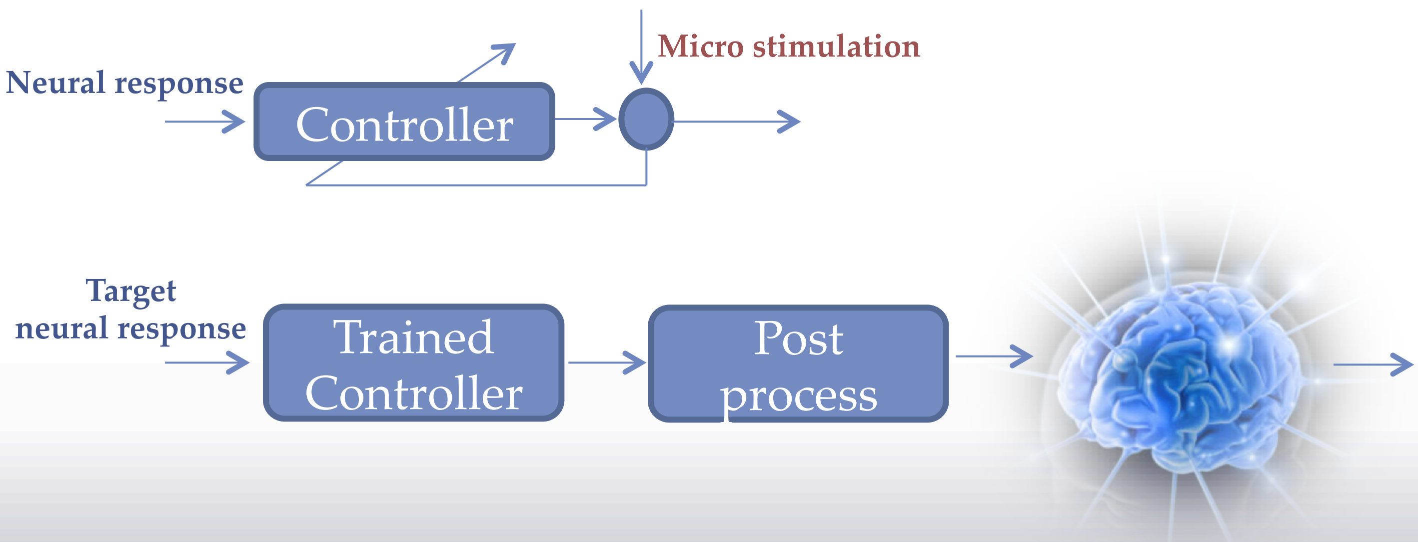


012)

Open loop somatosensory control

- Controller is first trained with Micro stimulation and the corresponding neural response (300 s).
- Then the target pattern (neural activity induced by tactile stimulation in S1) is input to the trained controller



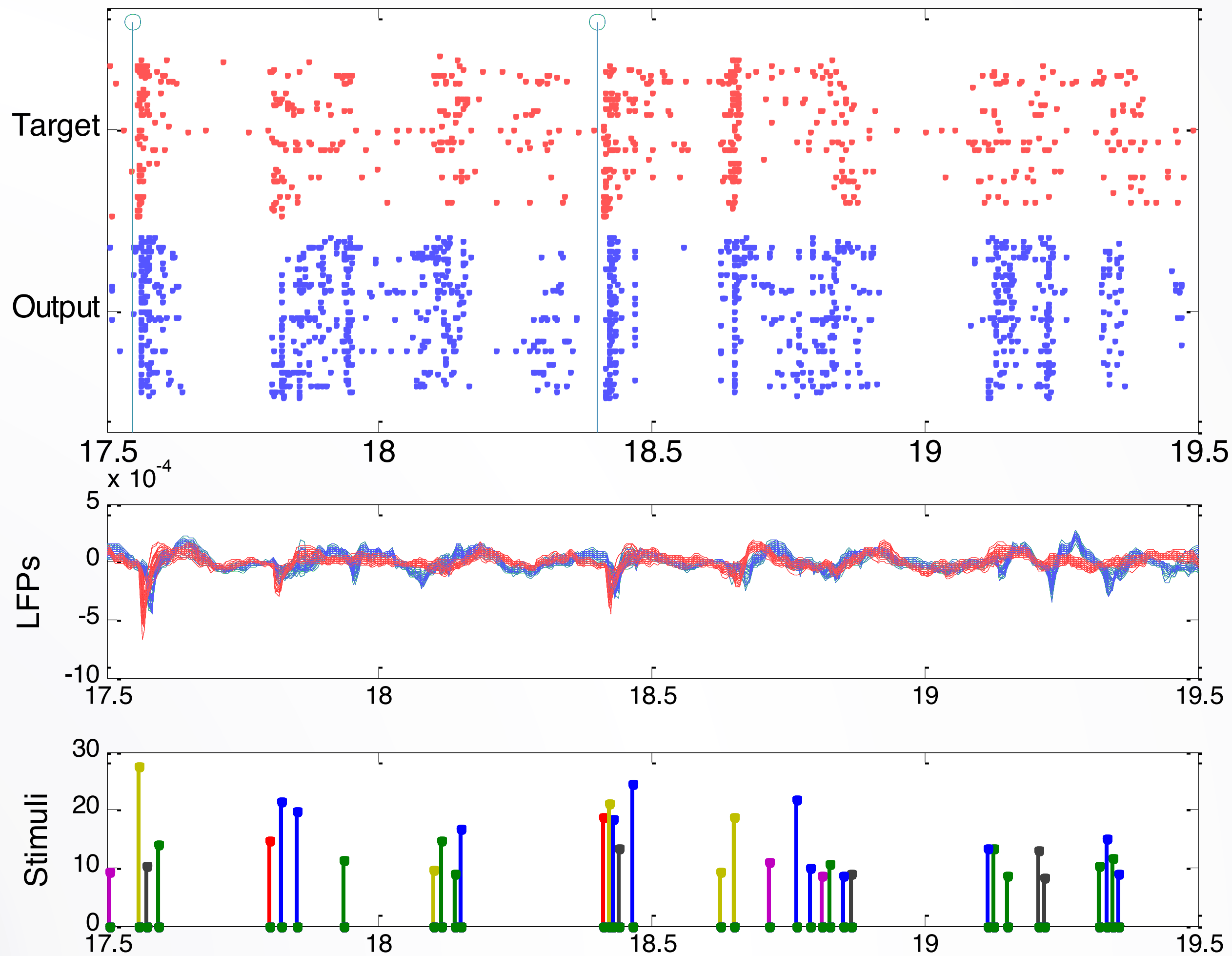


Open loop control results

- The entire sequence is fed offline to the controller producing a multichannel sequence of micro stimulation amplitudes (<u>the virtual touch)</u>
- Virtual touch needs to be formatted for specifications:
    - Minimum interval between stimulations is 10 ms
    - At a given time only the maximum stimulation is applied
    - The min/max stimulations are in the range 8-30 μA
- The generated micro stimulation is applied to the electrical stimulator
- The neural response to the stimulator is recorded and compared to the natural touch response.

