Neurocomputing 124 (2014), 22-32

# Sparse Coding of Human Motion Trajectories with Non-negative Matrix Factorization

Christian Vollmer[a,b], Sven Hellbach[a,b], Julian Eggert[b], Horst-Michael Gross[a]

[a] *Ilmenau University of Technology, Neuroinformatics and Cognitive Robotics Lab, Helmholtz-Platz 5, 98693 Ilmenau, Germany*
[b] *Honda Research Institute Europe GmbH Carl-Legien-Str. 30, 63073 Offenbach/Main, Germany*

**Abstract**

We use shift-invariant Non-negative Matrix Factorization (NMF) for decomposing continuous-valued time series into a number of characteristic primitives, i.e. the basis vectors, and their activations, which results in a model-independent and fully data driven parts-based representation. We interpret the basis vectors as short parts of motion that are shared between all trajectories in the data set, and the activations as onset times of those parts. The extension of the shift-invariant NMF by a new competition term between adjacent activations allows to gain temporally isolated activation events, which further supports this interpretation. We show that the resulting sparse and compact representation can be used for the prediction of motion trajectories, and that it can be beneficial for classification, because it allows the application of simple standard classification models with few parameters. In this paper we show that basis vectors can be extracted, which can be interpreted as short motion segments. We present results on trajectory prediction, and show that the sparse representation can be used for classification of trajectories of a single joint, like the one of a hand, obtained by motion capturing.

*Keywords:* non-negative matrix factorization, time series, human motion, sparse coding, prediction

*Email addresses:* christian.vollmer@tu-ilmenau.de (Christian Vollmer), hellbach@htw-dresden.de (Sven Hellbach), julian.eggert@honda-ri.de (Julian Eggert), horst-michael.gross@tu-ilmenau.de (Horst-Michael Gross)

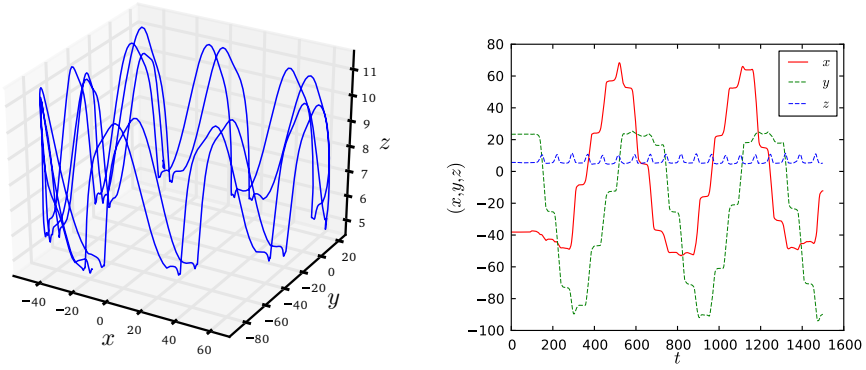



Figure 1: Example of the kind of trajectories analyzed in this paper. Shown is the trajectory of the right foot of a human walking in a circle, recorded using a motion capturing device with a sampling rate of 60Hz. Left: Trajectory in 3D space. Right: One-dimensional projections over time. For our decomposition, we approximate the problem by regarding all dimensions as independent. For the sake of clarity, in the following we concentrate on a single dimension (of the spatial dimensions $x$,$y$ and $z$ shown in the right graph) (the formal extension to multiple dimensions is straightforward and will be explored more in depth elsewhere).

## 1. Introduction

The understanding and interpretation of movement trajectories is a crucial step for the analysis of dynamic visual scenes with moving items. For example, consider the simple task for a robot of grasping an object which is handed over by the human interaction partner. Most approaches for describing motion patterns, like [1], rely on a kinematic model for the observed human motion. Using a particular model makes it difficult to adapt the approach to new tasks including other types of motion. Here, we aim at a generic, model-independent framework for decomposition, classification, and prediction. In this paper, we present an approach which finds prototypical segments of point trajectories that, in sequential combination, model the entire trajectory. We demonstrate our approach using one-dimensional components of trajectories of a single joints of a human, like the hand or foot, as shown in Fig. 1.

Non-negative Matrix Factorization (NMF) [2] is a blind source separation approach in concept similar to PCA and ICA that provides an efficient parts-based factorization for decomposing data under non-negativity constraints.

We use an extension of NMF with additional sparsity constraints as presented in [3] for decomposing a time series. The time series is decomposed into basis vectors and activations where the basis vectors can be interpreted as parts of motions that are shared amongst all trajectories in the data set, and thus, constitute a set of primitive motions. This yields a kind of representation, which is sometimes called *piano model* [4], where the sequence of activations can be understood in analogy to the keys of a piano, pressed by a piano player over time. Each pressed piano key then triggers a characteristic sound wave, the analog to our primitive, and the superposition of the primitives generates the melody, i.e. the analog to the time series.

A different interpretation, often used in the neural sparse coding literature, is to view the activations as spike-trains that mark the response of some kind of receptor. Both interpretations emphasize the fact that given the basis functions, we can transform the data into a more compact representation, where only the amplitudes and locations of the activations, or spikes, have to be stored to characterize a trajectory. Further, this representation is shift-invariant in the sense that the signal is characterized by the relative positions of the events. This property has been studied in early work on sparse coding of temporal signals. E.g. in [5], 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 over-complete set of basis functions. Given this set of basis functions, the amplitude and timing of the activations of those basis primitives 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 [6]) 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. Such methods target to achieve local sparsity (the activation events should be sparse and occur isolated in time); for this purpose, the authors use sequential selection heuristics like Matching Pursuit, where the subset of the activations is selected one after another by correlation and thresholding. A similar kind of decomposition has also been done for music in [7] to find a shift-invariant and sparse representation and to uncover latent structure in the data. Here the authors also use a (slightly different) heuristic to select the coefficients with the largest magnitude gradient, and get isolated and well-localized activities. We show in this paper, however, that spatio-temporally isolated activities can also be achieved without selection heuristics, but in-

stead 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 leads naturally to a competition between rivaling activities, eliminating spurious activity traces which are detrimental for further trajectory processing, as e.g. for classification. Further work that uses similar kinds of shift-invariant decomposition models include [8, 9, 10]. For recent publications, see [11, 12].

A different formulation of such a model that, however, results in similar representations and properties but at higher computational costs has been proposed in [13]. In that work, the motion primitives are described by a strict left-to-right Hidden Markov Model (HMM). The probability of triggering a primitive at time $t$ is described by an onset probability, very analog to the activation times in our approach (see Sec. 4.6). The primitives are then superimposed by means of a factorial Hidden Markov Model (fHMM), where each factor is a primitive HMM as described above. In [13] it was observed that the triggering probabilities are characteristic for a class in the data and provide a compact representation as a 'timing code'. Although very similar in concept, here we present a different method that allows the application of the efficient multiplicative learning and decomposition algorithms as gained from NMF. The resulting activations have very similar characteristics to the HMM approaches and can thus also be interpreted as a timing code of the trajectories.

The paper is organized as follows. In Sec. 2, we introduce the basics of the shift-invariant NMF-based decomposition, outline the necessary extensions of the NMF algorithm for the application on motion trajectories, and describe a penalty term that allows to improve the representational properties of the decomposition by getting decomposition results which have a smaller number of more isolated activity events than a straightforward NMF decomposition. In Sec. 3 we describe in more detail the preprocessing as well as the training and application phases for the decomposition of motion trajectories. We also explain how to use the algorithm continuously for online decomposition of a trajectory, and for trajectory prediction. In Sec. 4, we show the effects of the additional penalty term, typical results of the gained basis vectors, an evaluation of the predictive capabilities of our approach, and how classification can be done based on the sparse sets of activations.

## 2. Non-negative Matrix Factorization

Similar to other decomposition approaches like, e.g. PCA and ICA, Non-negative Matrix Factorization (NMF) [2] can be used to solve the source separation problem, where a set of training data $\mathbf{V}$ has to be decomposed into basis vectors $\mathbf{W} \geq 0$ and activations $\mathbf{H} \geq 0$:

$$\mathbf{V} \approx \mathbf{W} \cdot \mathbf{H}$$

Each training data sample is represented as a column vector $\mathbf{V}_i$ of the matrix $\mathbf{V}$. Each column $\mathbf{W}_j$ of the matrix $\mathbf{W}$ is a basis vector. In the activation matrix $\mathbf{H}$ the element $H_i^j \geq 0$ determines to which extent each basis vector $\mathbf{W}_j$ is activated to reconstruct a training sample $\mathbf{V}_i$.

Unlike PCA or ICA, however, NMF aims at a decomposition which only consists of non-negative elements. Thus, superposition of basis vectors always means accumulation of positive parts. There exists no primitive which is able to erase a 'wrong' superposition of other primitives, and different primitives cannot cancel out each other. This leads to basis vectors with different properties: They are often spatially more localized (as e.g. for PCA) and it has been argued that this supports parts-based representation schemes and has advantages for certain applications, like reported e.g. for face recognition [14].

