# Non-Negative Kernel Sparse Coding for the Analysis of Motion Data

Babak Hosseini    Felix Hülsmann    Mario Botsch
Barbara Hammer

CITEC centre of excellence, Bielefeld University, Germany

Preprint of the publication [1], as provided by the authors.
The final publication is available at Springer via
`http://dx.doi.org/10.1007/978-3-319-44781-0_60`

**Abstract**

We are interested in a decomposition of motion data into a sparse linear combination of base functions which enable efficient data processing. We combine two prominent frameworks: dynamic time warping (DTW), which offers particularly successful pairwise motion data comparison, and sparse coding (SC), which enables an automatic decomposition of vectorial data into a sparse linear combination of base vectors. We enhance SC via efficient kernelization which extends its application domain to general similarity data such as offered by DTW, and its restriction to non-negative linear representations of signals and base vectors in order to guarantee a meaningful dictionary. We also implemented the proposed method in a classification framework and evaluated its performance on various motion capture benchmark data sets.

**Keywords:** Kernel sparse coding, motion analysis, classification, interpretable models, dynamic time warping.

## 1 Introduction

Ubiquitous sensors such as Microsoft's Kinect, video cameras, and motion capture systems cause an increasing availability of human motion data as digital signals. However, it remains a challenge how to automate semantic search in motion data bases, unless such data are labeled manually. In this contribution we investigate in how far natural priors such as sparsity allow an automatic extraction of semantically meaningful entities based on the given data alone.

We hypothesize that semantics is mirrored by recurring signals, which are present in semantically similar motion data, and it is possible to infer such signals from given data based on their property that they allow a particularly efficient description of the signals. We will rely on two techniques which have proven successful in such settings: 1) Dynamic Time Warping (DTW) that enables an efficient grouping of time series of different lengths according to their semantic similarity, incorporating invariance to small temporal shift and distortion [2]. 2) Sparse Coding (SC), which extracts a dictionary from a given data set and enables a sparse linear representation of the signals based thereon [3]. The resulting dictionary elements constitute an interface based on which semantic search becomes possible: signals which decompose into the same dictionary elements have a large semantic overlap.

Classical SC deals with vectorial data. To combine it with DTW, we will resort to a kernel version of SC [4]. Several approaches apply SC for motion data, but they provide unreasonable base functions and linear combinations due to negative coefficients [5]. We will extend kernel SC to a non-negative version, and we will demonstrate its accuracy for various motion capture benchmark data sets.

## 2 Non Negative Kernel Sparse Coding

Sparse coding for *vectorial data* represents every measurement $y^i$ from a set of measurements via a sparse representation $y^i = \mathbf{D}x^i$ with a dictionary matrix $\mathbf{D}$ of basic primitives, which are shared by all measurements, and sparse coefficients $x^i$, which describe how the observation $y^i$ is generated by the basic primitives. In our setting, we deal with motion data instead, i.e. data are given as *time-series* $Y^i = (y^i(1)...y^i(T)) \in (\mathbb{R}^n)^*$ of possibly varying length $T$. We assume that a kernel is given for such time series (such as the DTW kernel), denoted as $\mathcal{K}(Y^i, Y^j) = \Phi(Y^i)^\top \Phi(Y^j)$ with feature map $\Phi$. In the feature space, sparse coding problem becomes $\Phi(Y^i) = \Phi(\mathbf{D})x^i$, where $\Phi(\mathbf{D})$ is the dictionary matrix in the feature space.