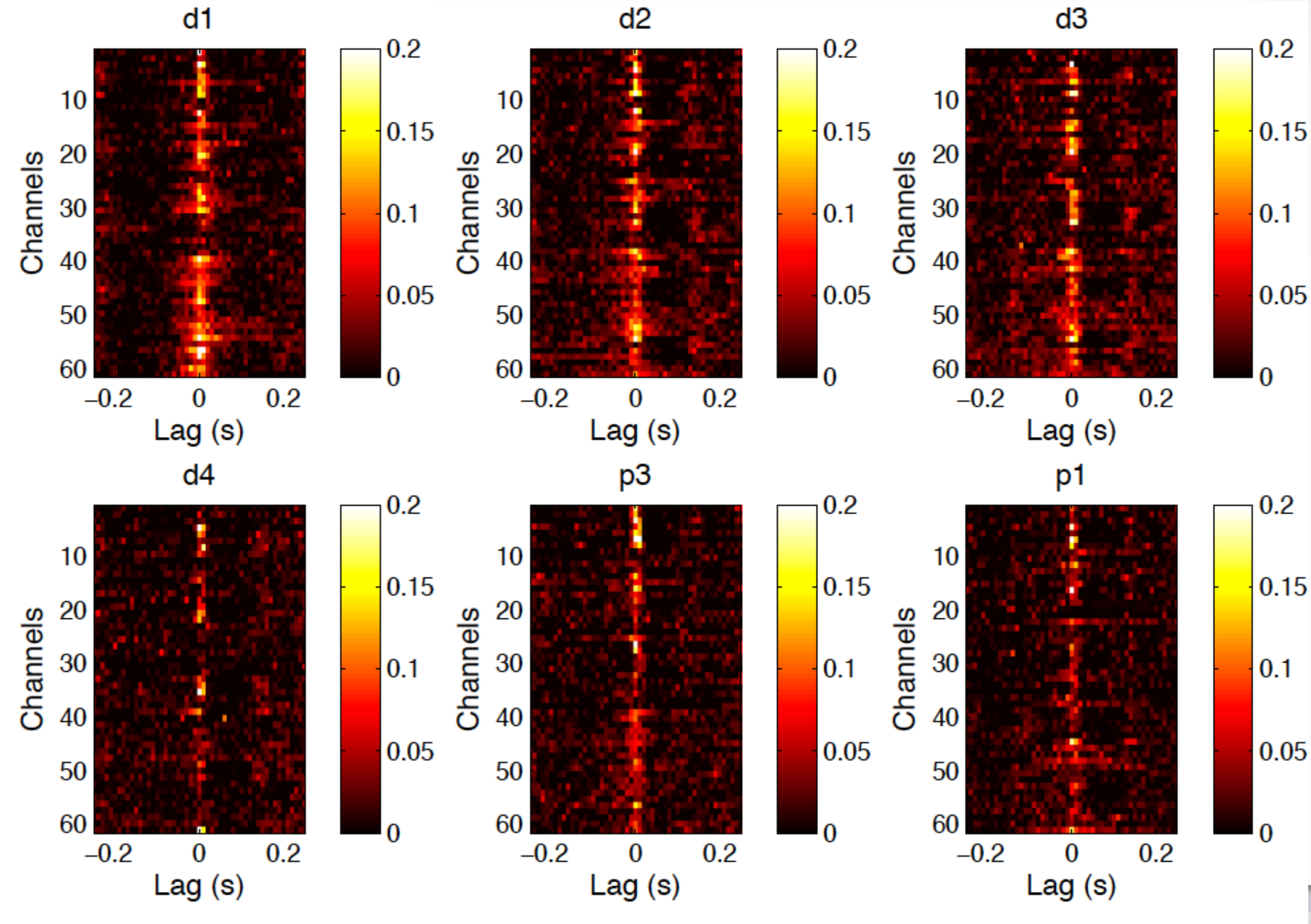
NOTE: they are <u>not</u> concurrently measured, they need to be aligned with the corresponding response in S1 for the tactile stimulation in the same paw area.