For calculating the decomposition, optimization-based methods are used. For this purpose, an energy function $E$ is used, which targets a good reconstruction of the data. Within the energy function, additional desired constraints can be formulated as it is shown for an activation sparsity constraint in [15] and for transformation invariance of the basis vectors in [16]. This leads to the following energy function:

$$E(\mathbf{W},\mathbf{H}) = \underbrace{\frac{1}{2}\sum_i \|\mathbf{V}_i - \sum_j \sum_{(\mathbf{m})} H_i^{j,(\mathbf{m})} \mathbf{T}^{(\mathbf{m})} \mathbf{W}_j\|^2}_{:=E_R(\mathbf{W},\mathbf{H})} + \underbrace{\lambda \sum_{i,j,(\mathbf{m})} H_i^{j,(\mathbf{m})}}_{:=E_\lambda(\mathbf{H})}$$

Here $\mathbf{T}^{(\mathbf{m})}$ is a general transformation operator with a transformation parameter vector $\mathbf{m}$. The activations $H_i^{j,(\mathbf{m})}$ now indicate the contribution weight of the $j$'th basis vector $\mathbf{W}_j$ subject to the transformation with parameters $\mathbf{m}$ to the reconstruction of the $i$'th input vector $\mathbf{V}_i$. The transformation operator allows to extract a small set of basic primitives and to interpret a signal as being composed of transformed instances of those primitives. In principle we

can incorporate arbitrary transformations; in this paper we restrict ourselves mainly to translations corresponding to temporal shifts, since one usually expects a primitive to occur in a similar form at different points in time in the signal. E.g. for a grid-based representation, the set of all $\mathbf{m}$ could be identified with the set of discrete shifts of the temporal index, and thus $\mathbf{m} := m_t$. Then, $\mathbf{T}^{(m)}\mathbf{W}_j$ becomes a temporally shifted version of $\mathbf{W}_j$ and $H_i^{j,(m)}$ is the contribution of the shifted primitive $\mathbf{W}_j$ to the reconstruction of the input $\mathbf{V}_i$. The set of all $H_i^{j,(m)}$ comprise a higher-order tensor indexed by $i$, $j$, and $m$. The restriction to discrete translational shifts allows the use convolutions and correlations for the efficient calculation of the presented algorithm, see Appendix A and Appendix B for the details of such formulations and the relation to shift-invariant approaches in related publications.
In section 3, we introduce a preprocessing of the raw trajectory data that has beneficial properties for the application of discrete sets of shifts as transformations. For the implementation, we use a grid-representation that results in a two-dimensional rasterization of the trajectories in form of gray scale images. The input vectors to the NMF are then two-dimensional images with one axis representing the discrete time index and the other axis representing a spatial extension of the trajectory, and we can take advantage of shift invariance in these two dimensions. As a consequence, the transformation parameter vector $\mathbf{m}$ describes discrete index shifts in both, temporal and spatial, dimensions $\mathbf{m} := \{m_t, m_y\}$ with $m_t, m_y \in \mathbb{Z}$. In this case $H_i^{j,(\mathbf{m})}$ becomes a four-dimensional array indexed by $i$, $j$, and the shift parameters $m_t, m_y$. For a more elaborate explanation see Appendix A.

By minimizing the energy equation, it is now possible to model the input data using the matrices $\mathbf{W}$ and $\mathbf{H}$. The reconstruction is aimed to be as close as possible to the input data $\mathbf{V}$ (reconstruction term $E_R$), while using only a few activations $H_i^{j,(\mathbf{m})}$, which is achieved by forcing the activations to be non-negative and simultaneously penalizing them by the sparsity constraint term $E_\lambda$, controlled by the parameter $\lambda$.

For the optimization we use an iterative method, which updates $\mathbf{W}$ and $\mathbf{H}$ in alternating fashion using multiplicative update rules similar to those introduced by [2]. For a background on this choice of update rules and details about the derivation and convergence properties, see Appendix B and [2], [15] and [16] (in this order).

*2.1. Extending NMF by spatiotemporally local activity competition*

Starting from sparse and translationally invariant NMF, in the following section, we extend the approach by an additional energy term to obtain isolated activations, as motivated by the analogy to the key pressing times of a piano player. It has been observed before that optimization through gradient descent for shift-invariant models leads to adjacent non-zero activities [7, 8, 17], or even spatially smeared activity traces (see Fig. 2). This problem has been dealt with previously by sequential selection and thresholding heuristics, like in Matching-Pursuit-type algorithms. To enforce locally isolated peak-like activities directly from within our optimization, we add an energy term $E_\mu(\mathbf{H})$ to the energy function that introduces local competition between neighboring activations by placing a kernel function $K_i^{j,r,(\mathbf{m}-\mathbf{m}')}$ on each activation $H_i^{j,(\mathbf{m})}$ to suppress all its neighboring activations $H_i^{r,(\mathbf{m}')}$

$$E_\mu(\mathbf{H}) = \mu \cdot \sum_{i,j} \sum_{\mathbf{m}} H_i^{j,(\mathbf{m})} \sum_{r} \sum_{\mathbf{m}'} K_i^{j,r,(\mathbf{m}-\mathbf{m}')} \cdot H_i^{r,(\mathbf{m}')}.$$

The parameter $\mu$ controls the penalty term. This penalty can be described as a competition, for a single data vector $\mathbf{V}_i$, of the activation $H_i^{j,(\mathbf{m})}$ with all surrounding $H_i^{r,(\mathbf{m}')}$ weighted by a kernel, centered on the former one. For $j = r$, $K_i^{j,r,(\mathbf{m}-\mathbf{m}')}$ describes the competition between all potential spatial and temporal translations of the primitive $\mathbf{W}_j$, and for $j \neq r$, $K_i^{j,r,(\mathbf{m}-\mathbf{m}')}$ describes the competition between activations for different primitives $\mathbf{W}_j$ and $\mathbf{W}_r$, i.e., the different primitives compete for a place in space and time. The competition weights are defined in the following way

$$K_i^{j,r,(\mathbf{m}-\mathbf{m}')} = \begin{cases} 0 & \text{for } j = r \text{ and } \mathbf{m} - \mathbf{m}' = 0 \\ k(\|\mathbf{m} - \mathbf{m}'\|_2 / h) & \text{otherwise} \end{cases},$$

where $k(x) = (1 - x^2)I(|x| \leq 1)$ is the kernel function and h is the kernel width. We found the non-normalized Epanechnikov kernel to be suited for our purpose, but other kernels that are monotonically decreasing could also be used. This way, activities directly adjacent to the one in the kernel center are penalized most and the penalty decreases with increasing distance to the kernel center. Note that we do not normalize the kernel, because that would result in weaker competition, the wider the kernel is, which is undesirable for our purpose. Note further that for $j = r$ and $\mathbf{m} = \mathbf{m}'$, we set the kernel value

to zero, because a non-zero value would lead to an activation competing with itself.

Adding $E_\mu(\mathbf{H})$ to the energy function, we get

$$E(\mathbf{W},\mathbf{H}) = E_{\mathbf{R}}(\mathbf{W},\mathbf{H}) + E_\lambda(\mathbf{H}) + E_\mu(\mathbf{H}). \tag{1}$$

Differentiating w.r.t. the activations and separating into positive and negative terms for the stationarity conditions yields the following multiplicative update equation for the activities (with $\overline{\mathbf{W}}_j$ indicating Euclidean-normalized basis vectors)

$$H_i^{j,(\mathbf{m})} \leftarrow H_i^{j,(\mathbf{m})} \odot \frac{(\mathbf{T}^{(\mathbf{m})}\overline{\mathbf{W}}_j)^T\mathbf{V}_i}{(\mathbf{T}^{(\mathbf{m})}\overline{\mathbf{W}}_j)^T\mathbf{R}_i + \lambda + \mu\sum_r\sum_{\mathbf{m}'} K_i^{j,r,(\mathbf{m}-\mathbf{m}')}\cdot H_i^{r,(\mathbf{m}')}}.$$

(Here the operation $\odot$ denotes a component-wise multiplication and the division also has to be calculated component-wise). We call the competition term in the denominator $\sum_{\mathbf{m}'} K_i^{j,r,(\mathbf{m}-\mathbf{m}')}\cdot H_i^{r,(\mathbf{m}')}$, the *competition map*. Note that this is simply a convolution of the competition kernel with the activation (with both the kernel and the activation matrices interpreted as 2D arrays indexed by the transformation vector $\mathbf{m}$), and in addition for 2D translational transformations $\mathbf{T}^{(\mathbf{m})}$ the complete update equation can be computed very efficiently using correlations (see Appendix B for details) and FFT (Fast Fourier Transformations).Using this update scheme we obtain very isolated activations, which are sharply localized and segregated from each other (see Fig. 2 for a comparison of the decomposition without and with local competition term for the activations).