Usually, the feature map $\Phi$ is not available, hence this problem cannot be solved directly. We follow the approach as proposed in [4]: we choose the dictionary as linear combinations of data $\Phi(\mathbf{D}) = \Phi(\mathbf{Y})\mathbf{A}$ with coefficient matrix $\mathbf{A}$. Often, $\mathbf{A}$ is chosen as an unconstrained matrix. However, we are interested in semantically meaningful features, i.e. dictionary elements should have the characteristics of motion signals and they should act as representatives for different motion groups. For this reason, we impose two constraints on $\mathbf{A}$ and $x^i$: The coefficient vector $x^i$, in addition to its sparseness, must be non-negative, such that motion signals are constructed from the dictionary elements as a meaningful mixtures of motions. For the same reason, the coefficient matrix $\mathbf{A}$ must be non negative, and the formation of meaningful groups of dictionaries is enforced by the sparsity of $\mathbf{A}$ by minimizing its $\mathrm{L}_1$ norm. Hence sparse coding becomes the following optimization problem, where $\mathbf{Y}$ refers to all observed sequences and $\mathbf{X}$ to its respective matrix of coefficients:

$$\begin{aligned} \min_{\mathbf{X},\mathbf{A}} \quad & \|\Phi(\mathbf{Y}) - \Phi(\mathbf{Y})\mathbf{A}\mathbf{X}\|_F^2 + \lambda\|\mathbf{A}\|_1^2 \\ \text{s.t} \quad & \|\mathbf{X}_i\|_0 \leq T, \quad a_{ij} \geq 0, \quad x_{ij} \geq 0 \quad \forall i,j \end{aligned} \tag{1}$$

$T$ limits the sparsity of the resulting SC. In order to solve this optimization problem (Eq.1), we use alternating optimization of the sparse coefficients and the dictionary. These two steps are realized by "Non-Negative Kernel Orthogonal Matching Pursuit (NNKOMP)" and "Non-Negative Kernel dictionary learning", described subsequently.

### 2.1 Non-Negative Kernel OMP

KNNOMP optimizes the coefficients $\mathbf{X}$ in (Eq.1) assuming a fixed dictionary characterized by coefficients $\mathbf{A}$. NNKOMP is based on the kernel OMP algorithm as proposed in [4], but enforcing non-negativity of the components. For this purpose, when adding non-zero coefficients in a greedy way in the kernel OMP, dictionary atoms with maximum *positive* correlation to the remaining residual error are selected. After selecting a new non-zero component based, coefficients $\mathbf{X}_i$ are optimized by the Non-negative least square algorithm (K-NNLS).

$$\min_{\mathbf{X}_i} \quad \|\Phi(\mathbf{Y}_i) - \Phi(\mathbf{Y})\mathbf{A}\mathbf{X}_i\|_2^2 \quad \text{s.t} \quad \mathbf{X}_i \geq 0, \quad \|\mathbf{X}_i\|_0 \leq T \tag{2}$$

For the K-NNLS method we use the active set "lsqnonneg" optimization algorithm from [6], and we kernelize the parts that calculate the intermediate solution pointand the gradient based on the variables selected in the passive set. As a result, the output of the K-NNLS would be used as the solution in the intermediate step of the NNKOMP algorithm.

### 2.2 Non-negative dictionary update

As the second part of our algorithm, we want to find the best dictionary $\Phi(\mathbf{Y})\mathbf{A}$ which minimizes (Eq.1) while using the obtained coefficients $\mathbf{X}$ as the output of NNKOMP in the previous section. Based on [4], the error function $\|\Phi(\mathbf{Y}) - \Phi(\mathbf{Y})\mathbf{A}\mathbf{X}\|_F^2$ can be re-formulated as:

$$\|\Phi(\mathbf{Y})\mathbf{E}_j - \Phi(\mathbf{Y})\mathbf{A}_j\mathbf{X}^j\|_F^2 \; ; \quad \mathbf{E}_j = (\mathbf{I} - \sum_{i \neq j}\mathbf{A}_i\mathbf{X}^i) \tag{3}$$

