Honda Research Institute EU
Honda Research Institute Europe GmbH
https://www.honda-ri.de/

Modeling Human Motion Trajectories by Sparse Activation of Motion Primitives Learned from Unpartitioned Data

Christian Vollmer, Julian Eggert, Horst-Michael Groß

2012

Preprint:

This is an accepted article published in German Conf. on Artificial Intelligence (KI). The final authenticated version is available online at: https://doi.org/[DOI not available]

Modeling Human Motion Trajectories by Sparse Activation of Motion Primitives Learned from Unpartitioned Data

Christian Vollmer[1], Julian P. Eggert[2], and Horst-Michael Gross[1]

[1]Ilmenau University of Technology,
Neuroinformatics and Cognitive Robotics Lab,
98684 Ilmenau, Germany
christian.vollmer@tu-ilmenau.de
[2]Honda Research Institute Europe GmbH
63073 Offenbach/Main, Germany
julian.eggert@honda-ri.de

Abstract. We interpret biological motion trajectories as being composed of sequences of sub-blocks or *motion primitives*. Such primitives, together with the information, *when* they occur during an observed trajectory, provide a compact representation of movement in terms of events that is invariant to temporal shifts. Based on this representation, we present a model for the generation of motion trajectories that consists of two layers. In the lower layer, a trajectory is generated by activating a number of motion primitives from a learned dictionary, according to a given set of activation times and amplitudes. In the upper layer, the process generating the activation times is modeled by a group of Integrate-and-Fire neurons that emits spikes, dependent on a given class of trajectories, that activate the motion primitives in the lower layer. We learn the motion primitives together with their activation times and amplitudes in an unsupervised manner from unpartitioned data, with a variant of shift-NMF that is extended to support the event-like encoding. We present our model on the generation of handwritten character trajectories and show that we can generate good reconstructions of characters with shared primitives for all characters modeled.

Keywords: sparse coding, non-negative matrix factorization, motion primitives, spiking neurons

1 Introduction

Studies in animal motor control suggest that the motor system consists of a control hierarchy, where a number of low-level motor primitives control muscle activations to perform small movements and a higher level controls the sequential activation of those motor primitives to perform complex movements [1]. In addition, motor primitives are shared amongst high-level motions.

We present a model for the generation of motion trajectories that is inspired from those results. We demonstrate our model on the generation of handwritten character velocity trajectories (see Fig. 1).

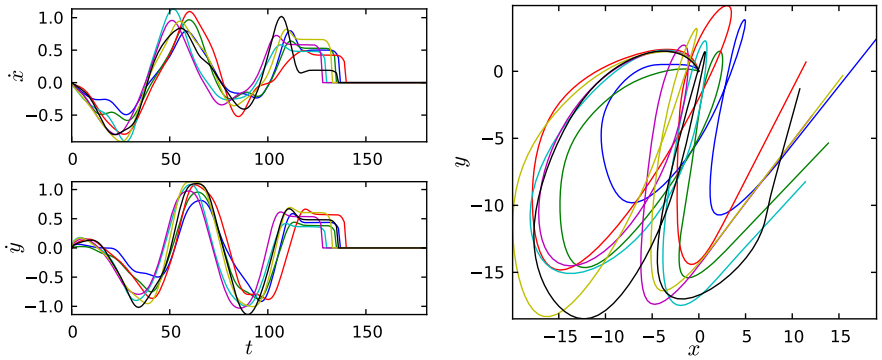

Fig. 1: Ten example handwritten character trajectories of the letter 'a' of our dataset, plotted as velocity space over time (left) and in pen space (right). Throughout this paper, training of the models and generation of trajectories is done in velocity space. The pen space representation is obtained by integrating the velocity over time. The pen space is shown for an intuitive visualization.

We model a trajectory as being composed of short parts that are shared between different trajectories and form a dictionary. We call those parts *motion primitives*. In a generative interpretation, a motion primitive can be activated at a certain point in time to generate a characteristic temporal sequence of points in space for a short time period. This dictionary can be learned from data, as will be presented later. See Fig. 2 for an exemplary set of motion primitives that have been learned from handwritten character trajectories. Given a dictionary of motion primitives, a trajectory can then be represented by the activation times and amplitudes of those primitives for generating the trajectory (see Fig. 3). This representation provides an alternative encoding of the trajectory in terms of sparse events, also called a sparse code.