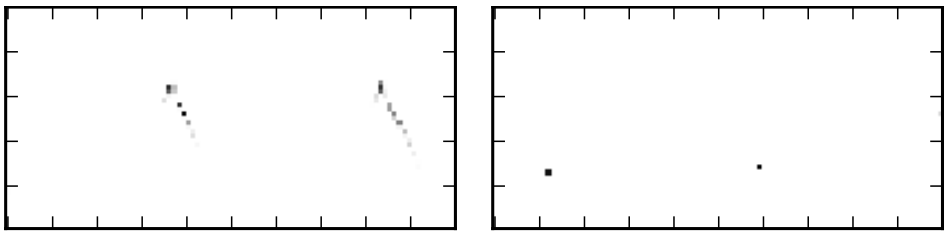
The following steps summarize the complete NMF update algorithm as used throughout this paper, and which minimizes the total energy function 1. During training, both the basis vectors and the activities are randomly initialized with Gaussian noise.

1. Normalize the basis vectors according to

$$\overline{\mathbf{W}}_j = \frac{\mathbf{W}_j}{\|\mathbf{W}_j\|}$$

2. Calculate the reconstruction

$$\mathbf{R}_i = \sum_j\sum_{\mathbf{m}} H_i^{j,(\mathbf{m})}\mathbf{T}^{(\mathbf{m})}\overline{\mathbf{W}}_j$$



(a) (b)

Figure 2: Activity vectors calculated without and with local competition, represented as 2-dimensional images with the $x$ and $y$-coordinates representing the temporal and spatial dimensions of the trajectory. Fig. 2(a) shows an activity vector obtained without the local competition. The activities are spread over a local region, resulting in pronounced *activity traces*. Fig. 2(b) shows the activity vector for the same input obtained incorporating spatial and temporal local competition. The resulting activities now become sharply localized peaks, isolated in space and time.

3. Update the activities

$$H_i^{j,(\mathbf{m})} \leftarrow H_i^{j,(\mathbf{m})} \odot \frac{(\mathbf{T}^{(\mathbf{m})}\overline{\mathbf{W}}_j)^T \mathbf{V}_i^T}{(\mathbf{T}^{(\mathbf{m})}\overline{\mathbf{W}}_j)^T \mathbf{R}_i + \lambda + \mu \sum_r \sum_{\mathbf{m}'} K_i^{j,r,(\mathbf{m}-\mathbf{m}')} \cdot H_i^{r,(\mathbf{m}')}}$$

4. Calculate the reconstruction with the new activities

$$\mathbf{R}_i = \sum_j \sum_{\mathbf{m}} H_i^{j,(\mathbf{m})} \mathbf{T}^{(\mathbf{m})} \overline{\mathbf{W}}_j$$

5. Update the basis vectors

$$\mathbf{W}_j \leftarrow \mathbf{W}_j \odot \frac{\sum_i \sum_{\mathbf{m}} (\mathbf{T}^{(\mathbf{m})})^T \mathbf{V}_i H_i^{j,(\mathbf{m})} + \overline{\mathbf{W}}_j \overline{\mathbf{W}}_j^T \sum_i \sum_{\mathbf{m}} (\mathbf{T}^{(\mathbf{m})})^T \mathbf{R}_i H_i^{j,(\mathbf{m})}}{\sum_i \sum_{\mathbf{m}} (\mathbf{T}^{(\mathbf{m})})^T \mathbf{R}_i H_i^{j,(\mathbf{m})} + \overline{\mathbf{W}}_j \overline{\mathbf{W}}_j^T \sum_i \sum_{\mathbf{m}} (\mathbf{T}^{(\mathbf{m})})^T \mathbf{V}_i H_i^{j,(\mathbf{m})}}$$

All vector divisions and the vector multiplications denoted by $\odot$ have to be applied componentwise. Steps 1 to 5 are iterated until a defined convergence criterion is reached, e.g. a threshold for the energy or the minimum decrease of energy. After training and during application, the basis vectors are held fixed, so that only steps 2 and 3 are iterated.

The Euclidean normalization of step 1 is usually not energy-preserving (i.e., it can lead to an energy increase), and would therefore counteract the energy minimization. As a consequence, we have formulated the energy function 1 in terms of the *normalized basis vectors*, which leads to the additional

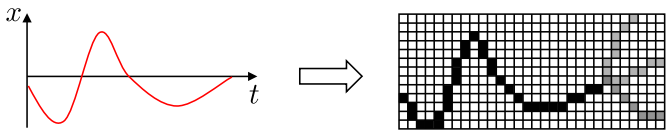



Figure 3: Preprocessing: The recorded motion trajectories are transferred into a grid representation. A grid cell is set to 1 if it is in the path of the trajectory and set to zero otherwise (each dimension is regarded separately). During the prediction phase multiple hypotheses can be gained by superimposing several basis primitives. This is indicated with the gray trajectories on the right side of the grid.

terms $(\overline{\mathbf{W}}_j \overline{\mathbf{W}}_j^T ...)$ in the update equations of step 5. For the details of normalization correction of the NMF algorithm see [15].

## 3. Decomposing Motion Trajectories

Lets assume, we are given a motion trajectory a time series of $N$ samples $\mathcal{T} = (\mathbf{s}_0, \mathbf{s}_1, \ldots, \mathbf{s}_{N-1})$ with values $\mathbf{s}_t = (x_t, y_t, z_t)^T$ for time steps $t = 0, 1, \ldots, N-1$. It is now possible to present $\mathcal{T}$ directly to the NMF approach. But this could result in an unwanted behavior, while trying to reconstruct the motion by use of the basis vectors. Imagine two basis vectors, one representing a left turn and another representing a right turn. A superposition of those basis primitives would result in a straight movement, since the two primitives would add up and cancel each other out.

The goal is to have a set of basis primitives, which can be concatenated one after the other. Furthermore, it is beneficial for a prediction task to be able to formulate multiple hypotheses. For achieving these goals, the $x$-$t$-trajectory is transferred into a grid representation, as shown in Fig. 3. Then, each grid cell $(x_i, t_j)$ represents a certain state (spatial coordinate) $x_i$ at a certain time $t_j$.

The grid size is chosen such that the number of columns, i.e. the temporal resolution, equal the number of samples. The number of rows, i.e. the spatial quantization, must be chosen to fit the desired degree of accuracy and is, thus, heavily dependent on the application domain. Clearly, the number of rows strongly affects the computational demands of our approach, and thus, a trade-off is necessary. For our experiments (see Sec. 4) we chose a number of rows of 50.

The 2D-grid is now presented as image-like input to the NMF algorithm as introduced in Sec. 2. Using the grid representation of the trajectory also

supports the non-negative character of the basis components and their activities. It has to be mentioned that the transformation to the grid representation is done for each of the dimensions individually. Hence, the spatio-temporal NMF has to be processed on each of these grids. Regarding each of the dimensions separately is often used as an approximation to reduce the complexity of the analysis of trajectories (compare [18]). Theoretically, the algorithm could also handle multidimensional grid representations. This is subject of ongoing research.

The sparse coding constraint helps to avoid trivial solutions. Since the input can be compared with a binary image, one possible solution would be a basis component with only a single grid cell filled. Using shift invariance, multiple differently located instances of such a basis vector could then be concatenated directly one after another. So, the trajectory would simply be 'copied' into the activities. Using the sparsity constraint helps avoiding this effect. However, the sparsity constraints alone leads to activations that are still not segregated, but 'smeared' in traces over a short period of time. The local competition removes those traces and segregates them such that there are only a few activations in the time interval of the length of a basis vector.

For real world tasks, the described approach can be divided into two phases, a training phase and an application phase. The goal of the training phase is to learn a set of basis primitives which allows the decomposition of an observed but unknown trajectory (see Fig. 4). As discussed in Sec. 3, the training samples are first transferred into a grid representation in a preprocessing step. These grid representations are then taken as input for the NMF approach and are therefore represented in the matrix $\mathbf{V}$. On this matrix $\mathbf{V}$ the NMF approach, extended by the sparsity constraint and by translation invariance as described in Sec. 2.1, is applied.

In the application phase, we keep the motion primitives, learned in the training phase, fixed, as indicated in Fig. 5. During the application phase, we assume that the motion of a dynamic object (e.g. a person) is tracked continuously. For getting the input for the NMF algorithm, a sliding window approach is taken. A certain frame in time is transferred into the already discussed grid like representation. For this grid the activation of the basis primitives is determined by trying to reconstruct the input. For the activity computation, the algorithm is identical to the steps 2 and 3 from Sec. 2.