Open loop control results --- Controlled state



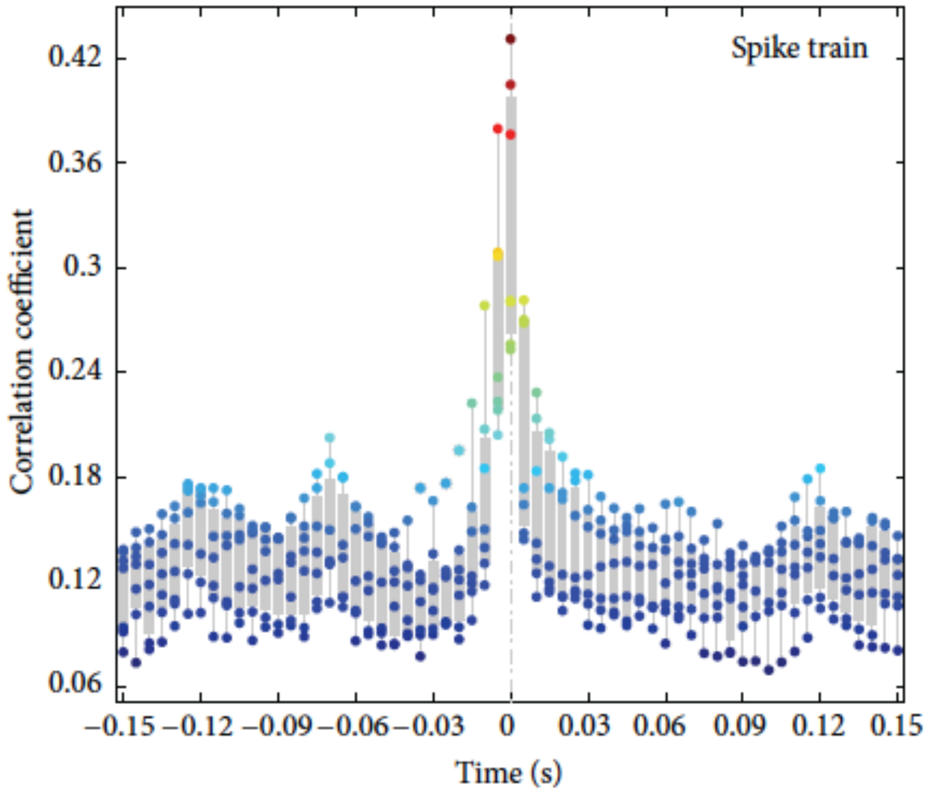
Open loop control results - Summary

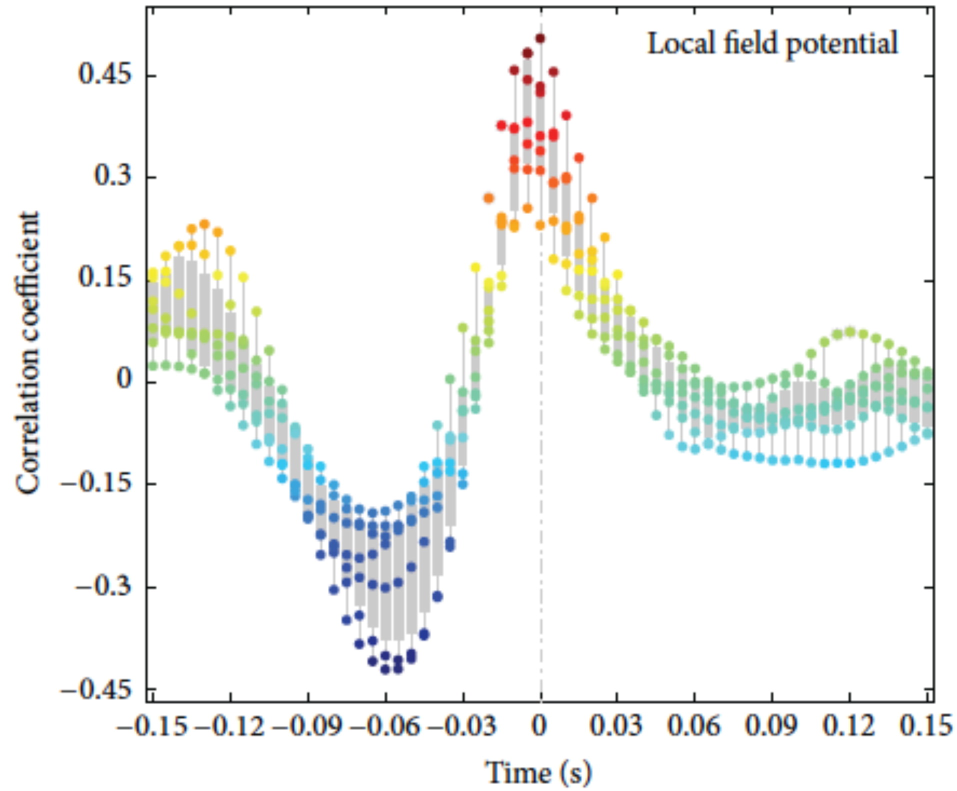
- Cross-correlation between target and system output for each channel



Open loop control results – Touch Timing

- **Box plot of correlation coefficient between target and system output (one per site)**





Open loop control results – Touch Site

- Test of discriminability per touch site (Matched virtual versus unmatched)
- Use 300 ms window after stim, and compared cc between natural and virtual
- Use one tail KS to test the alternative hypothesis that match is higher than unmatched

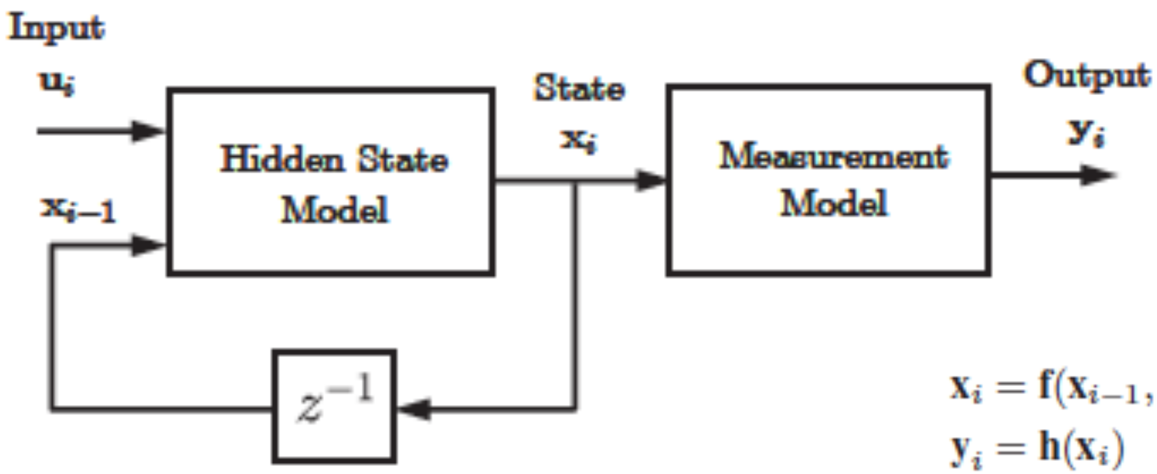
Spike trains

Touch site	CC		
	Matched virtual	Unmatched virtual	*P* value
d1	0.42 ± 0.06	0.35 ± 0.06	0.00
d2	0.40 ± 0.05	0.37 ± 0.06	0.01
d4	0.40 ± 0.05	0.37 ± 0.05	0.02
p3	0.38 ± 0.05	0.37 ± 0.06	0.11
p2	0.40 ± 0.07	0.36 ± 0.05	0.00
mp	0.41 ± 0.07	0.37 ± 0.06	0.00