$\Phi(\mathbf{Y})\mathbf{E}_j$ is the reconstruction error using all the dictionary columns except $\mathbf{A}_j$ and along with corresponding coefficients $\mathbf{X}$ which were estimated by NNKMOP. Therefore, the dictionary can be updated through solving the (Eq.3) for each $\mathbf{A}_j$. As an important constraint we have to take into account that the optimal dictionary should be used along with non-negative coefficients $\mathbf{X}$. Accordingly we formulate (Eq.3) as the following alternating optimization set:

$$\min_{\mathbf{X}^j} \|\Phi(\mathbf{Y})\mathbf{E}_j - \Phi(\mathbf{Y})\mathbf{A}_j\mathbf{X}^j\|_F^2 \quad \text{s.t} \quad \mathbf{X}^j \geq 0 \tag{4}$$

$$\min_{\mathbf{A}_j} \|\Phi(\mathbf{Y})\mathbf{E}_j - \Phi(\mathbf{Y})\mathbf{A}_j\mathbf{X}^j\|_F^2 + \lambda\|\mathbf{A}_j\|_1^2 \quad \text{s.t} \quad \mathbf{A}_j \geq 0 \tag{5}$$

In order to solve (Eq.4), we used the large-scale non-negative least squares algorithm from [7] which can be easily extended to a kernel version that fits to (Eq.4).

#### 2.2.1 NN-Kernel FISTA:

In order to solve the optimization problem in (Eq.5), we devised the non-negative kernel FISTA algorithm (NN-K-FISTA) which is a combination of the projected gradient technique [8] and the Shrinkage-Thresholding method [9]. We kernelize [9], by calculating $f(\mathbf{A}_j)$ and $\nabla f(\mathbf{a})$ for the objective function

$f$ based on the Mercer kernel's inner product property; the shrinkage function is substituted with $\tau_l(x) = (x-l)(\text{sgn}(x-l)+1)/2$. As the last step in the dictionary update part, we normalize the dictionary coefficients such that $\|\Phi(\mathbf{Y})\mathbf{A}_j\|_2^2 = 1$.

## 2.3 Label Consistent NN-KSC classifier

The proposed non-negative kernel sparse coding framework will be used as a semantic encoding scheme for the observed motion data. In addition, we will evaluate the ability to base a classifier on top of the proposed coding scheme, as follows: We extend the label-consistent sparse coding as proposed in [10, 11]. In the latter, kernelized KSVD has been used. We assume a labeling is present, $\mathbf{H}$ is the label matrix of training data where $\mathbf{H}(i,j) = 1$ if $\mathbf{Y}_j$ is contained in class $i$. In addition, we choose the matrix $\mathbf{Q}$ such that $\mathbf{Q}_j = \mathbf{Q}_i$ if $\{\mathbf{Y}_j, \mathbf{Y}_i\}$ are in the same class. The objective of sparse coding is now extended to enforce that coefficients $\mathbf{X}_i$ and $\mathbf{X}_j$ are similar for data in the same class, weighted by $\alpha$. Further, base functions tend to accumulate coefficients for exemplars of one class, weighted by $\beta$.

$$\begin{array}{ll} \min\limits_{X,A} & \|\Phi(\mathbf{Y}) - \Phi(\mathbf{Y})\mathbf{AX}\|_F^2 + \alpha\|\mathbf{Q} - \mathbf{QAX}\|_F^2 + \beta\|\mathbf{H} - \mathbf{HAX}\|_F^2 + \lambda\|\mathbf{A}\|_1^2 \\ \text{s.t} & \|\mathbf{X}_i\|_0 \le T, \quad a_{ij} \ge 0, \quad x_{ij} \ge 0, \quad \forall i,j \end{array} \tag{6}$$