As will be shown later, for similar trajectories, the activation times and amplitudes are again similar. Thus, we can characterize different classes of trajectories, e.g. the character classes 'a', 'b', etc., by the typical activation times and amplitudes of the motion primitives for those classes. To generate a trajectory from a desired class, first the activations have to be sampled and then the motion primitives have to be activated according to those activations.

Thus, we have a two-layered model for character trajectory generation. In the lower layer, given a set of activation times, a trajectory is generated by activating a number of motion primitives, that have been learned beforehand. As learning algorithm, we use the Non-negative Matrix Factorization (NMF).

In the upper layer of our two-layered model, the exact order and timing of the primitives is controlled with a timing model that stores knowledge about the typical activation times and amplitudes of the primitives for a desired class of

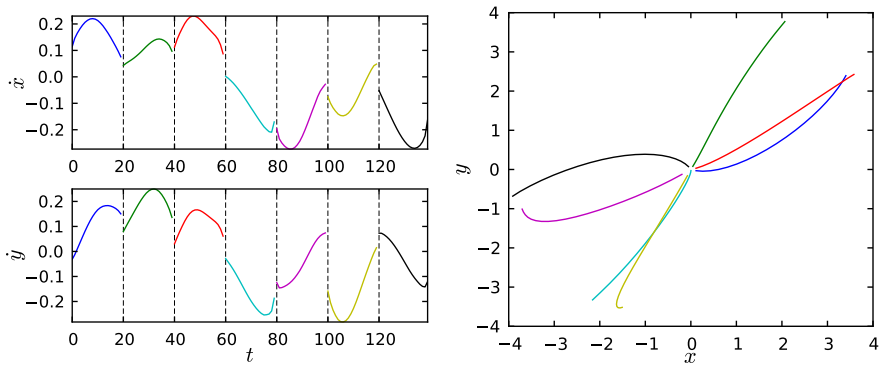

Fig. 2: Exemplary selection of motion primitives learned from the handwritten character data set in velocity space (left) and in pen space (right). In the left plot, the motion primitives (separated by dashed lines) have been appended to each other for visualization.

trajectories. The process generating the activation times is modeled by a group of Integrate-and-Fire (I&F) neurons (one for each motion primitive) that generate spikes, dependent on a given class of trajectories. The spikes are interpreted as activation times for the motion primitives in the lower layer. The input to the I&F neurons is the class-specific temporal activation density, which is also learned from the data.

We discuss related work in Sec. 2. Our approach is described in detail in Sec. 3. In Sec. 4, we show that we can generate visually appealing characters and illustrate some crucial parameter dependencies. Finally, we conclude our work in Sec. 5.

2 Related Work

There are a variety of approaches for sequencing of motion by means of motion primitives. The research in this area can be divided into two groups. In the first group, the existence of motion models is assumed that have been hand-crafted or learned in isolation in a supervised manner. In this group the most prominent approaches model motion primitives with Hidden Markov Models (see e.g. [6]) or Dynamic Movement Primitives (see e.g. [9]). Approaches in this group typically aim at a representation that can be used for reproduction on humanoid robots. In the second group, motion models are learned from the data in an unsupervised manner (see e.g. [12] and [5]). Those approaches typically aim at finding representations of data that are interpretable and uncover interesting features in the data. Our approach belongs to the second group.

In the domain of time series processing, sparse coding has been mainly used for auditory signal coding. In [8], the authors aim at computing a sparse representation of natural audio signals in form of spike trains, where the spikes mark activations of a fixed and hand-crafted set of basis functions. Given this set of basis functions, the amplitude and timing of their activations are learned.

The authors argue that such a representation provides a very efficient encoding and uncovers the underlying event-like structure of the signal. This work has been extended (e.g. in [11]) to also learn the basis functions to find an optimal dictionary or code of the signal. The authors show that the emerging basis functions can be compared to auditory receptors in animals and thus are naturally interpretable.

The main problem of such approaches is that the gradient-based techniques, used for optimization of the activations and the basis functions, lead to multiple adjacent activations with high values (activation traces; see Fig. 4 left), as opposed to sharply localized spikes desired for a sparse encoding. As a consequence, instead of optimizing the activations directly, heuristics like Matching Pursuit are used, where the subset of the activations is selected one after another by correlation with the basis functions and thresholding. We show in this paper, however, that temporally isolated activities can also be achieved without selection heuristics, but instead by directly formulating a penalty for adjacent activities and including this penalty as an additional energy-term for the basis vector decomposition model. During optimization, the penalty term naturally leads to a competition between rivaling activities, eliminating adjacent activations (see Fig. 4 right).