Local Field Potentials

Touch site	CC		
	Matched virtual	Unmatched virtual	*P* value
d1	0.42 ± 0.20	0.28 ± 0.23	0.00
d2	0.46 ± 0.13	0.28 ± 0.22	0.00
d4	0.41 ± 0.19	0.26 ± 0.21	0.00
p3	0.38 ± 0.18	0.29 ± 0.22	0.07
p2	0.33 ± 0.19	0.26 ± 0.23	0.20
mp	0.34 ± 0.17	0.25 ± 0.21	0.00

State Models in RKHS



$$\mathbf{x}_i = \mathbf{f}(\mathbf{x}_{i-1}, \mathbf{u}_i)$$
$$\mathbf{y}_i = \mathbf{h}(\mathbf{x}_i)$$

where

$$\mathbf{f}(\mathbf{x}_{i-1}, \mathbf{u}_i) \triangleq \left[f^{(1)}(\mathbf{x}_{i-1}, \mathbf{u}_i), \cdots, f^{(n_x)}(\mathbf{x}_{i-1}, \mathbf{u}_i)\right]^T = \left[\mathbf{x}_i^{(1)}, \cdots, \mathbf{x}_i^{(n_x)}\right]^T$$

$$\mathbf{h}(\mathbf{x}_i) \triangleq \left[h^{(1)}(\mathbf{x}_i), \cdots, h^{(n_y)}(\mathbf{x}_i)\right]^T = \left[\mathbf{y}_i^{(1)}, \cdots, \mathbf{y}_i^{(n_y)}\right]^T$$

This hidden state model can not be implemented in RKHS using the representer theorem!
(for translation invariant kernels)

Li Kan, Principe J., “Kernel Adaptive Recursive Filters for Grammatical Inference”, accepted in IEEE Trans. Neural Networks.

State Models in RKHS- Our Approach

Rewrite the dynamical system equations as

$$\mathbf{s}_i \triangleq \begin{bmatrix}\mathbf{x}_i \\ \mathbf{y}_i\end{bmatrix} = \begin{bmatrix}\mathbf{f}(\mathbf{x}_{i-1}, \mathbf{u}_i) \\ \mathbf{h}\circ\mathbf{f}(\mathbf{x}_{i-1}, \mathbf{u}_i)\end{bmatrix}$$

$$\mathbf{y}_i = \mathbf{s}_i^{(n_s - n_y + 1 : n_s)} = \begin{bmatrix}0 & \mathbf{I}_{n_y}\end{bmatrix}\begin{bmatrix}\mathbf{x}_i \\ \mathbf{y}_i\end{bmatrix}$$

$$\mathbf{g}(\mathbf{s}_{i-1}, \mathbf{u}_i) = \mathbf{f}(\mathbf{x}_{i-1}, \mathbf{u}_i)$$

$$\mathbf{x}_i = \mathbf{g}(\mathbf{s}_{i-1}, \mathbf{u}_i)$$
$$\mathbf{y}_i = \mathbf{h}(\mathbf{x}_i) = \mathbf{h}\circ\mathbf{g}(\mathbf{s}_{i-1}, \mathbf{u}_i).$$

Map the augmented state s(.) and u(.) to two separate RKHS and then create a product kernel $\mathcal{H}_{su} \triangleq \mathcal{H}_s \otimes \mathcal{H}_u$ (tensor product)

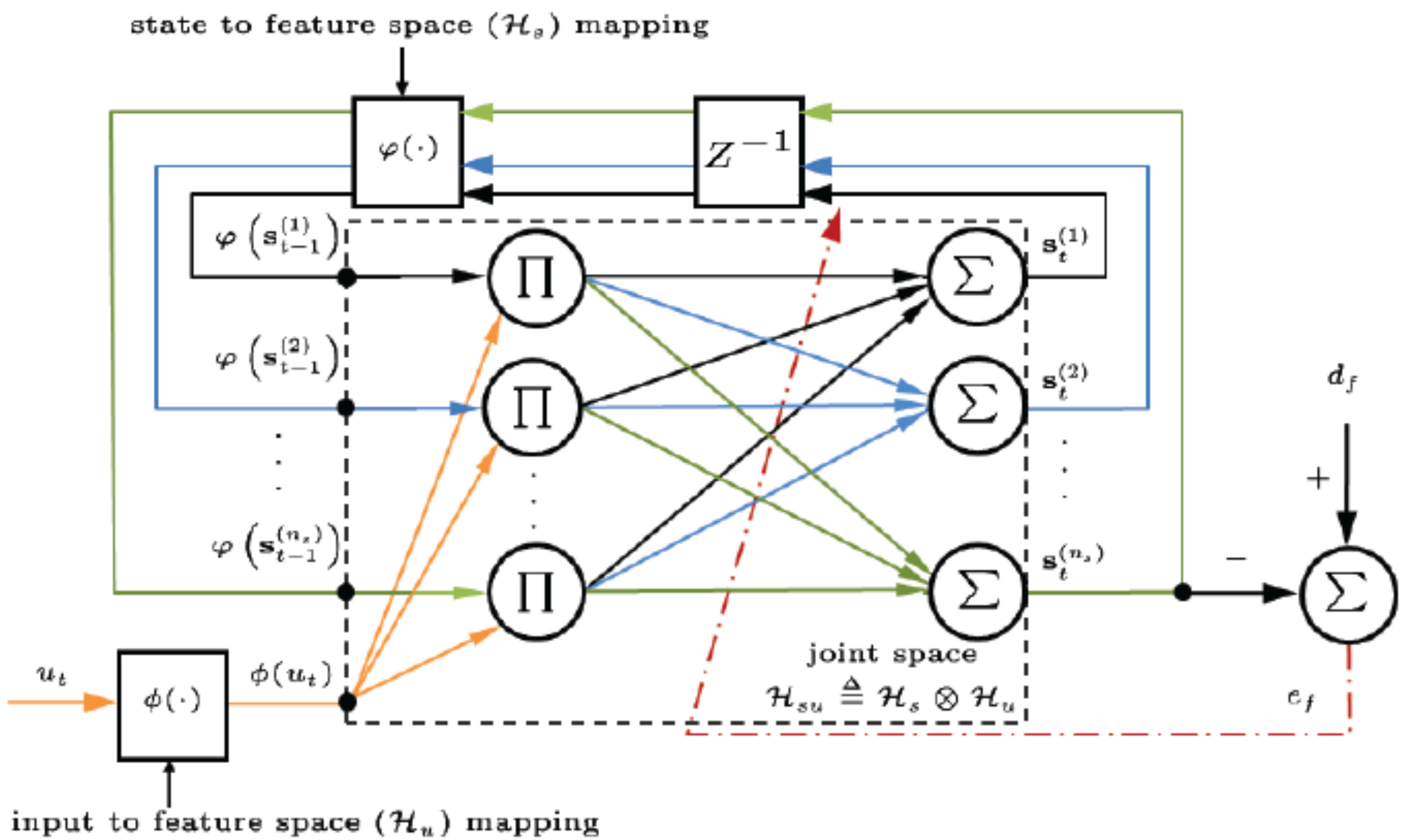
$$\mathbf{\Omega} \triangleq \mathbf{\Omega}_{\mathcal{H}_{su}} \triangleq \begin{bmatrix}\mathbf{g}(\cdot,\cdot) \\ \mathbf{h}\circ\mathbf{g}(\cdot,\cdot)\end{bmatrix}$$

