Online Kernel Slow Feature Analysis for Temporal Video Segmentation and Tracking

Stephan Liwicki, *Student Member, IEEE*, Stefanos Zafeiriou, *Member, IEEE*, and Maja Pantic, *Fellow, IEEE*

***Abstract*—Slow feature analysis (SFA) is a dimensionality reduction technique which has been linked to how visual brain cells work. In recent years, SFA was adopted for computer vision tasks. In this paper, we propose an exact kernel SFA (KSFA) framework for positive definite and indefinite kernels in Krein space. We then formulate an online KSFA which employs a reduced set expansion. Finally, by utilizing a special kind of kernel family, we formulate exact online KSFA for which no reduced set is required. We apply the proposed system to develop a SFA-based change detection algorithm for stream data. This framework is employed for temporal video segmentation and tracking. We test our setup on synthetic and real data streams. When combined with an online learning tracking system, the proposed change detection approach improves upon tracking setups that do not utilize change detection.**

***Index Terms*—Slow feature analysis, online kernel learning, change detection, temporal segmentation, tracking**

I. Introduction

SLOW FEATURE ANALYSIS (SFA) originates from theories in neural networks [1], and extensive studies in neural science found similarities between SFA and the properties of brain cells in the visual cortex [1], [2]. More recently, SFA found its way into computer vision [3], [4], [5], [6]. Here, SFA is employed as an unsupervised learning technique for dimensionality reduction of temporally arranged data such as video. In particular, it extracts an orthogonal subspace from the input data, similarly to principal component analysis (PCA). In contrast to PCA however, SFA considers the temporal information to find the most descriptive components that vary slowest over time [1]. The intuition behind SFA is linked to the assumption that the information contained in a signal changes not suddenly, but slowly. Note, a signal generally contains high variation (caused by noise), nonetheless, it is the seldom varying features that mark the separation between informative changes. SFA extracts these features, as it selects the important attributes which change least over time.

Manuscript received May 9, 2014; revised September 30, 2014 and February 9, 2015; accepted April 1, 2015. Date of publication May ??, 2015; date of current version May ??, 2015. This work is funded by the EPSRC project EP/J017787/1 (4D-FAB). The work by S. Zafeiriou is also partially supported by the EPSRC project EP/L026813/1 Adaptive Facial Deformable Models for Tracking (ADAManT). The work by M. Pantic is further supported by the European Community Horizon 2020 [H2020/2014-2020] under grant agreement no. 645094 (SEWA). S. Liwicki was funded by an EPSRC studentship and an INTEL fellowship.

S. Liwicki was with the Department of Computing, Imperial College London, United Kindom, when this work was produced. He is now with the Department of Engineering Science, University of Oxford, United Kingdom. (email: stephan.liwicki@eng.ox.ac.uk)

S. Zafeiriou and M. Pantic are with the Department of Computing, Imperial College London, United Kingdom. (email: {s.zafeiriou, m.pantic}@ic.ac.uk)

M. Pantic is also with the Faculty of Electrical Engineering, Mathematics and Computer Science, University of Twente, The Netherlands.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object dentifier ??/??????

A. Related Research

1) SFA in Computer Vision: Inspired by the slowness principle of neural networks [7], [5] employs SFA for invariant localization and recognition. This work extract slow features hierarchically. Multiple areas of the input images are analyzed for slow features individually. These features are then combined and reused as input signal for the higher levels of the hierarchy. In total, 4 SFA layers are introduced. The final layer's features are then used for regression or classification.

In [3], SFA's properties are exploited to segment video data temporally. The aim of this study is to extract unknown different actions in an image sequence. The individual segments are thought to be the activities in the examined video. After performing SFA on the complete video, the authors determine whether a split of the sequence is required. The decision is based on the median of change in the slow features. For the separation, the frame with the largest change is utilized as split position. SFA is once more performed on the resulting videos, and the process is repeated until no further split is necessary.

Another example of SFA applied to temporal segmentation is [6]. After formulating SFA as probabilistic model and solving for expectation maximization, the authors are able to extract the temporal phases of facial expressions through SFA. Their offline algorithm successfully extracts onset (beginning), apex (duration) and offset (ending) of facial action units.