More recently, NMF has been applied to find patterns in data like neural spike trains [10] or walking cycles of human legs with constant frequency [5]. The length of the basis vectors must be specified manually and is typically chosen to be of the length of the expected patterns, e.g. a single spike pattern or a single walking cycle. However, for human movement data like handwriting, where a pattern in this sense is a whole character, NMF in this form can not be applied due to temporal variations of the underlying patters, like different speed profiles. Our approach is to interpret a pattern to be a combination of even smaller subparts (see Fig. 3), where the parts themselves have a lower temporal variability and the variability of the whole pattern is captured by shifting the parts in a small local region.

The above mentioned models address only the learning of basis functions and their activations. Our model additionally learns typical activation patterns for different classes of trajectories. In [12] an approach is presented, which is similar to our approach, but where the primitives in the lower layer are modeled by a factorial HMM (fHMM). To the contrary, we use a sparse coding framework, specifically the Non-negative Matrix Factorization (NMF), for learning the motion primitives, together with their activation times from unpartitioned training trajectories in an unsupervised manner. Further, in contrast to [12], where the layers are learned jointly, we separate the learning for the benefit of decreased computational complexity.

3 Method

In the following, the steps for learning the parameters of the layers in our model and for the generation of the trajectories will be described. The learning pro-

cedure consists of three stages. In the first stage motion primitives are learned from training trajectories in an unsupervised manner. In the second stage the activations of those motion primitives for all training trajectories in one class are temporally aligned. In the third stage, the aligned activities of all training trajectories in one class are used to learn class-specific intensity matrices (the activation density) of the I&F neurons that control the sequence of primitives.

3.1 Motion Primitive Learning

We formulate the motion primitive learning in the NMF framework. In general, with NMF one can decompose a set of N input samples into a small number $K \ll N$ of basis vectors and coefficients, called activations, to superimpose these to reconstruct the inputs. By imposing a non-negativity constraint and specific sparsity constraints on the activations, the resulting basis vectors are interpretable as parts that are shared amongst the inputs and constitute an alphabet (or dictionary) underlying the data [7].

We use a combination of two variants of NMF called semi-NMF and shift-NMF for learning the motion primitives from the handwritten character velocity profiles. Semi-NMF [3] relaxes the non-negativity constraint, such that only the activations are required to be non-negative. This allows the motion primitives to have positive and negative values, which we require for the velocity-based trajectory representation. Shift-NMF [4] introduces translation-invariant basis vectors. Thus, a basis vector can occur anywhere in the input, which is necessary for temporal signals with reoccurring patterns. See Fig. 3 for an example of the resulting representation.

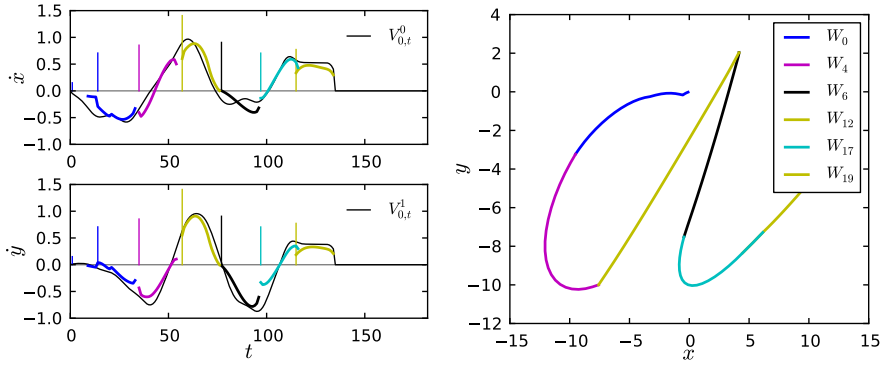

Fig. 3: Reconstruction of one character 'a' from the training data set after decomposition with NMF, according to eqs. 1 to 6. Left: reconstruction of the velocity profile of one input character (black line) by the learned parts (colored thick lines), scaled by their corresponding learned activations (vertical colored lines). The activations represent a sparse code of the trajectory. Right: velocity reconstruction (left) integrated over time, resulting in the position of the pen. The parts have also been colored. Note that shown here are the temporally integrated versions of the actual parts.