The optimization of this objective relates to a change of the kernel matrix as $\widetilde{\mathcal{K}}(\mathbf{Y}_i, \mathbf{Y}_i) = \mathcal{K}(\mathbf{Y}_i, \mathbf{Y}_j) + \alpha\langle\mathbf{Q}_i, \mathbf{Q}_j\rangle + \beta\langle\mathbf{H}_i, \mathbf{H}_j\rangle$. Using the new $\widetilde{\mathcal{K}}$ as the kernel function, (Eq.6) can be solved by the proposed NNKSC algorithm. The parameters $\alpha$ and $\beta$ control the trade-off between the reconstruction error and the classification accuracy. After optimizing the dictionary matrix $\mathbf{A}$, the NNKOMP (Eq.2) can be used to find sparse codes $\mathbf{X}$. This induces a labeling of the data via $l^i = \text{argmax}_j|\mathbf{H}(\cdot,j)\mathbf{AX}_i|$.
Furthermore, we are interested in having each column of $\mathbf{A}$ related to only one class of data. Doing so, we can partition $\mathbf{A}$ into separate class-specific dictionaries which will result in having specific prototypes and dictionary for each class of data. Therefore, in NN-K-FISTA algorithm the shrinkage-Threshold will be applied to only those elements of $\mathbf{A}_j$ related to data from classes with lower contributions in $\mathbf{A}_j$ (via updating the value of $\mathbf{HA}_j$ after (Eq.4)).

## 3 Datasets and Experiments

In this section we compare the proposed LC-NNKSC algorithm with other baselines on a few benchmarks. All datasets carry motion signals. Hence, first, we use the DTW algorithm to calculate a distance matrix $\mathbf{D}$ for the given samples. This is converted to a similarity matrix $\mathbf{K}$ using the Gaussian kernel $\mathcal{K}(x,y) = exp(-\frac{\|x-y\|^2}{\sigma})$. A valid Gram matrix results therefor by setting all its negative eigenvalues to zero (clipping). For the comparison, we choose the following methods:
**LC-K-KSVD:** We use a classification based on Kernel KSVD which has been proposed in [11]; this approach is closely related to the proposed NNKSC as regards its overall structure and objective.

**$k$NN:** We use the $k$-Nearest Neighbor classifier ($k = 3$) as a base line example, with which we classify the data samples based on the pairwise DTW distances.
**Kernel-Kmeans:** As another similar kernel based method, we apply the Kernel K-means clustering [12] to find $m$ (equal to size of dictionary $\mathbf{A}$) cluster prototypes. Afterward, the distance of each validation data $\mathbf{Y}_i$ to all prototypes would be calculated as $D_i = \text{diag}(E^\top \mathcal{K}(Y, Y)E) - 2\mathcal{K}(\mathbf{Y}_i, Y)E + \mathcal{K}(\mathbf{Y}_i, \mathbf{Y}_i)$, where $E$ is the normalized cluster assignment matrix based on [12]. After passing $D$ into a Gaussian function to convert it to a normalized similarity matrix and keeping the first $T$ biggest elements for each data, the result has a similar structure to $\mathbf{X}$ in the NNKSC algorithm. Then we feed the coefficients into a multi-class linear SVM in order to classify the validation data.
**Affinity Propagation:** We chose Affinity Propagation algorithm [13] as an approach which selects prototypes from the data samples in a clustering manner. There, the gram matrix would be used as the similarity matrix, and the class labels of validation data would be determined based on the closest neighboring prototype to each data sample.
**Kernel PCA:** As the last method for the comparison, we use the kernel-PCA approach from [14] to project the DTW based gram matrix $\mathcal{K}$ into $M$ dimension space resulting in data vectors $X$. We apply a multi-class linear SVM to classify the generated data vectors.

In order to prevent local optima, for each method, we repeat the same experiment with 10 different initial points (or initial dictionaries) and we choose the one with the best result for the comparison.

## 3.1 Evaluation Criteria

**Classification:** We measure the correct classification rate as the first metric to evaluate the performance of the algorithms. Each dataset is randomly split into train, test and the validation parts with 50%, 25% and 25% number of data respectively, and the learning process of the dictionary is stopped according to the increases in error curve of the test data. Finally, the classification accuracy and other measures are calculated based on the validation data.
**Reconstruction error:** Among the utilized methods, only LC-NNKSC and LC-KKSVD belong to a sparse coding framework and provide a reconstruction error (Eq.2) as a measure of their accuracy in a sparse representation of the data.
**Class based sparsity:** In addition, because another important concern of our framework is to provide sparse representation for the data, we also consider the level of sparseness for the coefficients $\mathbf{X}$. So in order to measure the sparseness in the classification framework we consider $SP_i$ as the number of non-zero elements in $\sum\limits_{k \in Class_i} |\mathbf{X}_k|$ for each class of the data, and we present the best and the worst $SP_i$ for each algorithm.
**Dictionary sparseness:** Furthermore, to study the dictionary interpretability, we calculate the relevance of each dictionary atom $d_j$ to the data classes. We can find the contribution of each data class in $\mathbf{D}_j$ via $c = \mathbf{H}\mathbf{A}_i$ where $\mathbf{H}$ is the class label matrix as in (Eq.6). Then the dictionary sparseness is measured as $DS = c_x / \sum c_k$ where $c_x$ is the biggest element in $c$.