2) Online Learning: The literature above reveals SFA's ability for temporal video segmentation in image sequences, although only in offline setups. With an ever increasing importance of realizing online applications, incremental learning methods have become a popular research topic [8], [9], [10], [11]. In particular, real-time object tracking has been shown to benefit from online models [9], [11], [12]. Commonly in online learning, however, the appearance model used to describe the tracked object (target) is susceptible to drift [11]. With change detection and SFA in particular a tracking system can detect when drift is likely (during changes). In this work, we want to combine online tracking systems with an incremental approach to SFA to improve drift suppressants.

An online learning system to find slow features is required for the detection of changes in video streams. All methods presented thus far require the complete video *a priori*. To the best of our knowledge, the only incremental version of SFA (IncSFA) is proposed in [4]. As SFA can be solved in two stages *via* PCA and minor component analysis (MCA), IncSFA utilizes a combination of the candid covariance-free incremental PCA (CCIPCA) in [13], and the sequential extraction of minor components in [14]. CCIPCA is based on statistical efficiency and incrementally estimates the data distribution by means of scale and mean. Consequently, IncSFA learns a rough estimation of the true slow features. Furthermore,

IncSFA is designed to learn from multiple complete videos of similar motions, rather than the data points of a single video. Therefore, IncSFA is very suitable for learning features of sequences that reoccur in its entirety, *e.g.* complete activities. However, online learning for change detection, where a scene is seen only once, is not supported by this approach.

3) Non-linear Features: SFA is originally designed for data signals with quadratic expansions. In this case, since SFA is a statistical component analysis, it is quite likely to suffer from the curse of dimensionality. Hierarchical SFA, as in [5], somewhat improves this situation, but important properties could be missed. More recently, [15] introduces kernel SFA (KSFA) for standard positive definite kernels. Similarly to a quadratic expansion, kernel methods allow for a mapping of features into a high-dimensional feature space. However, this feature space is never required to be computed explicitly, as it is realized by the kernel which represents the dot-product of two samples in such space. The computation of kernels is often more efficient than non-linear expansions – at least in the amount of computer memory used. Furthermore, kernel functions allow for more flexible feature spaces. Typically, as in [15], the selection of standard kernels such as Gaussian RBFs (GRBFs) is encountered. These kernels seldom utilize the domain-dependent property of the data. In recent work we present a kernel specifically designed for image gradients [12]. Inspired by its success in tracking and recognition, we now wish to apply this kernel to SFA. An important aspect of our kernel is that it is not positive definite and, thus, the appropriate space in which our kernel can represent a dot-product is a Krein space. Therefore, an extension of KSFA into Krein space is required.

4) Online Kernel Learning: Typically, the classification or regression functions of online kernel methods are written as a weighted sum of kernel combinations, taken from a set of stored instances, usually referred to as support or reduced set. At each step a new instance is fed to the algorithm and depending on the update criterion the algorithm adds the instance to the support set. One of the major challenges in online learning is that the support set may grow arbitrarily large over time [10], [16], [17].

Many techniques that try to bound this set have been presented in the literature [18], [19], [20], [21]. For example, [22] propose online kernel algorithms for classification, regression and novelty detection using a stochastic gradient descent algorithm in the Hilbert space defined by the kernel. In order to avoid the arbitrary growth of the support set, the authors adopt simple truncation and shrinking strategies. In [23] an online regression algorithm which uses an alternative model reduction criterion is proposed. Instead of using a sparsification procedure, the increase in the number of variables is controlled by a coherence parameter, a fundamental quantity that characterizes the behavior of dictionaries in sparse approximation problems.

In contrast to reducing the support set needed, in [24] the kernel function is approximate by a finite number of functions which can be explicitly calculated. Finally, [10], obtain a reduced set expansion to bound the samples in the support set. Such samples are referred to as pre-images, and they are optimized to fulfill the kernel method.

Notice, all methods above are *approximations* in one form or another, which inadvertently reduces the accuracy over time. For example, while a reduced set expansion with pre-images allows for arbitrary kernels, the computation of pre-images has drawbacks: (1) the reduced set representation provides only an approximation to the exact solution and (2) the optimization problem for finding the expansion inevitably increases the complexity of the algorithm. In contrast, a specific kernel family can be applied which allows us to formulate a special case of *exact* online KSFA without pre-image computation or kernel approximations. One robust, domain-specific kernel function of such class is given by [12].