In standard, offline approaches to NMF, each new observation demands a new random activity initialization for the optimization problem. During the application phase, the sliding window approach implies that a new time

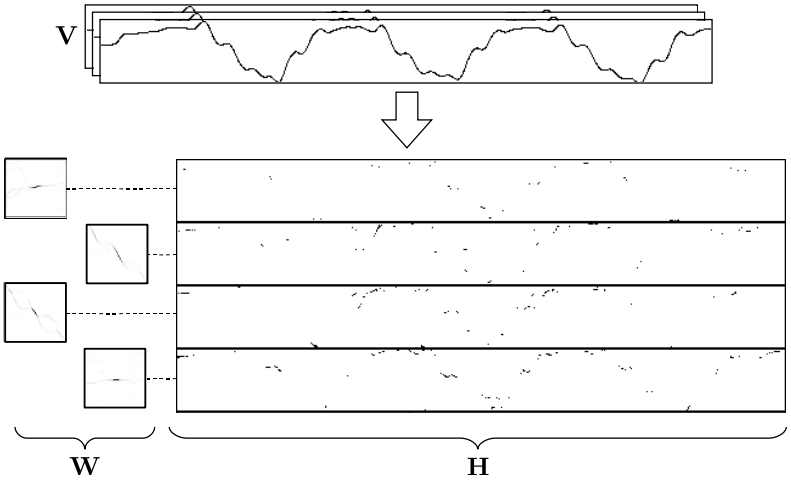



Figure 4: Training with Spatio-Temporal NMF. Given is a set of training samples stored in the columns of matrix **V**. The described algorithm computes the basis vectors **W** (motion segments/ motion primitives) and the corresponding activities **H**. The learned motion primitives are then used for processing arbitrary trajectories.

step reflects in a shift (e.g. to the left as time proceeds) of **V**. For identical activity initialization, the calculated next time step activities then also resemble the old, but shifted activities. To reduce the number of iterations until convergence, we therefore use the shifted activities from the previous time step as initialization for the current one.

An interesting question is whether our approach can be used to extrapolate trajectory information into the future, i.e., to use the learned basis vectors and the currently calculated activities during an online trajectory reconstruction to predict the input data for the next time steps. For this purpose the proposed algorithm had to be extended. Since the algorithm contains the transformation invariance constraint, the computed basis primitives can be translated to an arbitrary position on the grid. This means that they can also be moved in a way that they exceed the borders of the current input grid. Up to now, the size of reconstruction was chosen to be the same size as the sliding window on the input grid, and reconstruction results beyond the sliding window borders were clipped. To be able to solve the prediction task, we simply extend the reconstruction grid to the right – or into the future (see Fig. 5). So, the previously clipped information is now available for prediction. In this setting, the algorithm tries to use the most recent past observable input together with the learned basis vector structure

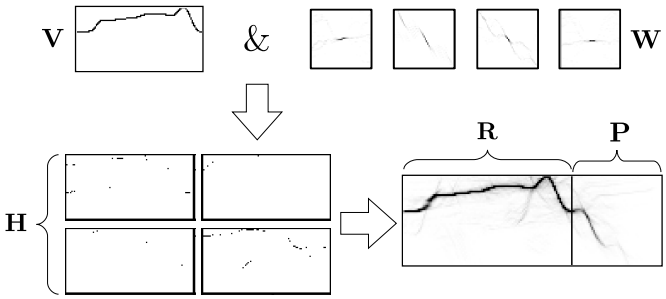



Figure 5: The application phase of the spatio-temporal NMF. The basis primitives $\mathbf{W}$, which were computed during the training, are used to reconstruct (matrix $\mathbf{R}$) the observed trajectory $\mathbf{V}$. This results in a set of sparse activities – one for each basis primitive – which describe on which position in space and time a certain primitive is used. Besides the reconstruction of the observed trajectory (shown in the lower right part, $\mathbf{R}$), it is furthermore possible to predict a number of time steps into the future. Hence, the matrix $\mathbf{R}$ is extended by the prediction grid $\mathbf{P}$.

to find those basis vectors (and shifts) which partially match best, and uses the unmatched part of the basis vectors for a trajectory prediction (the portion of basis vectors used for past matching vs. prediction can be controlled via translational offsets in the basis vectors). The predictive capabilities of our approach are examined in Sec. 4.

## 4. Results

In this section, we show experimental results of the various aspects of NMF for motion trajectory analysis and application scenarios that our approach can be used for. After describing the datasets we used for our evaluation, we show some typical basis vectors that emerge for the respective data set. We analyze the benefits of the local activity competition and discuss the predictive properties of our approach. We then briefly show that the gained encoding can help in building efficient motion trajectory classifiers.

### *4.1. Used Data Sets*

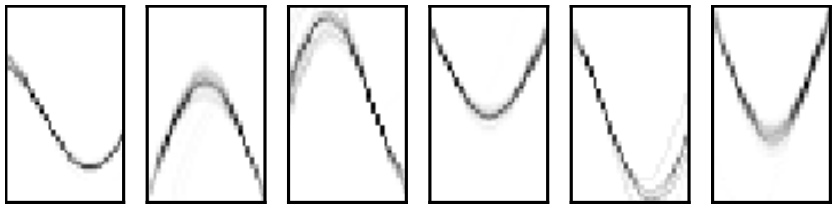
For the evaluation of our methods two data sets were used, one artificial and one real. The artificial data set consists of 500 Sinus curves, plotted on grids of size $100 \times 50$ (width $\times$ height), with a period length randomly varying in the range $[80, 100]$ (i.e. there is a full sinus period or a little bit

more in the grid) and an amplitude varying in the range $[10, 25]$. If not stated differently in the results below, we used basis vectors of size $30 \times 50$.

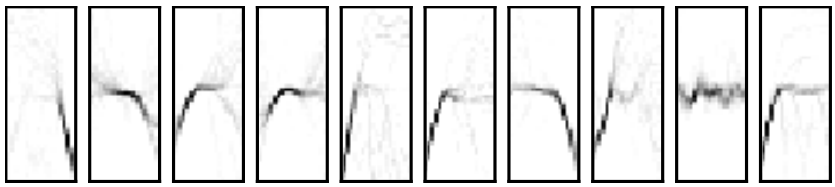
The real data set consists of movement data from the Perception Action Cognition (PACO) Lab at the University of Glasgow [19]. We will refer to this data set as the PACO data set. The data contains trajectories of 30 persons performing four different actions, 'knock', 'lift', 'throw', and 'walk'. In each recording session 15 markers were recorded. For the prediction experiments we use only the trajectories from the 'walk' class and only the marker of the right foot. From those three-dimensional trajectories again, we use only the $x$ component. Those are down-sampled to a frequency of 30Hz plotted on grids that have a number of columns equal to the number of samples. As mentioned in section 3, a trade-off between accuracy and computational cost has to be made, when choosing the number of rows, i.e. the quantization of the signal value. We found that a number of 50 steps is reasonable for the application presented here. For the classification experiments we used the data of all four classes and only the marker of the right hand, which seemed most discriminative. We use the z component of the three-dimensional trajectories, since this is the one most characteristic for the different motions. During one recording, the respective action, e.g. 'knock', is repeated several times. For the classification experiments we segmented the data and used the single segments. For further processing we normalized the amplitudes of all segments to $[0, 1]$ and the length to 50 by resampling. The segments are then plotted on grids of size $50 \times 50$. If not stated differently in the results below, we used ten basis vectors of size $20 \times 50$.

### *4.2. Emergence of parts of motions*

Using the PACO data set, we extracted 12 basis vectors from 10% of the exemplars, randomly chosen from all classes. Fig. 6(b) shows the obtained basis vectors. Every basis vector represents a common part that is shared between some or all trajectories in the data set. For the sinus data set NMF extracts the respective alternations of the sinus (Fig. 6(a)). Note the light-gray parts, where the variance in the data is represented. For the PACO dataset, some basis vectors represent straight movements, some show upswings and some downswings (Fig. 6(b)). As can be seen, the variance in the data is also captured by the basis vectors. They usually have some very well defined part (high values) and another part that shows some alternative movements that were present in the data. To obtain such nicely defined basis vectors the sparsity parameter $\lambda$ must be tuned, dependent on the actual

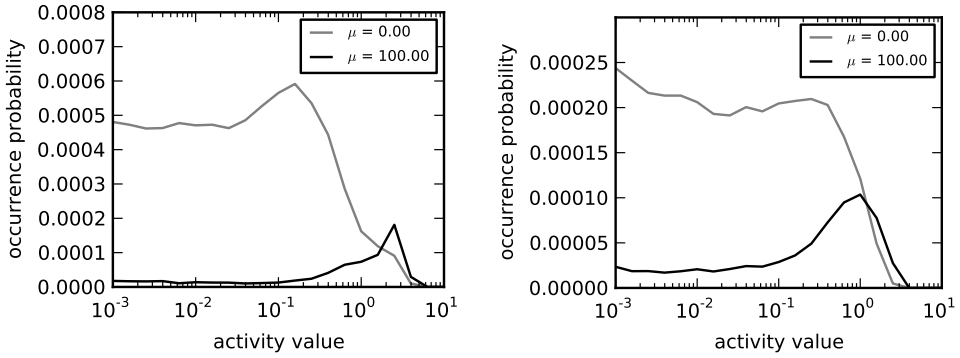


(a) Basis vectors from the Sinus dataset.