$$\psi(\mathbf{s}_{i-1}, \mathbf{u}_i) \triangleq \varphi(\mathbf{s}_{i-1}) \otimes \phi(\mathbf{u}_i) \in \mathcal{H}_{su}.$$

$$\mathbf{s}_i = \mathbf{\Omega}^T \psi(\mathbf{s}_{i-1}, \mathbf{u}_i)$$
$$\mathbf{y}_i = \mathbf{W}_m \mathbf{s}_i.$$

$$\begin{aligned}\langle \psi(\mathbf{s},\mathbf{u}), \psi(\mathbf{s}',\mathbf{u}')\rangle_{\mathcal{H}_{su}} &= \mathcal{K}_{su}(\mathbf{s},\mathbf{u},\mathbf{s}',\mathbf{u}') \\ &= (\mathcal{K}_s \otimes \mathcal{K}_u)(\mathbf{s},\mathbf{u},\mathbf{s}',\mathbf{u}') \\ &= \mathcal{K}_s(\mathbf{s},\mathbf{s}') \cdot \mathcal{K}_u(\mathbf{u},\mathbf{u}').\end{aligned}$$

State Models in RKHS



Parameters can be trained with Real time Recurrent Learning

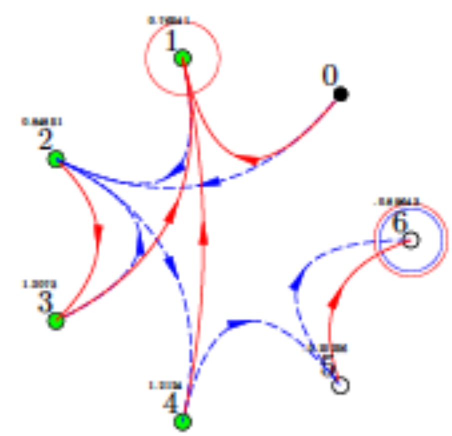
Results in Tomita Grammars

No.	Description
1	1*
2	(10)*
3	No odd number of consecutive 0's after an odd number of consecutive 1's.
4	Any string with fewer than three consecutive 0's.
5	Any even length string with an even number of 1's.
6	Difference b/w number of 1's and 0's is a multiple of 3.
7	0*1*0*1*

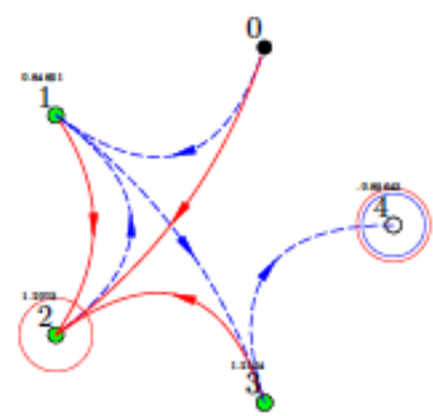
TABLE II: QKARF DFA for Tomita grammars.

Grammar	QKARF size	Extract. DFA size	Min. DFA size
#1	20	4	3
#2	22	6	4
#3	46	8	6
#4	28	7	5
#5	34	5	5
#6	28	5	4
#7	36	8	6