5) Unsupervised Video Segmentation: Some related works in the broader area of unsupervised video segmentation are [25], [26], [27]. In [25] a method for clustering events is proposed. Their work is only suitable for offline processing and requires the number of clusters *a priori*. This is also the case for the clustering algorithm in [26]. A method for joint segmentation and classification of human actions in video is proposed in [27]. Their method is supervised, *i.e.* a model for human actions is learned from a set of labeled training samples. Then, given a testing video with a continuous stream of human activities, the algorithm in [27] finds the globally optimal temporal segmentation (*i.e* the change points between actions) and class labels. Our methodology takes a different direction. In particular, we detect the temporal changes in video streams online. We do not require the number of clusters, nor train to a predefine set of examples. Thus, the methods in [26], [27], [28] constitute excellent post-processing tools for clustering or classifying the events.

B. Contributions

First, we introduce a general kernel framework for SFA with positive definite and indefinite kernels. Furthermore, we formulate an online learning algorithm for the proposed KSFA which computes the slow components at each given time-step incrementally. We emphasize, in contrast to IncSFA [4] which learns from multiple videos, our incremental version of SFA is designed to learn from individual data points of a single data stream. Then, we extend our learning framework to formulate an exact incremental kernel KSFA, which uses the kernel family of [12]. Finally, we develop the first SFA-based real-time change detection algorithm which we employ for temporal video segmentation and visual tracking. In summary, our contributions are as follows:

1) We propose exact KSFA with arbitrary derivative approximations for any kernel in Hilbert or Krein space.
2) We formulate an accurate framework for general online KSFA for which we apply a reduced set expansion in Krein space. In contrast to [4], our online learning system computes the slow features at each time-step, from individual data points of a single video sequence.
3) We propose incremental KSFA that does not require a reduced set expansion, exploiting the properties of our domain-specific kernel in [12].
4) We introduce SFA-based online change detection which we apply to temporal video segmentation and tracking.

In [29], we introduce KSFA in Krein space and develop an incremental KSFA algorithm for our special, domain-specific kernel in [12]. We implement SFA's change detection algorithm, and apply it to temporal video segmentation. In this paper we extend our work, and introduce an incremental KSFA for arbitrary kernels in Krein or Hilbert space. Notice, the setup in [29] depends on a version of the scatter matrix which is not available in general kernel methods. With our new algorithm, we propose a true scatter-matrix-independent version of SFA, as we use the kernel matrix throughout. We also introduce a tracking framework with change detection.

C. Notation

We summaries some mathematical notation for the readers convenience in table I. See text for details.

TABLE I
SUMMARY OF MATHEMATICAL NOTATION.

$\mathbb{R}$	space of real numbers
$\mathbb{C}$	space of complex numbers
$\mathcal{H}$	infinite dimensional Hilbert space
$\mathcal{K}$	non-positive definite Krein space
$\lvert\mathcal{K}\rvert$	associated Hilbert space of $\mathcal{K}$
$\mathbf{J}$	fundamental symmetry of $\mathcal{K}$
$\mathbf{K}$	kernel matrix
$k(.)$	kernel function
$\phi(.)$	implicit kernel mapping
$\langle .,. \rangle$	inner product
$\dot{\mathbf{X}}$	derivative of $\mathbf{X}$
$\tilde{\mathbf{X}}$	centralized matrix of $\mathbf{X}$
$\mathbf{C}$	helper matrix to compute mean
$\mathbf{M}$	helper matrix to compute centralized matrix
$\boldsymbol{\mu}$	vector of mean values
$(.)^\mathsf{T}$	matrix transpose
$\overline{(.)}$	complex conjugate
$(.)^\mathsf{H}$	complex conjugate transpose
$(.)^*$	conjugate transpose with fundamental symmetry
$\oplus$	direct sum of spaces
$\odot$	element-wise multiplication

II. SFA WITH INDEFINITE KERNELS

We propose a general kernel SFA. Contrasting [15] indefinite kernels are handled. Section II-A introduces the theory of positive definite and indefinite kernels. SFA and its kernelized optimization is presented in section II-B. Finally, section II-C introduces our algorithm for SFA in Hilbert and Krein space.