(b) Basis vectors from the PACO dataset.

Figure 6: Basis primitives gained by Spatio-Temporal NMF. The value for each grid cell is coded in gray scale from white (low) to black (high). A certain value stands for the influence of this grid cell, in the sense that light gray parts can be superimposed well, while dark gray to black parts indicate unambiguous trajectory segments. The appearance of the Sinus basis vectors in (a) supports the idea that the proposed algorithm indeed yields a parts-based decomposition (in this case, different half-periods of the sinus wave) that can be used for a reconstruction of Sinus-like trajectories by appropriate concatenation.




(a) Sinus dataset (b) PACO dataset

Figure 7: Distribution of magnitudes of non-zero activities for $\mu = 0$ and $\mu = 100$ on a logarithmic scale. Fig. (a) shows the distribution for the Sinus dataset, Fig. (b) the distribution for the PACO dataset. Activities with a magnitude below $10^{-3}$ are regarded as zero activities. For greater $\mu$ there are significantly less non-zero activities, and higher activity magnitudes (values of 0.1 and above) are favored against lower-valued ones.

data (in our setting we used parameters that varied between 0.1 and 1.0).

### *4.3. Effects of local activity competition*

In the following, we show the results of two experiments to demonstrate the benefits of the local activity competition with respect to the overall sparseness as well as the reconstruction abilities of the resulting decomposition.

The first experiment was conducted to show that local sparsity forces the activities towards a more binary encoding scheme and reduces the many activities with low values in favor of only a few activities with higher values. Fig. 7 shows histograms of activities after the decomposition for two data sets, the Sinus data set and the PACO data set. As can be seen, by using local activity competition the activities below a threshold of $10^{-1}$ are considerably reduced in favor of fewer activities with higher values above the threshold. Thus, the relaxation process of the NMF algorithm 'has to decide' for a fewer outstanding activities and the less prominent ones are suppressed.

Fig. 8 shows a plot of the reconstruction cost $E_R$ versus the sparsity cost $l^0_\epsilon$ for $\epsilon = 10^{-3}$. The sparsity measure $l^0_\epsilon = |\{(i,j,t,x), H_i^{j,(t,x)} > \epsilon\}|$ counts all activities that have a value of above $\epsilon$, as proposed by [20]. For sparsity cost values of below 0.0125 (which means that less than 1.25% of the activities are above $\epsilon$) the gain of the local activity competition becomes

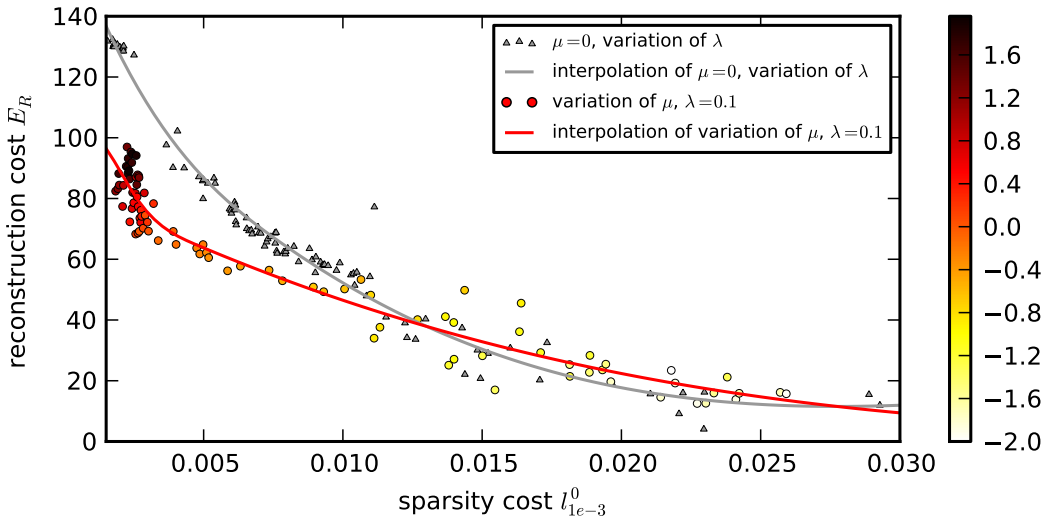



Figure 8: Plot of NMF reconstruction cost $E_R$ versus the sparsity cost measure $l^0_\epsilon$. The triangle markers show the change in the cost tradeoff of a number of different values of $\lambda$ and a fixed value of $\mu = 0$, i.e. without local activity competition. This serves as a baseline for the values shown with the circle markers, where the value of $\lambda = 0.1$ was held fixed and the value of $\mu$ was varied between $10^{-2}$ and $10^2$. Encoded in the face color of the circles is the value of $\mu$ as shown in the color bar on the right side. The scale on the color bar shows the decimal power of the value of $\mu$.

visible. For example for a reconstruction cost of 70 only about 0.3% of all possible activations need to be active using the local activity competition, whereas without, about 0.7% are active. Also notice that the weight $\mu$ must be larger than $\mu = 0.1$ for the local activity competition to show a measurable effect.

### *4.4. Computational Cost*

To evaluate the average computational cost we used the Sinus dataset with input images of size $100 \times 50$ and basis vectors of size $50 \times 50$. We make a distinction between the computational cost in the training phase and in the application phase. In the training phase both the basis vectors and the activities must be updated and thus the complexity is $O(JN)$ for $J$ basis vectors and $N$ inputs. Learning 10 basis vectors from 100 input images in the training phase on a Intel Core2 Quad 2.6GHz machine using only one core (no parallelization), one NMF update step takes 2775 ms. Since the NMF update can be parallelized, we can make full use of all four available cores,

which reduces the update time to 898 ms. The cost in both cases consists of about $25\% - 30\%$ for the update of the basis vectors, $40\% - 50\%$ for the update of the activities, and 20% for the update of the competition maps. It takes around 60 update steps for the energy value to converge to a local minimum.

In the application phase, the basis vectors are held fixed. Only the current input image is processed, which consists of the grid transformation of the currently observed trajectory. Since we do not need to update the basis vectors, the time complexity reduces to $O(J)$ and the time for one update step takes 19 ms. Parallelization does only have marginal effects in that case. Also the number of update steps necessary for convergence reduces to 10 to 20, dependent on the quality of the basis vectors.

### *4.5. Prediction*

For the example scenario from the introductory section (see Sec. 1), a robust identification and tracking of the single body parts is needed. To be comparable and to avoid errors from the tracking system influencing the test results, the PACO data set is used. The movement data has been subsampled to a resolution of 30 samples per second. Figure 9 shows prediction over a 10-step horizon into the future. A prediction of 10 steps means a prediction of 0.33 seconds into the future. Since most trackers work with a lower resolution, a larger prediction horizon is also possible.

For the experiments, the size of the basis primitives was chosen to be $50 \times 100$ grid cells. The input grid size was set to $500 \times 100$ during the training phase. During application phase, a window of 100 steps into the past was observed and the prediction is done over 10 steps into the future.

In figure 9, one can see that the non-determinism inherent in prediction of movement data is captured by our approach. For example, in the prediction window between steps 180 to 189, all possible variations that have been 'seen' during training in the data are present (light grey trajectory traces). This way, our approach can not only provide the most likely prediction, but a number of alternative predictions, each with a certain likelihood. The likelihood can be computed by normalizing over a column in the prediction window. This ability may become useful for further trajectory processing steps, as it is known to enhance the performance of, e.g., probabilistic trackers.