## 3.2 Datasets

**CMU Motion Dataset**: We use the Human motion capture dataset from the CMU graphics laboratory [15], which was captured by aVicon infra-red system. We combined the movement data of subject 86 from the dataset which is a combination of 9 different types of human movements such as “walking”,”running”, “clapping”,... . Then the data is segmented in order to brake down the long movements into smaller segments as single periods of each type of motion. Consequently, we obtain 9 classes of data with 10 sample per class, and for implementing LC-NNKSC we used $\alpha = 1$ and $\beta = 5$.
**Cricket Umpire’s Signals:** For our classification experiment we use Cricket Umpire’s Signal data provided in [16]. This dataset contains 180 sample of data from 12 different classes of umpire signals related to the cricket game. In order to perform the sparse coding classification we choose $\alpha = 0.5$ and $\beta = 1$.
**Articulatory Words:** The articulatory words dataset is the facial (ex. lips and tongue) movement signals captured via EMA sensors [17]. The dataset is used to categorize 25 classes of different words uttered by the subjects in total 575 sample of data. For this dataset we choose $\alpha = 0.2$ and $\beta = 0.5$.
**Squat dataset:** The squat dataset is gathered in our institute as a part of the large-scale intelligent coaching project. The data is a set of squat movements performed by three sport coaches while being captured by the optical MOCAP system [18]. Each squat is segmented into three movement primitives ”preparation”, ”going down” and ”comming up”, which generates 87 sample of data and 9 class labels together with the coach labels. Classification of this dataset is performed while using 1 and 0.2 as the $\alpha$ and $\beta$ respectively.

## 3.3 Classification Results

For all the 4 dataset we choose the number of dictionary elements $\mathbf{A_i}$ as twice as the number of total classes. As a rule of thumb, we assume the data in each class can be reconstructed with a low error using only 2 atoms related to that class. We use the same value as the number of prototypes and the mapping dimension in K-Kmeans and K-PCA respectively. Also for the NNKSC and the LC-KKSVD algorithms we choose the sparsity limit $T = 4$ to see how the algorithm is going to use these 2 additional redundancy levels for the dictionary learning and the reconstruction.
In Table.1, the classification result are provided. We can see that for all datasets the proposed algorithm achieved the highest classification accuracy among the evaluated methods; however for Cricket and Words datasets the LC-KKSVD provided similar accuracy rates to LC-NNKSC (83.33 % and 97.33 %) while having smaller reconstruction errors due to the non-negative restrictions. Also in some of the datasets, the affinity propagation and the kNN managed to obtain performance levels equal to the proposed method, for example both have 100 % classification accuracy for CMU dataset; nevertheless they do not provide any reconstruction model for the data in comparison to the sparse coding framework.
Table.2 brings the the sparsity analysis of the results, as the best and the worst measures (bDS, wDS) for the relevance of dictionary elements to the classes, as well as the best and worst number of class based sparsity (bSP, bSP). Ac-