For ease of notation, we separate the spatial dimensions ($\dot{x}$ and $\dot{y}$) of the trajectories into distinct matrices, denoted by the upper index d. Let $\mathbf{V}^d \in \mathbb{R}^{N \times T}$ denote the matrix of N training trajectories of length T (shorter trajectories are padded with zeros), with elements $V^d_{n,t}$. The single trajectories are denoted as vectors $\mathbf{V}^d_n$. Let $\mathbf{W}^d \in \mathbb{R}^{K \times L}$ be the matrix of K basis vectors of length L, with elements $W^d_{k,l}$. We denote the single basis vectors by $\mathbf{W}^d_k$. Let $\mathbf{H} \in \mathbb{R}^{N \times K \times T}$ be the tensor of activations $H_{n,k,t}$ of the k-th basis vector in the n-th reconstruction at time t. In semi-NMF the activations are constrained to be non-negative, and thus $\forall n, t, k : H_{n,k,t} \geq 0$.

We learn $\mathbf{W}^d$ and $\mathbf{H}$ with NMF by minimizing the following energy function

$$F = \frac{1}{2} \sum_d \left\| \mathbf{V}^d - \mathbf{R}^d \right\|_2^2 + \lambda_g \sum_{n,k,t} H_{n,k,t} + \lambda_h h(\mathbf{H}) \ . \tag{1}$$

The matrices $\mathbf{R}^d \in \mathbb{R}^{N \times T}$ are the reconstructions of the trajectories by activation of the basis vectors $\mathbf{W}^d$ through activations $\mathbf{H}$, which can be formulated as a temporal convolution

$$R^d_{n,t} = \sum_k \sum_{t'} H_{n,k,t'} \hat{W}^d_{k,t-t'} \ . \tag{2}$$

Here, we introduced *normalized basis vectors* $\hat{\mathbf{W}}^d_k$, where the normalization is done jointly over all dimensions d. This normalization is necessary during learning to avoid scaling problems as described in [4].

The first two terms of the energy function 1 formalize the standard approximation scheme commonly used for sparse non-negative matrix factorization (see e.g. [2]), where the first term is the distance measure and the second term is a penalization of the overall sum of activations. Additionally, we introduced the function h, which is crucial to get an encoding interpretable as spike-like activations and will be described later.

This optimization problem can be solved by alternatingly updating one of the factors $\mathbf{H}$ or $\mathbf{W}^d$, while holding the other fixed. For semi-NMF usually a combination of least-squares regression of the basis vectors and multiplicative update of the activations is used [3]. The former, however has very high computational demands in the case of shift-NMF and is not applicable for our problem. Thus, we have to resort to gradient descent techniques. The following steps are repeated iteratively until convergence after initializing $\mathbf{H}$ and $\mathbf{W}^d$ with Gaussian noise.

1. Build reconstruction according to Eq. 2
2. Update the activities by gradient descent and make them non-negative

$$H_{n,k,t} \leftarrow \max\left(H_{n,k,t} - \eta_H \nabla_{H_{n,k,t}} F, 0\right) \tag{3}$$

$$\nabla_{H_{n,k,t}} F = -\sum_{d,t'} \left(V^d_{n,t'} - R^d_{n,t'}\right) \hat{W}^d_{k,t'-t} + \lambda_g + \lambda_h \nabla_{H_{n,k,t}} h \tag{4}$$

3. Build reconstruction according to Eq. 2

4. Update the basis vectors by gradient descent

$$W^d_{k,l} \leftarrow W^d_{k,l} - \eta_W \nabla_{W^d_{k,l}} F \tag{5}$$

$$\nabla_{W^d_{k,l}} F = -\sum_{n,d'} \sum_{t'} \left(V^{d'}_{n,t'} - R^{d'}_{n,t'} \right) H_{n,k,t'-l} \nabla_{W^d_{k,l}} \hat{W}^{d'}_{k,l} \tag{6}$$

The factors η_H and η_W are the learning rates. Note that the temporal correlations (all the sums over t') can be computed very efficiently in Fourier space. Note further, that expansion of the gradient in eq. 6 introduces dependencies between the dimensions. The derivation of the update equations by gradient descent from eq. 1 is straight forward and, thus, ommitted here.