A. Kernel Functions in Hilbert and Krein Space

A Hilbert space $\mathcal{H}$ is a complete vector space which is defined by an inner product onto complex space $\langle .,. \rangle_\mathcal{H} : \mathcal{H} \times \mathcal{H} \rightarrow \mathbb{C}$, which induces a norm and, thus, a metric. Generally, $\mathcal{H}$ is an implicitly defined infinite dimensional hyperspace. A kernel function $k_\mathcal{H} : \mathbb{C}^P \times \mathbb{C}^P \rightarrow \mathbb{C}$ defines the unknown mapping $\phi_\mathcal{H} : \mathbb{C}^P \rightarrow \mathcal{H}$ which transforms the original data into Hilbert space, and the inner product is realized by

$$\langle \phi_\mathcal{H}(\mathbf{y}), \phi_\mathcal{H}(\mathbf{x}) \rangle_\mathcal{H} = \phi_\mathcal{H}(\mathbf{x})^\mathsf{H} \phi_\mathcal{H}(\mathbf{y}) = k_\mathcal{H}(\mathbf{x}, \mathbf{y}) \quad (1)$$

where $\mathbf{x}$ and $\mathbf{y}$ are members of the original space $\mathbb{C}^P$ and $(.)^\mathsf{H}$ indicates the Hermitian transpose. For many applications, $k_\mathcal{H}$ is employed and $\phi_\mathcal{H}$ is never explicitly required [15], [30], [31]. For example, the linear distance between two samples in the non-linear Hilbert space is given *via* the kernel as

$$\begin{aligned} l^2(\phi_\mathcal{H}(\mathbf{x}), \phi_\mathcal{H}(\mathbf{y})) &= k_\mathcal{H}(\mathbf{x}, \mathbf{x}) - k_\mathcal{H}(\mathbf{x}, \mathbf{y}) \\ &\quad - k_\mathcal{H}(\mathbf{y}, \mathbf{x}) + k_\mathcal{H}(\mathbf{y}, \mathbf{y}). \end{aligned} \quad (2)$$

The following properties are important for the inner product of a Hilbert space ($\forall \mathbf{x}, \mathbf{y}, \mathbf{z} \in \mathcal{H}, \forall a, b \in \mathbb{C}$):

$$\langle \mathbf{x}, \mathbf{y} \rangle_\mathcal{H} = \overline{\langle \mathbf{y}, \mathbf{x} \rangle_\mathcal{H}} \quad (3)$$

$$\langle a\mathbf{x} + b\mathbf{y}, \mathbf{z} \rangle_\mathcal{H} = a\langle \mathbf{x}, \mathbf{z} \rangle_\mathcal{H} + b\langle \mathbf{y}, \mathbf{z} \rangle_\mathcal{H} \quad (4)$$

$$l(\mathbf{x}, \mathbf{z}) \leq l(\mathbf{x}, \mathbf{y}) + l(\mathbf{y}, \mathbf{z}) \quad (5)$$

where $\overline{(.)}$ denotes the complex conjugate. Therefore, if a positive definite kernel is selected, the space is Hilbert.

Similar to Hilbert spaces, kernels in Krein space define an implicit mapping $\phi_\mathcal{K} : \mathbb{C}^P \rightarrow \mathcal{K}$ from feature space onto $\mathcal{K}$, and provide the inner product $\langle .,. \rangle_\mathcal{K} : \mathcal{K} \times \mathcal{K} \rightarrow \mathbb{C}$ such that

$$\langle \phi_\mathcal{K}(\mathbf{y}), \phi_\mathcal{K}(\mathbf{x}) \rangle = k_\mathcal{K}(\mathbf{x}, \mathbf{y}) \quad (6)$$

for $\mathbf{x}, \mathbf{y} \in \mathbb{C}^P$. They also satisfy the analogous properties for eq. (3) and eq. (4). However, when the distance is similarly to eq. (2), the triangular inequality may not hold [32], [33].