Table 1: Classification accuracy(%) and the reconstruction error (%) from applying the selected methods on the chosen datasets

| | CMU | | Cricket Signals | | Articulatory Words | | Squat | |
|---|---|---|---|---|---|---|---|---|
| | Acc | Rec. Err | Acc | Rec. Err | Acc | Rec. Err | Acc | Rec. Err |
| LC-NNKSC | 90.91 | 4.17 | 83.33 | 11.07 | 97.33 | 14.52 | 100 | 0.14 |
| LC-KKSVD | 86.36 | 7.44 | 83.33 | 10.1 | 97.33 | 7.8 | 85 | 3.4 |
| K-Means+SVM | 68 | – | 56.25 | – | 90 | – | 81 | – |
| Affinity P. | 90.1 | – | 68.75 | – | 92 | – | 100 | – |
| K-PCA+SVM | 50 | – | 56.25 | – | 60.66 | – | 37 | – |
| kNN | 86.36 | – | 79.16 | – | 96.66 | – | 100 | – |

Table 2: The best and worst class based sparseness (bSP and wSP), and the best and worst dictionary sparseness (bDS(%) and wDS(%)) for the different selected approaches

| | CMU | | | | Cricket Signals | | | | Articulatory Words | | | | Squat dataset | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | bSP | wSP | bDS | wDS | bSP | wSP | bDS | wDS | bSP | wSP | bDS | wDS | bSP | wSP | bDS | wDS |
| LC-NNKSC | 1 | 2 | 100 | 100 | 1 | 4 | 100 | 100 | 1 | 3 | 100 | 98.1 | 1 | 1 | 100 | 100 |
| LC-KKSVD | 5 | 9 | 100 | 76 | 5 | 13 | 100 | 44 | 5 | 16 | 100 | 56 | 3 | 8 | 100 | 87 |
| Affinity P. | 4 | 6 | – | – | 6 | 4 | – | – | 5 | 11 | – | – | 4 | 5 | – | – |
| K-Means | 4 | 17 | 100 | 50 | 5 | 27 | 100 | 16 | 5 | 50 | 100 | 50 | 4 | 12 | 100 | 60 |

cording to the table.2, LC-NNKSC provide models for the datasets with better sparseness regarding both the dictionary atoms and the class data reconstruction. For all datasets, it defines each dictionary atom using the data of a single class which results in almost 100 % dictionary sparseness. For the squat data the algorithm managed to reconstruct the data of each class using only one specific atom (wSP=bSP=1), meaning that only half of the dictionary is needed to model this data with NNKSC. Also, due to the value of wSP in Cricket and Words data (4 and 3 respectively), apparently there exist classes which require more than 2 dictionary atoms to be reconstructed and categorized efficiently.
The LC-KKSVD too has a high classification accuracy for Cricket and Words data, but this performance is lower than Affinity Propagation in the other 2 datasets. Furthermore, from the sparseness point of view, it is outperformed even by Affinity propagation by providing lower class based sparsity.

## 4 Conclusion

In this paper we presented a non-negative kernel based sparse coding approach for modeling and classification of motion data. According to the results, the non-negative approach provides much sparser representation for the data comparing to the conventional Kernel SC method, using fewer number of prototypes to reconstruct the motion signals. Additionally, where it is possible the LC-NNKSC approach forces dictionary elements to be created using positive linear combina-

tion of data only from individual classes. Doing so, the obtained dictionary can be easily broken down to class based dictionaries as separate prototype-based models for each class of data. In addition these sub-dictionaries can be used as a warm start in further classification tasks even when there is different combination of classes. All together, the LC-NNKSC classifier provides dictionary prototypes and sparse coefficients which are more class based consistent and makes it possible to have individual models for reconstruction of each class of data as well as for its classification.
Based on the strength of this method in constructing prototype based models for the motion data, there is a considerable potential for future works on the clustering and designing generative models of motion data using this framework or its variants.

## 5 Acknowledgment

This research was supported by the Cluster of Excellence Cognitive Interaction Technology 'CITEC' (EXC 277) at Bielefeld University, which is funded by the German Research Foundation (DFG).

## References

[1] B. Hosseini, F. Hülsmann, M. Botsch, and B. Hammer, "Non-negative kernel sparse coding for the analysis of motion data," in *International Conference on Artificial Neural Networks.* Springer, 2016, pp. 506–514.