Since two slightly shifted versions of the same basis vector are highly correlated with each other, typically, there are multiple non-zero activities at adjacent locations, which contradicts our idea of spike-like activations that are temporally isolated (see Fig. 4 (left)). Although non-isolated activities might give smoother trajectories, for the interpretation of the activities as temporal events that mark the beginning of motion parts, it is important to have clearly segregated activation peaks. In most approaches this is implemented by a heuristic like Matching Pursuit, which selects a subset of few activations beforehand. Instead, we enforce sharply localized activations directly by formulating a penalty for adjacent activations into the energy function by adding a term h that introduces a competition between adjacent activities. The competition is implemented by convolution of the activations with a triangular kernel function $z_H(k, k', t - t')$ that penalizes neighboring activities.

$$h(\mathbf{H}) = \sum_{n,k,t} H_{n,k,t} \sum_{k',t'} z_H(k, k', t - t') H_{n,k',t'} \tag{7}$$

$$z_H(k, k', t - t') = \begin{cases} 0 & \text{if } k = k', t - t' = 0 \\ \left(1 - \left|\frac{t-t'}{w}\right|\right) \cdot I\left(\left|\frac{t-t'}{w}\right| < 1\right) & \text{otherwise} \end{cases}, \tag{8}$$

where w is the kernel width, which we set to twice the length of the basis vectors. In the case of $k = k'$, activities of the same basis vector and adjacent to t are penalized, such that isolated spike-like activities emerge. In the case of $k \neq k'$, the activities of all other basis vectors that try to reconstruct the same part of the input are penalized. Thus, we enforce that approximately only one basis vector can be active during a time interval of L (the length of a basis vector) steps and that it can be active only once during that interval. See Fig. 4 for an illustration of the effect of the local activity competition.

After applying NMF to the data, we have a representation of the input in terms of learned basis vectors and activities. We interpret the basis vectors as motion primitives and their corresponding activities as temporal activations of the motion primitives. See Fig. 3 for an illustration of the resulting representation.

We observed that the activations are similar for trajectories of the same class. Thus, we can characterize a class by the average activations of a class. This will be used to build a model for the generation of activities for a given class.

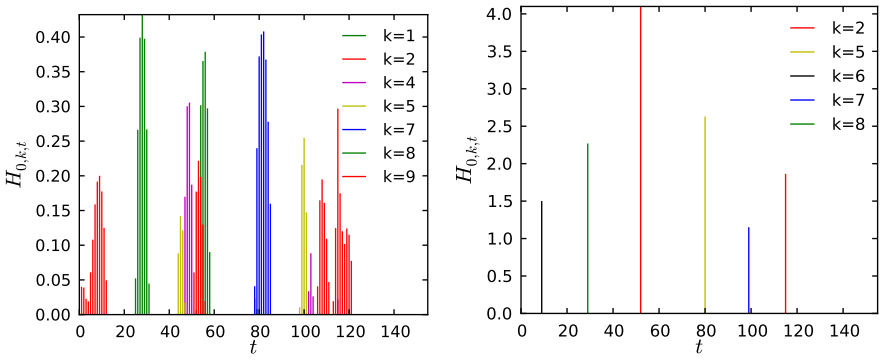

Fig. 4: Effect of the local activity competition. Without local activity competition (left), the activities (vertical colored lines) are distributed over adjacent locations. With the local sparsity extension (right), only one activity in a local cluster wins and all other activities are forced to zero.

3.2 Alignment of Activity Patterns

On top of the motion primitive layer we build a model for the generation of activations of motion primitives, given a character class. This model will be parametrized by an intensity matrix $\mathbf{I} \in \mathbb{R}^{K \times T}$ which is the relative frequency of an activation greater than zero of primitive $\mathbf{W}_k$ at time t and a scaling matrix $\mathbf{S} \in \mathbb{R}^{K \times T}$, which is the average amplitude of an activation of the k-th primitive at time t

$$I_{k,t} = \frac{1}{N} \sum_n \bar{H}_{n,k,t} \ , \qquad S_{k,t} = \frac{\sum_n H_{n,k,t}}{N I_{k,t}}, \tag{9}$$

where $\bar{H}_{n,k,t} = \Theta(H_{n,k,t})$ is the binarized activity and Θ is the Heavyside function.

The training trajectories in the data set, however, exhibit some variation in start time and average speed, which is also reflected in the activation patterns after the NMF step. This negatively affects the computation of $\mathbf{I}$ (see blue line in Fig. 5 (left)) and $\mathbf{S}$, because instead of having localized peaks, the intensities are spread over time. We align the activity patterns by associating with each training trajectory an offset a_n and stretching factor b_n and optimizing a_n and b_n iteratively by gradient ascent on the correlation between the individual activity pattern $\bar{H}_{n,k,t}$ and a linear interpolation of $I_{k,t}$. After this optimization, we apply eq. 9 again, but with the aligned activities, i.e. corrected by a_n and b_n to get the aligned intensity matrix $\hat{\mathbf{I}}$ and the aligned scaling matrix $\hat{\mathbf{S}}$.