Extracted Automaton

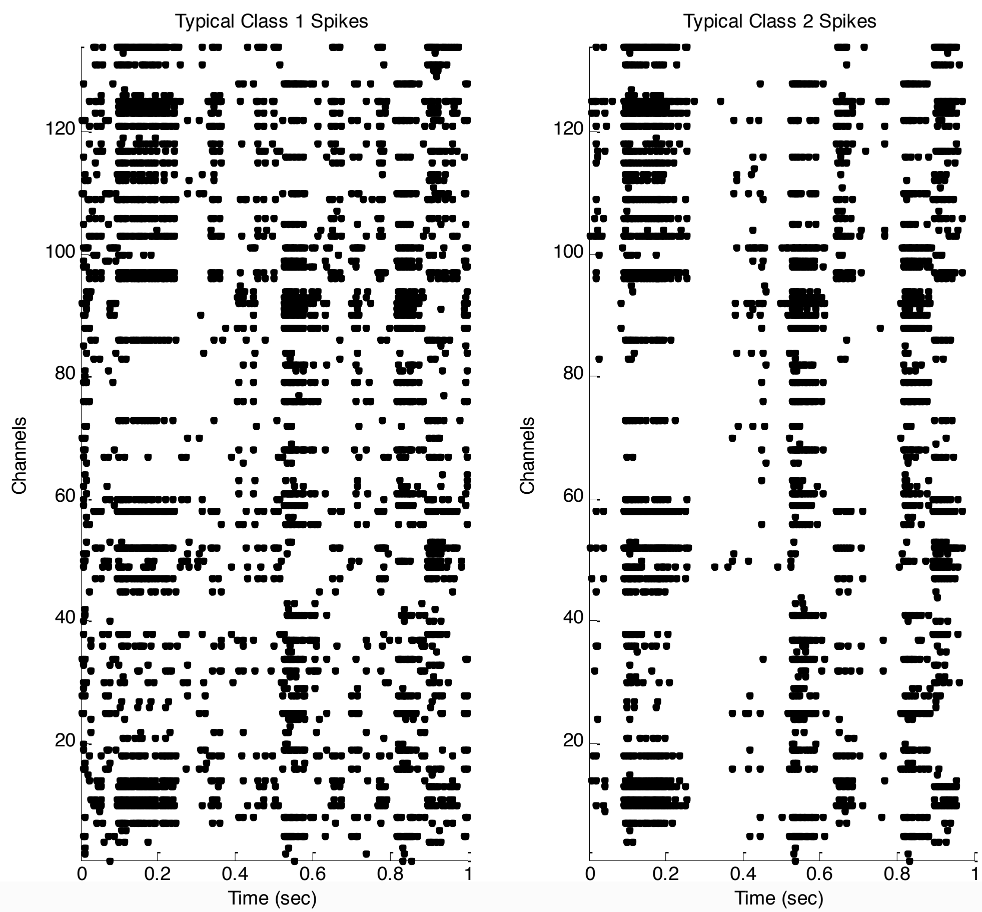


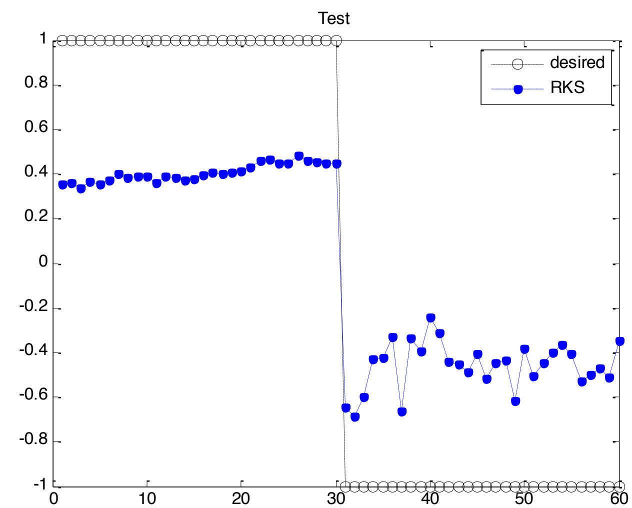
Minimized DFA



Tomita
#4

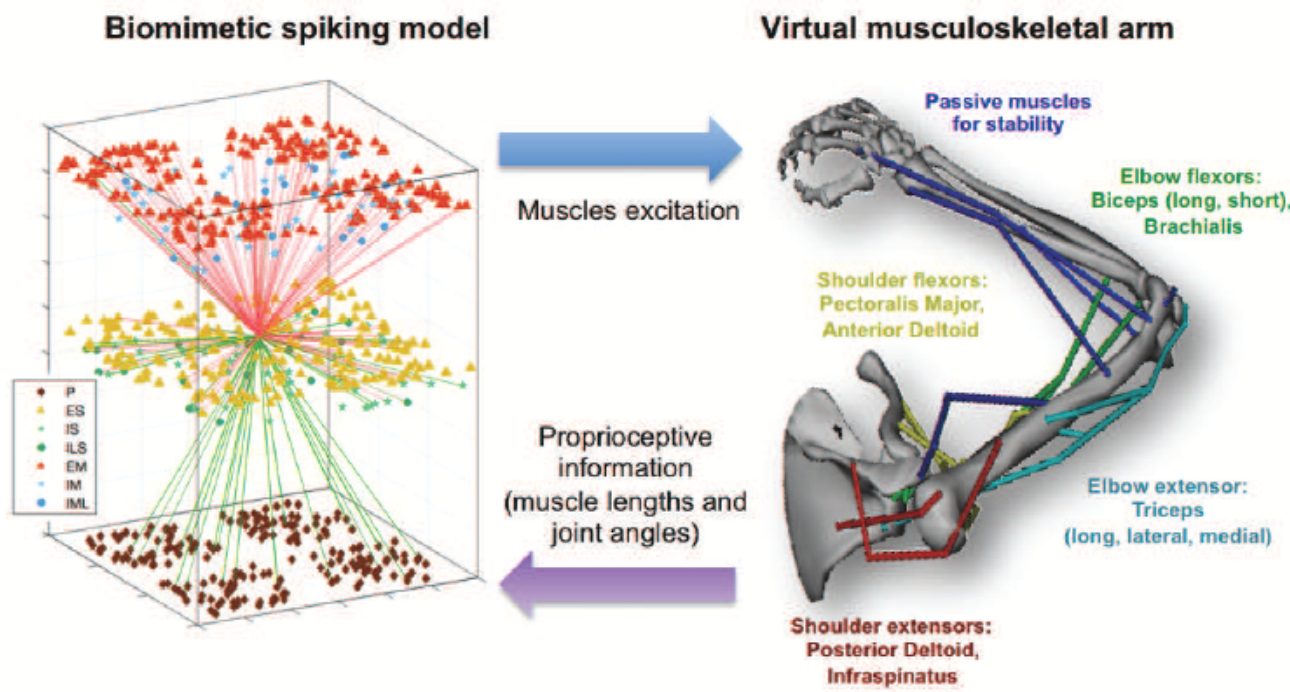
Distinguish pulse trains with KAARMA





Examples of the two classes of spikes generated using the same stimulation sequences with additive normal noise (left) and uniform noise (right) of the same power.