[2] M. Shokoohi-Yekta, B. Hu, H. Jin, J. Wang, and E. Keogh, "On the non-trivial generalization of dynamic time warping to the multi-dimensional case." in *SDM*, 2015.

[3] M. Aharon, M. Elad, and A. Bruckstein, "K-SVD: An algorithm for designing overcomplete dictionaries for sparse representation," *IEEE Transactions on Signal Processing*, vol. 54, no. 11, pp. 4311–4322, 2006.

[4] H. Van Nguyen, V. M. Patel, N. M. Nasrabadi, and R. Chellappa, "Design of non-linear kernel dictionaries for object recognition," *IEEE Transactions on Image Processing*, vol. 22, no. 12, pp. 5123–5135, 2013.

[5] T. Kim, G. Shakhnarovich, and R. Urtasun, "Sparse coding for learning interpretable spatio-temporal primitives," in *Advances in neural information processing systems*, 2010, pp. 1117–1125.

[6] L. Shure, "Brief history of nonnegative least squares in matlab," *Blog available at: http://blogs. mathworks. com/loren*, 2006.

[7] H. V. B. Mark and R. K. Michael, "Fast algorithm for the solution of large-scale non-negativity-constrained least squares problems," *Journal of Chemometrics*, vol. 18, no. 10, pp. 441–450, 2004. [Online]. Available: http://dx.doi.org/10.1002/cem.889

[8] C.-J. Lin, "Projected gradient methods for nonnegative matrix factorization," *Neural computation*, vol. 19, no. 10, pp. 2756–2779, 2007.

[9] A. Beck and M. Teboulle, "A Fast Iterative Shrinkage-Thresholding Algorithm for Linear Inverse Problems," *Society*, vol. 2, no. 1, pp. 183–202, 2009.

[10] Z. Jiang, Z. Lin, and L. S. Davis, "Label consistent k-svd: Learning a discriminative dictionary for recognition," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 35, no. 11, pp. 2651–2664, 2013.

[11] Z. Chen, W. Zuo, Q. Hu, and L. Lin, "Kernel sparse representation for time series classification," *Information Sciences*, vol. 292, pp. 15–26, 2015.

[12] J. Shawe-Taylor and N. Cristianini, *Kernel methods for pattern analysis.* Cambridge university press, 2004.

[13] R. Guan, X. Shi, M. Marchese, C. Yang, and Y. Liang, "Text clustering with seeds affinity propagation," *Knowledge and Data Engineering, IEEE Transactions on*, vol. 23, no. 4, pp. 627–637, 2011.

[14] B. Schölkopf, A. Smola, and K.-R. Müller, "Kernel principal component analysis," in *Artificial Neural NetworksICANN'97.* Springer, 1997, pp. 583–588.

[15] *Carnegie-Mellon mocap database*, Carnegie Mellon Univ. Std., Mar. 2007. [Online]. Available: http://mocap.cs.cmu.edu/

[16] M. H. Ko, G. West, S. Venkatesh, and M. Kumar, "Online context recognition in multisensor systems using dynamic time warping," in *Intelligent Sensors, Sensor Networks and Information Processing Conference, 2005. Proceedings of the 2005 International Conference on.* IEEE, 2005, pp. 283–288.

[17] J. Wang, A. Samal, and J. R. Green, "Preliminary test of a real-time, interactive silent speech interface based on electromagnetic articulograph," 2014.

[18] T. Waltemate, F. Hülsmann, T. Pfeiffer, S. Kopp, and M. Botsch, "Realizing a low-latency virtual reality environment for motor learning," in *Proceedings of the 21st ACM Symposium on Virtual Reality Software and Technology.* ACM, 2015, pp. 139–147.