For evaluating the quality of the prediction, two measures have been chosen. Firstly, the prediction is compared with the grid representation of

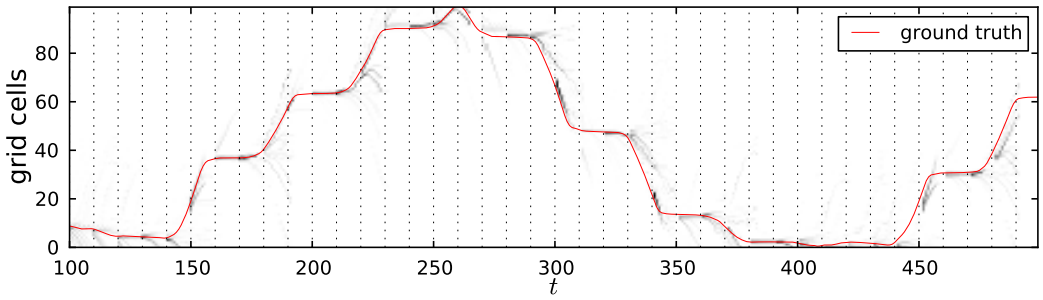



Figure 9: Prediction capabilities over a horizon of 10 steps into the future. A window of the 100 past values has been taken, transformed into the grid representation, and predicted 10 steps into the future. Then the window has been shifted 10 steps into the future (to the right) and the extrapolated trajectory for the next 10 steps has been generated (light grey traces). The dotted lines mark the start of each 10-step prediction. The ground truth (red line) has been projected onto the grid for comparison.

the true, actual trajectory $\mathbf{G}$. For each occupied grid cell the value of the column-wise normalized prediction is added:

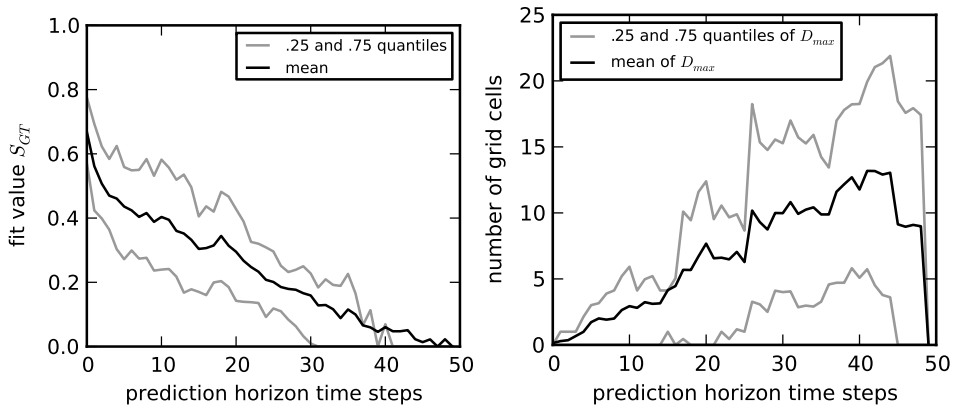
$$S_{GT}(t) = \frac{\mathbf{P}_t^T}{\sum_i \mathbf{P}_t^i} \cdot \mathbf{G}_t \tag{2}$$

The normalization of the prediction is done separately for each time slice (i.e., for each column in the grid) $\mathbf{P}_t$ within the prediction horizon.

Secondly, $D_{max}(t)$ measures the distance of the most likely path of our prediction to the ground truth.

$$D_{max}(t) = \mathrm{dt}(\mathbf{M})_t^T \cdot \mathbf{G}_t \tag{3}$$

For this purpose, the maximum prediction $\mathbf{M}$ is extracted, such that $M_t^i = 1$ where $\mathbf{P}_t$ has its maximum value and $M_t^i = 0$ otherwise, for each time slice $t$. Then a distance transform is applied that assigns each pixel of the output image the distance to the nearest non-zero pixel of the input image. The basis primitives can at most be shifted by their width out of the reconstruction grid $\mathbf{R}$. Thus, the maximum size of the prediction horizon equals the width of the basis primitives. In practice, this maximum can not be reached, because the basis primitives need a reliable basis in the trajectory part where the input is observable. Nevertheless, we have chosen to use the theoretical maximum as a reference for the evaluation.




(a) (b)

Figure 10: Measures for the prediction quality for different prediction horizon length (in sampling time steps). Fig. (a) shows $S_{GT}(t)$ (see Eqn. 2), which measures the correlation of the prediction with the ground truth. A fit value of 1.0 indicates perfect prediction at that time step. As it is expected the accuracy of the prediction decreases for a longer prediction period. Fig. (a) shows the distance $D_{max}(t)$ between the most likely hypothesis of the prediction and the ground truth trajectory (see eq. 2). Low values indicate that the prediction has a low distance to the ground truth trajectory.

The results of the prediction of a sample trajectory are depicted in Fig. 10. The predictions were performed at each tenth time step of the chosen trajectory. Fig. 10(a) shows the expected decrease of the average prediction quality $S_{GT}(t)$ with increasing prediction horizon. Nevertheless, the decrease is smooth and no sudden collapses can be observed. Fig. 10(b) shows the distance $D_{max}(t)$ between the most likely hypothesis of the prediction and the ground truth trajectory. In the first few steps, the prediction lies within a distance of only a few grid cells. With longer prediction horizon, the distance increases. The decrease in prediction quality with longer prediction horizon is partly due to the decreasing support for basis vectors from the input and partly due to the increasing presence of multiple hypotheses with roughly equal likelihood.

As mentioned earlier, the discretization into the grid introduces a loss of accuracy. For the presented application, however, we found that this discretization still gives good results. To make the output more smooth one could use interpolation techniques, e.g. via splines. This would however demand certain smoothness assumptions about the data. In our application on human motions, those assumptions hold. In general, however, this narrows

the applicability of our approach.

*4.6. Classification*

The activation patterns for a single trajectory can be seen as an alternative encoding of that trajectory in terms of events, where an event is marked by the time-point of an activation. In the following, we show that the transformation of a trajectory into that representation preserves all important discriminative characteristics of the original data necessary to perform classification. For this purpose, we built a very simple classifier that takes only the activation times (event times) as input, and applied it on the PACO data set. This data set consisted of trajectories from four classes ('knock','lift','throw','walk'), with a certain number of repetitions of the respective movement in each recording. In addition, a temporal partitioning is available, so that we can also access the single repetitions, and we used these as input for the classification. In short, we take a single repetition of a movement, transform it into an event-like encoding using the basis vector decomposition, and feed the activation times (event times) as input to the classifier.

More specifically, let $T_j = \{m_t | \sum_{m_y} H_i^{j,\{m_t,m_y\}} > \Theta\}$ be the set of time indexes when the activation of a basis vector $j$ was above a threshold $\Theta$, and thus considered active. An observation is then defined as a set of activations for all primitives $\{T_j\}_{j=1}^J$. Fig. 11 shows activation times of the first three basis components for a number of exemplars from the classes 'knock' and 'lift'.

For classification, we use a very simple model. As an approximation, we regard the primitives as independent. For each class, we have a set of $J$ Gaussian mixture models (GMM). The $j$-th GMM models the distribution of activation times of primitive $j$ for the given class. During application phase, GMM $j$ gives the likelihood of the observed observation times of primitive $j$. The likelihood of primitive $j$ is weighted by the relative frequency of activation times in the training data of that primitive for the given class, such that primitives that are rarely activated have a smaller influence on the classification result. Formally, this results in the following model

$$P(\{T_j\}_{j=1}^J | c) \propto \prod_j w_j^c \prod_{t \in T_j} \sum_{r=1}^R m_r^{c,j} \cdot \mathcal{N}(t; \mu_r^{c,j}, \sigma_r^{c,j}),$$

where $w_j^c$ is the relative frequency of the event that primitive $j$ has been

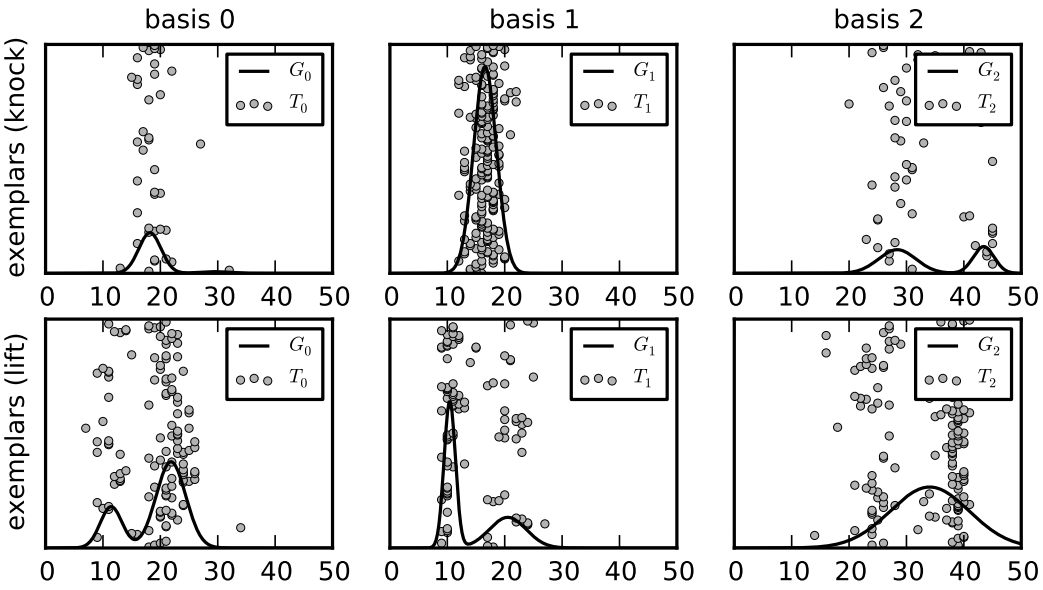



Figure 11: Activation times of the first three basis vectors for the two classes 'knock' and 'lift' (circles) and the corresponding learned basis-wise Gaussian Mixture Models (solid lines). The relative heights of the components of the GMM's have been scaled by their respective weights. This GMM model yields a classification rate of 0.944, which is considerably good, regarding the simple nature of the classifier and the very reduced representation of the data.

active for class $c$ and $\{(m_r^{c,j}, \mu_r^{c,j}, \sigma_r^{c,j})\}_{i=1,\ldots,R}$ are the mixture parameters of the GMM for class $c$ and primitive $j$ with $R$ components.