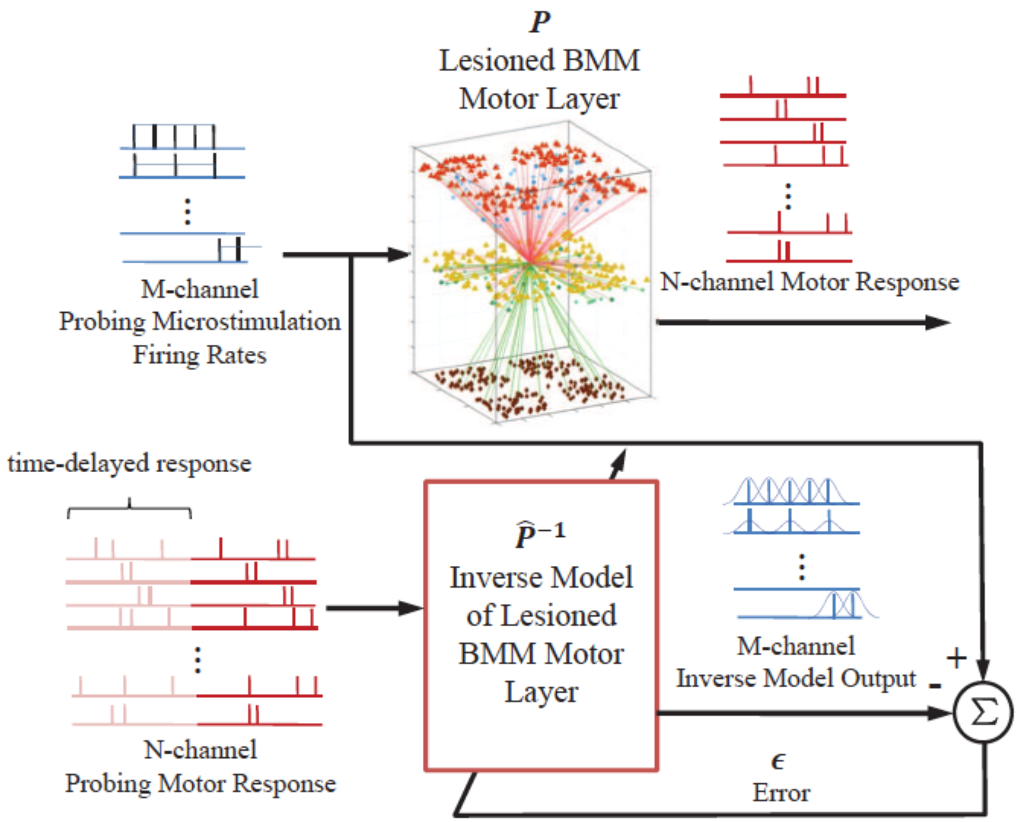
Repairing Brain Lesions with Stimulation

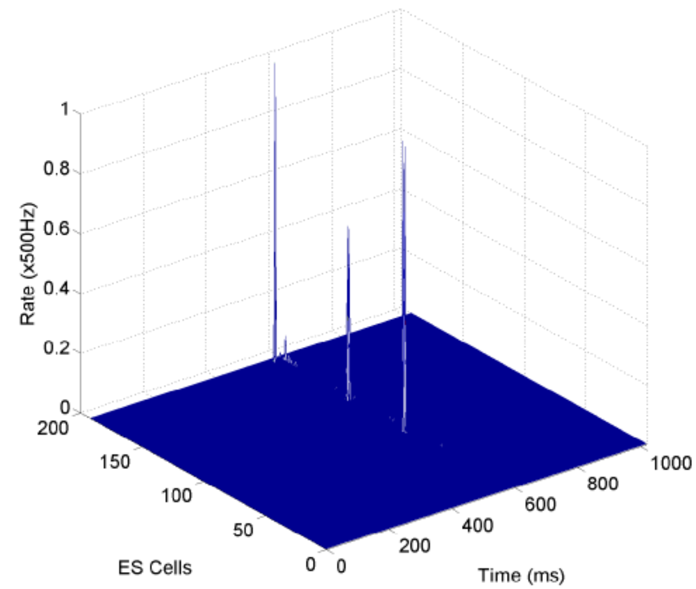


The biomimetic spiking models are built in NEURON using the architecture of the motor cortex (500 neurons), and trained with spike timing dependent reinforcement learning. 10% of the cells are then silenced to mimic a lesion. The other neurons are then probed to obtain a model P(z) of the transfer function from the lesioned M1 to arm movement (plant model).

Li K., Dura S., Francis J., Lytton B., Principe J., "Repairing Lesions Via Kernel Adaptive Inverse Control in a Biomimetic Model of Sensorimotor Cortex", accepted IEEE Neural Eng Workshop, Montepellier, 2015

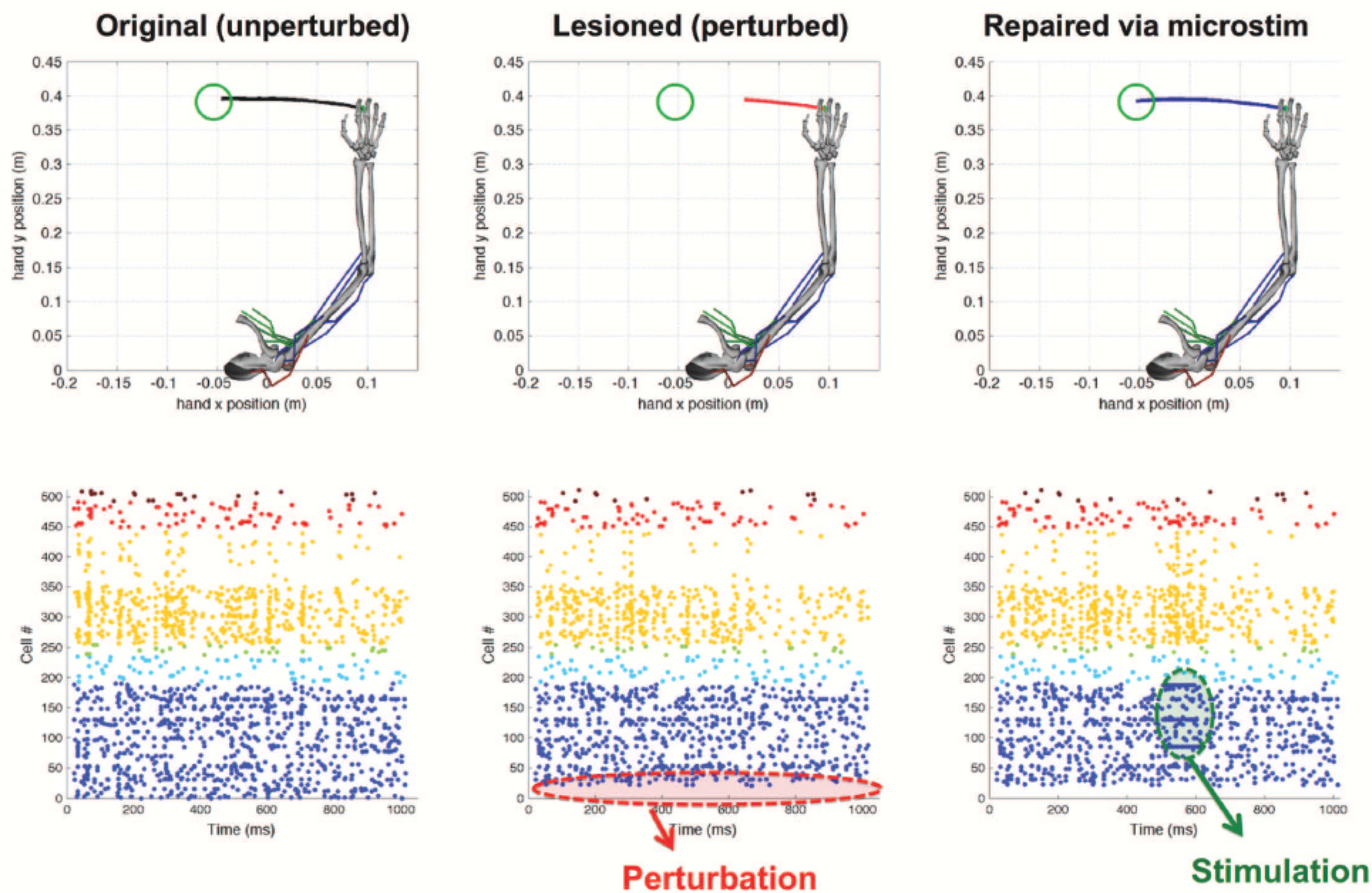
Repairing Brain Lesions with Stimulation