## 6 Appindex

### 6.1 Kernel Non-Negative OMP

Fig.6.1 presents our proposed NNKOMP algorithm.

```
Task: Solving $\min_x \|\Phi(z) - \Phi(\mathbf{Y})\mathbf{A}x_i\|_2^2$ with $x_i \geq 0$ and $\|\vec{x}_i\|_0 \leq T$
Input: Data sample z, dictionary coefficient A, sparseness limit T, kernel
       K
Output: non-negative sparse code $x$ with $T$ non-zero elements
Initialization: $x = 0, I = 0$
Loop:
1   $\tau_i = max([\mathcal{K}(z,Y) - A_I x \mathcal{K}(Y,Y)]a_i \ ,0), \ \forall i \notin I$ ;
2   $i_{max} = \arg\max_i |\tau_i|, \ \forall i \notin I$ ;
3   $I = I \cup i_{max}$ ;
4   Solving $\min_x \|\Phi(z) - \Phi(\mathbf{Y})\mathbf{A}_I x\|_2^2$ s.t $x \geq 0$ using KNNLS;
5   Stop if $\|x\|_0 = T$ , otherwise go to step 1 ;
```

Figure 1: The KNNOMP algorithm

To kernelize the Non-Negative LS method from [6], we changed the parts that calculate the intermediate solution point $s^P$ and the gradient $w$ as in Eq.7, where $A^P_{I_s}$ is related to the variables selected in the passive set $P$. As a result, the output of the K-NNLS would be used as the solution $x_s$ in step 4 of the algorithm in Fig.6.1.

$$\begin{aligned} s^P &= [(A^P_{I_s})^\top \mathcal{K}(Y,Y) A^P_{I_s}]^{-1} (A^P_{I_s})^\top \mathcal{K}(z,Y)^\top b \\ w &= A^\top_{I_s} [\mathcal{K}(z,Y)^\top - \mathcal{K}(Y,Y) A_{I_s} x] \end{aligned} \tag{7}$$

### 6.2 NN-Kernel FISTA:

Our proposed NN-K-FISTA algorithm is shown in (Fig.6.2).

```
Task: Solving $\min_a f(a) + \|a\|_1^2$ s.t $a \geq 0$
Input: function $f(a, \mathcal{K}, E)$, $\lambda$
Output: non-negative sparse dictionary atom $a$ which fits into (Eq.5)
Initialization: $k = 0, \ t = 1, \ 0 < \eta < 1, \ 0 < \alpha, \ \delta$
Step K: $(k \geq 1)$ , find the first possible $i \in \mathbb{N}$ such that with
        $\alpha_k = \eta^i \alpha_{k-1}$:
1   $a^{k+1} = \tau_{\alpha_k \lambda}(a^k - \alpha_k \nabla f(a^k))$;
2   $f(a^{k+1}) - f(a^k) > (a^{k+1} - a^k)\nabla f(a^k) - \|a^{k+1} - a^k\|_2^2/(2\alpha)$ ;
3   $t_{k+1} = (1 + \sqrt{1 + 4t_k^2})/2$ ;
4   Stop if $f(a^{k+1}) < \delta$, otherwise $a^{k+1} = a^{k+1} + (a^{k+1} - a^k)(t_k - 1)/t_{k+1}$
```

Figure 2: The NNK-FISTA algorithm

In this algorithm $f(a_j)$, $\nabla f(a)$ and $\tau_\alpha(x)$ would be calculated as:

$$\begin{array}{ll} \|\Phi(\mathbf{Y})E_j - \Phi(\mathbf{Y})a_j x_R^j\|_F^2 & = \mathbf{tr}(E_j - a_j x_R^j)^\top \mathcal{K}(Y,Y)(E_j - a_j x_R^j)) \\ \nabla f(a_j) & = -2\mathcal{K}(Y,Y)(E_j - a_j x_R^j){x_R^j}^\top \\ \tau_l(x) & = (x-l)(sign(x-l)+1)/2 \end{array} \quad (8)$$

The dictionary atoms $a_j$ will be normalized as:

$$a_j = a_j/\sqrt{a_j^\top \mathcal{K}(Y,Y)a_j}, \quad x_R^j = x_R^j\sqrt{a_j^\top \mathcal{K}(Y,Y)a_j} \quad (9)$$