Fig. 11 shows the learned GMMs for the classes 'knock' and 'lift' for the first three basis vectors. We used the GMMs as a maximum likelihood classifier and validated the ability to classify the data correctly by stratified ten-fold cross-validation. The overall data set consisted of about 250 exemplars per class. Our experiments resulted in a mean classification rate of 0.944.

To get a base line for comparison, we also classified the data in the original representation (the raw trajectory segments). We resampled the segments with 50 steps and interpreted them as vectors. For each class, we trained a GMM on the vectors of that class. The classification rate was 0.940, and thus, comparable to the one in the transformed representation.

The comparable performance of the classifiers suggests that the characteristics of the data are still present when only regarding the activation times of the primitives. This implies that we can use the NMF decomposition to detect events and to build a classifier that only relies upon these events. However since the event-like encoding of the data results in a highly compressed representation of the data, we expect that alternative classification models can take full advantage from such properties and achieve similar classification performance at highly reduced computational costs.

## 5. Conclusion

In this paper, we summarized our research efforts on motion trajectory analysis using a new variant of Spatio-Temporal NMF. Furthermore, we introduced a new extension of the Non-Negative Matrix Factorization for obtaining sparse and isolated activations.

We are using our approach to decompose human motion trajectories into basis primitives and their activations over time. Our approach is able to gain the basis primitives from the training set in an unsupervised manner. Hence, no further model knowledge is needed. This new way of representing motion trajectories allows us to describe our motion in a sparse, event-like way.

This activation-based representation gives us an alternative input coding that can be used efficiently for classification. The sparse, event-based encoding enables the use of compact classifiers with a reduced set of parameters.

Furthermore, since the basis primitives adhere to the temporal characteristics of the trajectories used for training, it is possible to use our method

for a long-term prediction of an observed movement.

We have shown that applying our approach to human motion trajectories leads to the extraction of basis vectors interpretable as primitive motions, and to a compact and sparse representation of the trajectories. We have also shown, that the basis vectors can be used for model-independent multi-hypotheses prediction with good prediction accuracy, dependent on the discretization and on the quality of the basis vectors obtained. In addition we showed how a classifier can be built based exclusively on the timing code of the representation, and that such a classifier performs comparably to other simple classifiers on the raw trajectory data, which suggest, that all discriminative properties are maintained during the transformation into the sparse encoding.

In this paper, as an approximation, we have analyzed the dimensions of our Cartesian input domain independently, processing each of them as an individual trajectory. As a next step we want to test our method with the trajectories in the fully coupled input space.

In our current approach, the number of basis primitives has to be fixed in advance. While this was not a severe issue for the presented data as long as this number was large enough, it would be desirable to test methods to determine an appropriate basis vector size automatically. For example, iterative formulations for incremental basis vector learning (being different from incremental decomposition techniques like Matching Pursuit) could be investigated. Furthermore, relevance learning can be used to estimate which basis primitives bear the most important information for a post-processing classification task.

## Appendix A. Shift invariance

For the formulation of the energy function 2, we adhere to a generalized notation that is close to that commonly used in when dealing with sparse, shift-invariant representations. The background is that the data is reconstructed by a set of only a few contributions from arbitrarily transformed basis functions (e.g. *kernel* functions in [5]). Introducing

$$\begin{aligned} \mathrm{corr}_{\mathbf{V},\mathbf{F}}(\mathbf{m}) &:= \sum_{\mathbf{n}} V^{\mathbf{m}+\mathbf{n}} F^{\mathbf{n}} \\ \mathrm{conv}_{\mathbf{V},\mathbf{F}}(\mathbf{m}) &:= \sum_{\mathbf{n}} V^{\mathbf{m}+\mathbf{n}} F^{-\mathbf{n}} \end{aligned} \tag{A.1}$$

as standard correlations and convolutions on vectors $\mathbf{V}$ and filters $\mathbf{F}$ with discrete entries $V^{\mathbf{m}}$, $F^{\mathbf{m}}$ indexed by $\mathbf{m}$ (which is written as a vector because it may be more than 1-dimensional) and translational transformations (with $(T^{(\mathbf{m})})^{-1} = (T^{(\mathbf{m})})^T$)

$$\begin{aligned} (T^{(\mathbf{m})}\mathbf{W}_j)^{\mathbf{k}} &:= W_j^{\mathbf{k}-\mathbf{m}} \\ ((T^{(\mathbf{m})})^T\mathbf{W}_j)^{\mathbf{k}} &:= W_j^{\mathbf{k}+\mathbf{m}} \end{aligned} \tag{A.2}$$

we have that the $i$-th data vector is modeled by a reconstruction composed of shifted basis vectors weighted by their (shift-dependent) activities,

$$R_i^{\mathbf{k}} = (\sum_j \sum_{\mathbf{m}} H_i^{j,(\mathbf{m})} \mathbf{T}^{(\mathbf{m})} \mathbf{W}_j)^{\mathbf{k}} = \sum_j \sum_{\mathbf{m}} H_i^{j,(\mathbf{m})} W_j^{\mathbf{k}-\mathbf{m}} = \sum_j \mathrm{conv}_{\mathbf{H}_i^j, \mathbf{W}_j}(\mathbf{k}) \tag{A.3}$$

so that we the following reconstruction energy

$$E_R(\mathbf{W}, \mathbf{H}) = \frac{1}{2} \sum_i \|\mathbf{V}_i - \sum_j \sum_j \mathrm{conv}_{\mathbf{H}_i^j, \mathbf{W}_j}\|^2 \ . \tag{A.4}$$

If we assume only temporal shift invariance, eq. A.3 is a mixture of convlutions which has the same form as the approaches used e.q. in [5, 7, 8] and others.

In our model we used a discretized, grid-based representation with the $x$ and $y$ coordinates denoting temporal and spatial translations, so that $\mathbf{H}_i^j$, $\mathbf{W}_j$, $\mathbf{V}_i$ and $\mathbf{R}_i$ can all be understood as 2-dimensional *images* with temporal and spatial extension along the axes. The convolution is then calculated over both the temporal and the spatial translations, and $H_i^{j,\mathbf{m}}$ is the activity of the $j$'th basis vector shifted my $\mathbf{m} := \{m_t, m_y\}$, i.e., $m_t$ pixels along the temporal extension and $m_y$ pixels along the spatial extension.

## Appendix B. Update equations

In [21], the authors showed that for non-negative basic-vector decomposition, the multiplicative NMF equations (written here in extended per-image form for better comparison with the translationally invariant version later on)

$$H_i^j \leftarrow H_i^j \odot \frac{(\mathbf{W}_j)^T \mathbf{V}_i}{(\mathbf{W}_j)^T \mathbf{R}_i} \tag{B.1}$$

and

$$\mathbf{W}_j \leftarrow \mathbf{W}_j \odot \frac{\sum_i \mathbf{V}_i H_i^j}{\sum_i \mathbf{R}_i H_i^j} \tag{B.2}$$

for the activities and the basis vectors can be understood as derived from a diagonally rescaled gradient descent method and its convergence properties were ahown using auxiliary functions similar to those used for Expectation-Maximization convergence proofs. The same type of proof can be conducted for the transformation invariant energy $E_R$. This leads to update equations

$$H_i^{j,(\mathbf{m})} \leftarrow H_i^{j,(\mathbf{m})} \odot \frac{(T^{(\mathbf{m})}\mathbf{W}_j)^T \mathbf{V}_i}{(T^{(\mathbf{m})}\mathbf{W}_j)^T \mathbf{R}_i} \tag{B.3}$$

and

$$\mathbf{W}_j \leftarrow \mathbf{W}_j \odot \frac{\sum_i \sum_{\mathbf{m}} (T^{(\mathbf{m})})^T \mathbf{V}_i H_i^{j,(\mathbf{m})}}{\sum_i \sum_{\mathbf{m}} (T^{(\mathbf{m})})^T \mathbf{R}_i H_i^{j,(\mathbf{m})}} \tag{B.4}$$

which are very similar in form to the original NMF equations above, the only differences being the additional transformation indices $\mathbf{m}$. These are the first factors in the quotients of the update equations from section 2.1. (See however [22] for a more precise discussion of convergence properties and [23] for a theorem that proves that every limit point of the alternated NMF algorithm is a stationary point of the basis vector optimization problem $E_R$.)