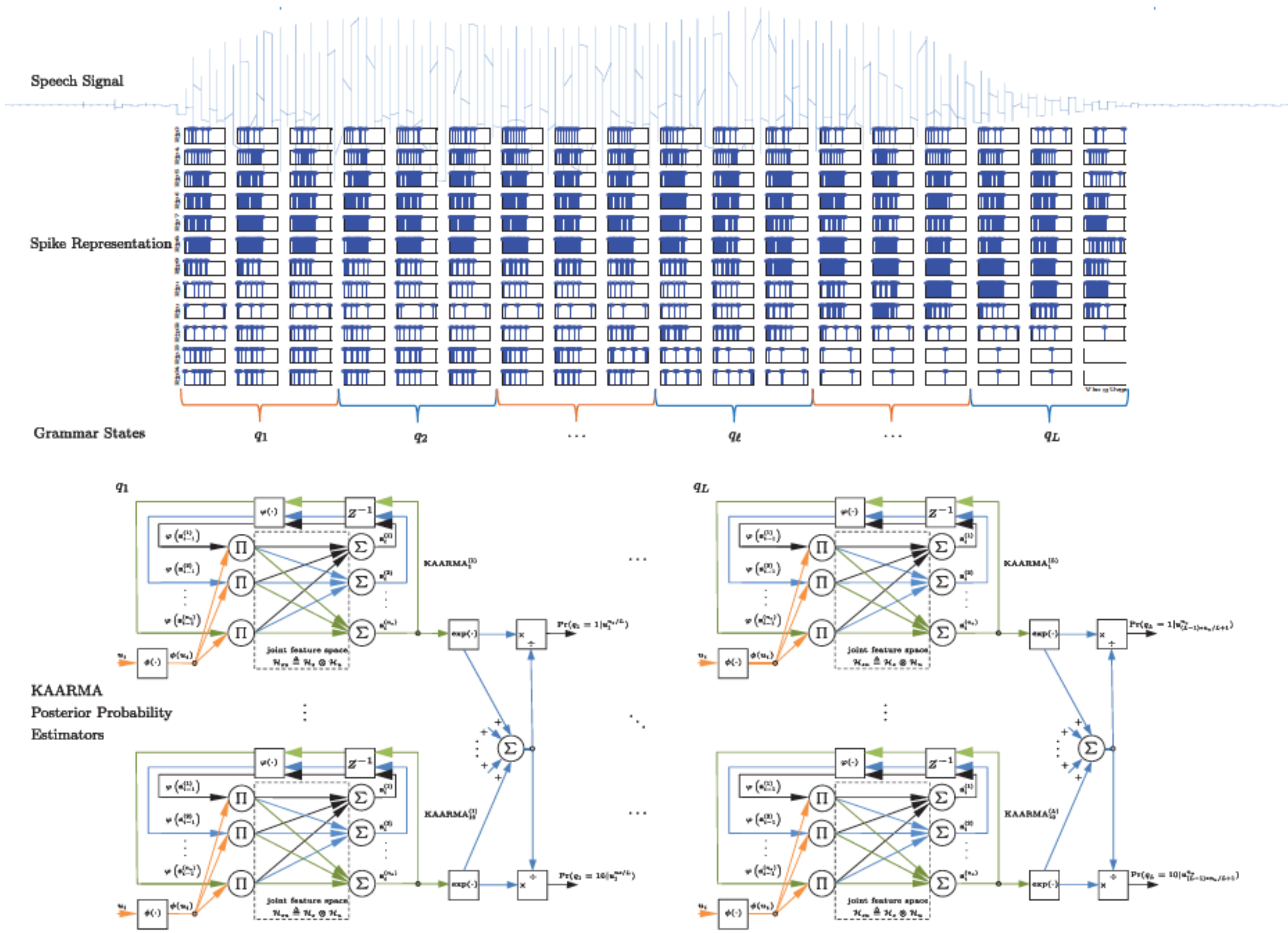
We use a KAARMA to implement the inverse model of the kinematics and using the normal spike train response (AIC) we can find out the best correction (electrical stimulation) to compensate for the handicap. Notice that the stimulation is very local in time as the right figure shows.

Repairing Brain Lesions with Stimulation



Micro stimulation is capable of correcting the movement choosing the time, the channels and the pattern for best results (done in Neuron simulator).

Isolated Digit Recognition with Spike Trains



KAARMA (98.64%) outperformed the HMM (98%) in the Ti 46 speech database

Conclusion

- Neural systems are difficult to model with traditional approaches.
- We showed how a functional analysis approach is able to provide an efficient way to process spikes trains in a function space (RKHS) and also integrate this information with LFPs collected from the same electrodes.
- This multiscale approach was shown to improve the performance of a controller that stimulates VPL to produce brain activity in S1 that is similar to the one obtained by stimulating the subjects forepaw.
- The method still needs further refinements to produce a practical controller for sensorimotor stimulation, but the integration of multi-scale brain activity with kernels is very promising.
- Methodology can be used in many different neurotech problems