3.3 Activity Generation

To model the generation of activities in the upper layer, we use a stochastic Integrate-and-Fire (I&F) model. For each motion primitive, there is one I&F neuron that activates the motion primitive by generating a spike. As input to

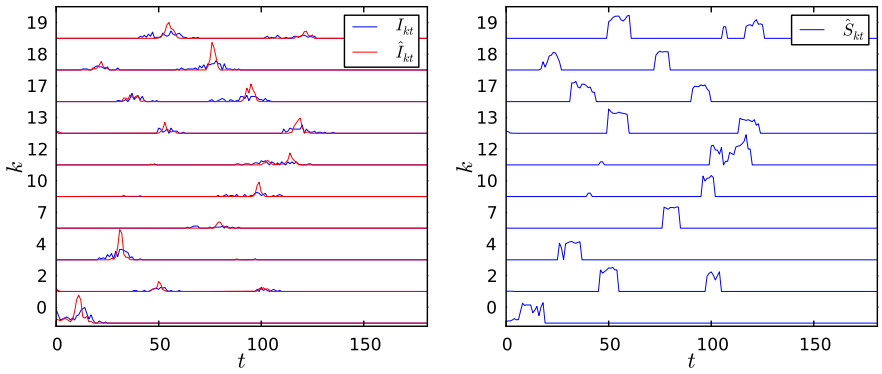

Fig. 5: Left: unaligned ($\mathbf{I}$) and aligned ($\hat{\mathbf{I}}$) intensity matrices for the character 'a' for the first ten basis vectors with highest maximum intensity. The intensities have been normalized to the maximum intensity of 0.235. Right: scaling matrix obtained from the aligned activity patterns for the same set of basis vectors. The scaling values have been normalized to the maximum value of 5.83.

the neurons, we use the average activations we computed earlier as the aligned intensity matrix $\hat{\mathbf{I}}_{k,t}$ (see Sec. 3.2).

The k-th neuron generates spikes $v_{k,t} \in \{0, 1\}$, which are interpreted as activation times of basis primitives, and thus are the generated counterpart of $\bar{H}_{n,k,t}$. The internal state $u_{k,t}$ of the k-th neuron is modeled by a leaky integrator

$$u_{k,t} = \begin{cases} u_{k,t-1} - \nu u_{k,t-1} + \hat{I}_{k,t} & : t - t' \geq \delta t_{ref} \\ \hat{I}_{k,t} - \nu u_{k,t-1} & : t - t' = 1 \\ 0 & : 1 < t - t' < \delta t_{ref} \ , \end{cases} \tag{10}$$

where t' is the time of the last spike before t, δt_{ref} is the absolute refractory time during which the load remains zero, and $\nu \in (0, 1)$ controls the amount of leakage. The neuron fires, i.e. $v_{k,t} = 1$, when $u_{k,t}$ exceeds a noisy threshold θ_t, which is sampled from a Gaussian during each simulation step.

After simulation of the I&F neurons, $v_{k,t}$ indicates the activation times for the motion primitives. For reconstruction of the actual trajectory, we also need the average amplitudes of the activations, which we computed earlier as the scaling matrix $\hat{\mathbf{S}}$ (see Sec. 3.2), since the generated spikes are only binary and they have to be scaled to actually use them as activations of the basis primitives. The generated trajectory, which we call $\tilde{\mathbf{R}}^d \in \mathbb{R}^T$, is then computed by the convolution of the basis vectors with the scaled spikes

$$\tilde{R}^d_t = \sum_k \sum_{t'} v_{k,t'} \hat{S}_{k,t'} \hat{W}^d_{k,t-t'} \ . \tag{11}$$

4 Results

We demonstrate our model on the Character Trajectories Data Set [12] available from the UJI Machine Learning Repository. We use the subset of all characters

consisting of only one stroke, since there is no principled approach to deal with trajectories consisting of multiple strokes and, thus, having a large discontinuity, yet. This will be investigated in future research.

Figure 6 shows the results of the sampling of one character trajectory. It can be seen from the lowest plot on the left side, that exactly one spike is generated in regions of high intensity (as indicated by the upper plot on the left).