For the special case of translational invariance only (see Appendix Appendix A), we can directly write the update equations with the help of correlations,

$$H_i^{j,\mathbf{m}} \leftarrow H_i^{j,\mathbf{m}} \odot \frac{\mathrm{corr}_{\mathbf{V}_i,\mathbf{W}_j}(\mathbf{m})}{\mathrm{corr}_{\mathbf{R}_i,\mathbf{W}_j}(\mathbf{m})} \ . \tag{B.5}$$

and

$$\mathbf{W}_j \leftarrow \mathbf{W}_j \odot \frac{\sum_i \mathrm{corr}_{\mathbf{V}_i,\mathbf{H}_i^j}}{\sum_i \mathrm{corr}_{\mathbf{R}_i,\mathbf{H}_i^j}} \ . \tag{B.6}$$

For the final update equations from section 2.1, we have additionally considered sparsity terms and a normalization constraint of the basis vectors. Ways to introduce sparsity terms into multiplicative, NMF-type algorithms are explained e.g. in [24]. The normalization becomes necessary because the solutions are only unique up to a scaling factor: the linearity in the reconstruction allows the system to freely scale either the activities or the basis vectors since $R = (\alpha W)H = W(\alpha H)$. Together with sparsity terms which

penalize large activities, such a system tends to scale up the basis vectors while at the same time driving the overall sparsity towards 0. The introduced basis vector normalization prevents this effect.

The concrete form of our update equations from section 2.1 has been gained by calculating the stationary point conditions for the activities and the basis vectors through derivations by $H$ and $W$ while holding $W$ and $H$ fixed, respectively, and then sorting positive and negative factors and formulating them as a quotient for the multiplicative update. Near-zero components of the update equations have been thresholded to a small $\delta$ (see again [23] for a justification of this). Using 2D Fast-Fourier transforms and performing appropriate sums in Fourier space allows to implement the update rules very efficiently. For this purpose, the vectors $\mathbf{V}_i$, $\mathbf{R}_i$, $\mathbf{H}_i^j$ and $\mathbf{W}_j$ are all treated as 2D images over the temporal and the spatial dimension and padded to be of the same size.

## References

[1] H. Hoffman, S. Schaal, A Computational Model of Human Trajectory Planning based on Convergent Flow Fields, in: Abstracts of the 37st Meeting of the Society of Neuroscience, 2007.

[2] D. D. Lee, H. S. Seung, Algorithms for Non-negative Matrix Factorization, in: T. K. Leen, T. G. Dietterich, V. Tresp (Eds.), Advances in Neural Information Processing Systems 13, MIT Press, 2001, pp. 556–562.

[3] S. Hellbach, J. Eggert, E. Koerner, H.-M. Gross, Basis Decomposition of Motion Trajectories Using Spatio-temporal NMF, in: C. Alippi, M. Polycarpou, C. Panayiotou, G. Ellinas (Eds.), Artificial Neural Networks ICANN 2009, volume 5769 of *Lecture Notes in Computer Science*, Springer, 2009, pp. 804–814.

[4] A. Cemgil, H. Kappen, D. Barber, A Generative Model for Music Transcription, IEEE Transactions on Audio, Speech, and Language Processing 14 (2006) 679–694.

[5] M. S. Lewicki, T. J. Sejnowski, Coding time-varying signals using sparse, shift-invariant representations, Advances in Neural Information Processing Systems 11 11 (1999).

[6] E. C. Smith, M. S. Lewicki, Efficient Auditory Coding, Nature 439 (2006) 978–82.

[7] T. Blumensath, M. Davies, Sparse and shift-Invariant representations of music, IEEE Transactions on Audio Speech And Language Processing 14 (2006) 50–57.

[8] R. Grosse, R. Raina, H. Kwong, A. Y. Ng, Shift-invariant sparse coding for audio classification, Cortex 9 (2007) 8.

[9] S. Lesage, R. Gribonval, F. Bimbot, Shift-invariant dictionary learning for sparse representations: extending K-SVD (2008).

[10] G. Monaci, F. T. Sommer, P. Vandergheynst, Learning Sparse Generative Models of Audiovisual Signals, European Conference on Signal Processing 2008 (2008) 1–5.

[11] C. Ekanadham, D. Tranchina, E. P. Simoncelli, Sparse Decomposition of Transformation-Invariant Signals With Continuous Basus Pursuit, in: 2011 IEEE International Conference on Acoustics Speech Signal Processing, pp. 4060 –4063.

[12] C. Ekanadham, D. Tranchina, E. P. Simoncelli, Recovery of Sparse Translation-Invariant Signals With Continuous Basis Pursuit, IEEE Transactions on Signal Processing 59 (2011) 4735–4744.

[13] B. Williams, M. Toussaint, A. Storkey, Modelling motion primitives and their timing in biologically executed movements, in: Advances in Neural Information Processing Systems 20.

[14] M. Rajapakse, L. Wyse, NMF vs. ICA for Face Recognition, in: International Symposium on Image and Signal Processing and Analysis, volume 2, pp. 605–610.

[15] J. Eggert, E. Koerner, Sparse Coding and NMF, in: 2004 International Joint Conference on Neural Networks, volume 4, pp. 2529–2533.

[16] J. Eggert, H. Wersing, E. Koerner, Transformation-invariant Representation and NMF, in: 2004 International Joint Conference on Neural Networks, volume 4, pp. 2535–2539.

[17] E. Smith, M. S. Lewicki, Efficient coding of time-relative structure using spikes, Neural Computation 17 (2005) 19–45.

[18] A. Naftel, S. Khalid, Classifying Spatiotemporal Object Trajectories Using Unsupervised Learning in the Coefficient Feature Space, Proceedings of the third ACM international workshop on Video Surveillance and Sensor Networks 12 (2006) 227–238.

[19] http://paco.psy.gla.ac.uk, Body movement library, 2011.

[20] N. Hurley, S. Rickard, Comparing Measures of Sparsity, IEEE Transactions on Information Theory 55 (2009) 4723–4741.

[21] D. Lee, H. Seung, Algorithms for non-negative matrix factorization, Advances in neural information processing systems 13 13 (2001) 621–624.

[22] C.-J. Lin, On the Convergence of Multiplicative Update Algorithms for Nonnegative Matrix Factorization, IEEE Transactions on Neural Networks 18 (2007) 1589–1596.

[23] N. Gillis, F. Glineur, Nonnegative factorization and the maximum edge biclique problem, CORE Discussion Papers 2008064, Universit catholique de Louvain, Center for Operations Research and Econometrics (CORE), 2008.

[24] A. Cichocki, A. H. Phan, R. Zdunek, Nonnegative Matrix and Tensor Factorizations: Applications to Exploratory Multi-way Data Analysis and Blind Source Separation, 2009.

## Vitae

Christian Vollmer is a Ph.D. student at the Neuroinformatics and Cognitive Robotics Lab at the Ilmenau University of Technology since 2009. His research is funded by Honda Research Institute Europe GmbH and focusses on the analysis of human motion in the field of human robot interaction. He studied Computer Science at Ilmenau University of Technology from 2004 to 2009.

Sven Hellbach is head of a junior research group at the University of Applied Sciences in Dresden. Between 2010 and 2011 he was associated with the Department of Biological Cybernetics at the University of Bielefeld. He received his Ph.D. after being a Ph.D. student at the Neuroinformatics and Cognitive Robotics Lab at the Ilmenau University of Technology from 2005 to 2010 being closely affiliated with the Honda Research Institute Europe GmbH. He studied Computer Science at the Ilmenau University of Technology from 2000 to 2005. His research focus is set to motion analysis, image processing and neural network in particular in the field of mobile robotics.

Julian Eggert studied physics at the Technical University of Munich, Germany, where he also received his Ph.D degree in theoretical biophysics (Prof. van Hemmen) in 2000. In 1999, he joined the Honda Research Institute in Offenbach, Germany, concentrating on biophysically realistic large-scale models for vision systems. Since 2003, he worked as a Senior Research Scientist and since 2007 as a Chief Scientist heading a research division at the Honda Research Institute (HRI) Europe GmbH at Offenbach, Germany. His interests include probabilistic modeling of cognitive systems, perception models for dynamic scene interpretation and gating in hierarchical neural networks via feedback and attention.

Horst-Michael Gross is full professor of Neuroinformatics and head of the Neuroinformatics and Cognitive Robotics Lab at the Ilmenau University of Technology. He received his doctoral degree in Computer Science in 1989 from the Ilmenau University of Technology. Among his current research interests are neural computing, cognitive robotics, and multi-modal human-robot interaction in real world environments.