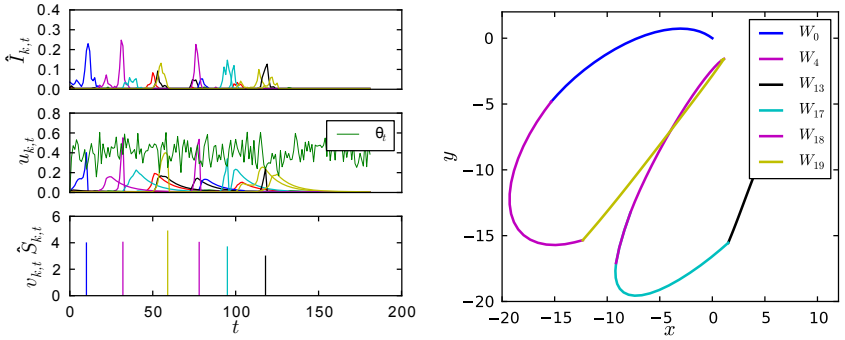

Fig. 6: Generation of trajectories through spiking neurons. Left: Process of spike generation, where the upper plot shows the aligned intensity matrix $\hat{\mathbf{I}}$, the middle plot shows the resulting load $u_{k,t}$ from the integration by the I&F model according to eq. 10, and the lowest plot shows the spikes $v_{k,t}$ that are generated and scaled by $\hat{S}_{k,t}$. Right: the trajectory that results from the convolution of the basis vectors with the scaled spikes in pen space, according to eq. 11.

The most crucial parameter of our model is the number of motion primitives, which must be chosen manually, because it has great influence on the quality of the reconstructions in the NMF step (see Sec. 3.1). Figure 7 (left) shows the reconstruction error, which is a measure of the quality of the approximation of the input, dependent on the number of character classes $|C|$ (i.e. 'a', 'b', etc.) in the training data set and the number of basis components used. For a fixed number of classes, e.g. $|C| = 20$, the reconstruction error decreases with increasing number of motion primitives. From $K = 15$ to $K = 20$ there is only a minor decrease in reconstruction error. This indicates that $K = 20$ is sufficient for this data set. Note, however, that this is highly dependent on the data set. The necessity to manually choose the number of basis components is a restriction of our approach. However, one can automatize the selection process by running the optimization multiple times with increasing number of basis components and stop when the relative decrease in error through addition of a basis vector is small.

We tested the behavior of the reconstruction error, when the variability of the training data is reduced, by reducing the number of character classes $|C|$. As expected, for a smaller number of character classes, the reconstruction error saturates at smaller K. Thus, the less variability in the data set the fewer motion primitives are needed. Except for Fig. 7 (left), in all the simulations of this paper,

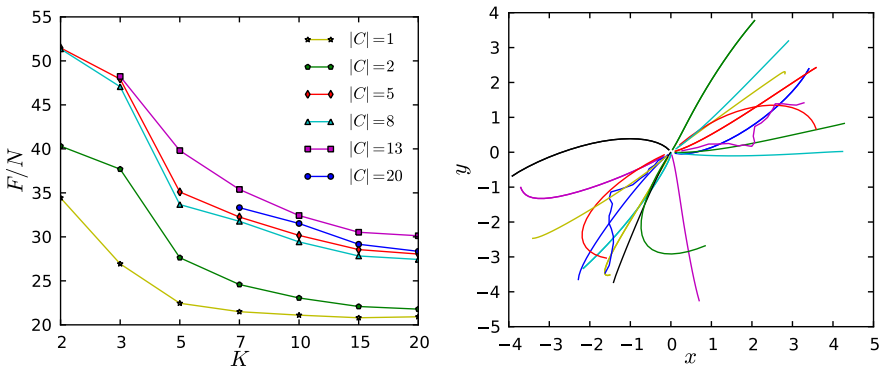

Fig. 7: Left: relation between cost F (normalized on the number of inputs N), number of basis vectors K and number of classes $|C|$. Choosing more than $K = 15$ basis vectors does not result in significant decrease of reconstruction error. Right: 20 learned basis primitives in pen space (i.e. temporally integrated). Overlapping basis vectors that appear very similar here, differ in the speed of execution.

we consistently used $K = 20$ motion primitives. Figure 7 (right) shows the 20 learned basis vectors in pen space. Note, that motion primitives that seem similar here differ in their speed of execution.

Figure 8 shows a number of representatives of successfully generated characters for all classes. The quality of the generated characters is sensible on the mean of the Gaussian firing threshold (see Sec. 3.3). If it is is chosen too high, some parts are not activated and thus missing in the trajectory, which results in defects in some characters. Further the scaling of the basis vectors sometimes results in overlong strokes like in the characters 'l' and 'm'.

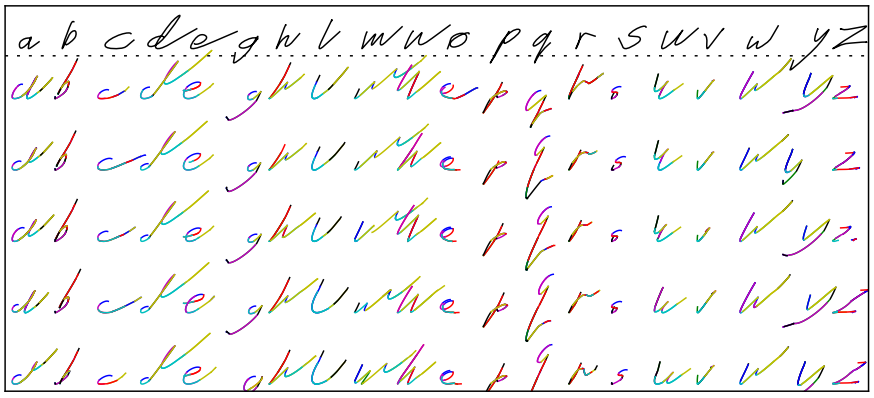

Fig. 8: The top row shows one example training character for each class from the training data set. The other rows show successfully generated samples for all 20 character classes. For some classes of the generated characters, like 'd' and 'z', small defects like missing parts can be observed.

5 Conclusion

We presented a model for learning the generation of handwritten characters based on a locally sparsified and translationally invariant NMF decomposition followed by an event-based activation through spiking neurons. The decomposition of the input patterns into smaller parts and their corresponding composition by learning their timing regime allows for an efficient handling of the temporal variations inherent in human movement data. We have shown that with the proposed model the handwritten characters can be successfully synthesized as a sequence of successive stroke parts.

The Integrate-and-Fire model for activation of primitives, however, sometimes results in defects in the resulting trajectories. Here we see room for improvement. The fact that our model delivers single, isolated spikes in regions with high intensity, invites for direct statistical models e.g. of Hidden Markov type. This will be investigated in future research.

References

1. Bizzi, E.: Modular organization of motor behavior in the frog's spinal cord. Trends in Neurosciences 18(10), 442–446 (Oct 1995)
2. Cichocki, A., Zdunek, R., Phan, A., Amari, S.: Nonnegative Matrix and Tensor Factorizations. Wiley (2009)
3. Ding, C., Li, T., Jordan, M.I.: Convex and semi-nonnegative matrix factorization for clustering and low-dimension representation (2006)
4. Eggert, J., Wersing, H., Körner, E.: Transformation-invariant representation and NMF. In: Proceedings of the 2004 IEEE International Joint Conference on Neural Networks. vol. 4, pp. 2535–2539. IEEE (2004)
5. Kim, T., Shakhnarovich, G., Urtasun, R.: Sparse Coding for Learning Interpretable Spatio-Temporal Primitives. Advances in Neural Information Processing Systems 22 (Dec 2010)
6. Kulic, D., Ott, C., Lee, D., Ishikawa, J., Nakamura, Y.: Incremental learning of full body motion primitives and their sequencing through human motion observation. International Journal of Robotics Research 31(2), 330–345 (2011)
7. Lee, D., Seung, H.: Learning the parts of objects by non-negative matrix factorization. Nature 401(6755), 788–91 (Oct 1999)
8. Lewicki, M.S., Sejnowski, T.J.: Coding time-varying signals using sparse, shift-invariant representations. Advances in Neural Information Processing Systems 11 (1999)
9. Meier, F., Theodorou, E., Stulp, F., Schaal, S.: Movement segmentation using a primitive library. In: 2011 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). pp. 3407–3412. IEEE (2011)
10. Roux, J.L., de Cheveign, A., Parra, L.C.: Adaptive Template Matching with Shift-Invariant Semi-NMF. Advances in Neural Information Processing Systems 21 (2009)
11. Smith, E., Lewicki, M.S.: Efficient coding of time-relative structure using spikes. Neural Computation 17(1), 19–45 (2005)
12. Williams, B., Toussaint, M., Storkey, A.: A primitive based generative model to infer timing information in unpartitioned handwriting data. In: Proceedings of the 20th international joint conference on Artifical intelligence (IJCAI'07) (2007)