A Krein space is composed out of two Hilbert spaces, $\mathcal{K}_-$ and $\mathcal{K}_+$, such that $\mathcal{K} = \mathcal{K}_- \oplus \mathcal{K}_+$, where $\oplus$ denotes the direct sum (*i.e.* $\mathcal{K}_-$ and $\mathcal{K}_+$ are orthogonal in terms of $\langle .,. \rangle_\mathcal{K}$). Thus, two orthogonal projections can be extracted from $\mathcal{K}$: $\mathbf{F}_+$ onto $\mathcal{K}_+$ and $\mathbf{F}_-$ onto $\mathcal{K}_-$, known as fundamental projections. By use of the fundamental symmetry $\mathbf{J} = \mathbf{F}_+ - \mathbf{F}_-$, an associated Hilbert space $\lvert\mathcal{K}\rvert$ is found. We write the relationship between $\mathcal{K}$ and $\lvert\mathcal{K}\rvert$ in terms of a "conjugate", as

$$\mathbf{x}^*\mathbf{y} \triangleq \langle \mathbf{y}, \mathbf{x} \rangle_\mathcal{K} = \mathbf{x}^\mathsf{H}\mathbf{J}\mathbf{y} = \langle \mathbf{J}\mathbf{y}, \mathbf{x} \rangle_{|\mathcal{K}|} \quad (7)$$

where $\mathbf{x}, \mathbf{y} \in \mathcal{K}$. That is, $\mathcal{K}$ can be turned into its associated Hilbert space $\lvert\mathcal{K}\rvert$ by using its positive definite inner product $\langle .,. \rangle_{|\mathcal{K}|}$, as $\langle \mathbf{x}, \mathbf{y} \rangle_{|\mathcal{K}|} = \langle \mathbf{x}, \mathbf{J}\mathbf{y} \rangle_\mathcal{K}$. In finite dimensional Krein spaces (*i.e.* $\dim(\mathcal{K}_+) + \dim(\mathcal{K}_-) < \infty$, where $\dim(.)$ finds the dimensionality), the fundamental symmetry is given by

$$\mathbf{J} = \begin{bmatrix} \mathbf{I}_{\dim(\mathcal{K}_+)} & \mathbf{0} \\ \mathbf{0} & -\mathbf{I}_{\dim(\mathcal{K}_-)} \end{bmatrix} \quad (8)$$

where $\mathbf{I}_n$ is the $n \times n$ identity, and $\mathbf{0}$ implies zero padding.

Kernels in Hilbert or Krein spaces are important, as they provide feature representations for dissimilarities in non-linear spaces. Plenty successful applications exist in the literature for Hilbert kernels [15], [30], [31]. The use of Krein kernels is more seldom [33]. In [12], we introduced an indefinite Krein kernel for tracking and recognition. It is important to note here, that methods which employ Krein spaces are also valid for Hilbert spaces, as $\mathcal{K} \supset \mathcal{H}$. Thus, in the following we consider indefinite kernels in Krein spaces only.

B. SFA's Optimization Problem

Given N sequential observation vectors as columns of $\mathbf{X} = [\mathbf{x}_1 \cdots \mathbf{x}_N] \in \mathbb{C}^{P \times N}$, SFA finds a descriptive output $\mathbf{o}_n$ for each $\mathbf{x}_n$ such that the output signal $\mathbf{O} = [\mathbf{o}_1 \cdots \mathbf{o}_N] \in \mathbb{C}^{F \times N}$ changes *slowest* over time [1].[1] The output of each

[1]The original SFA is defined only in the real valued space, however, an extension into complex space is trivially archived by replacing the transpose operation with the Hermitian transpose.

individual sample is formed as the concatenation of F mappings $s_f : \mathbb{C}^P \rightarrow \mathbb{C}, f = 1,\dots,F$, such that $\mathbf{o}_n = [s_1(\mathbf{x}_n) \cdots s_F(\mathbf{x}_n)]^\mathsf{T}$, where $(.)^\mathsf{T}$ computes the transpose. SFA minimizes the *slowness* for these values, defined as

$$\Delta(s_f) = \frac{1}{N}\sum_{n=1}^{N} |\dot{s}_f(\mathbf{x}_n)|^2. \tag{9}$$

Hence, SFA minimizes the time derivative $\dot{s}_f$ of s_f for the whole video sequence. Although, $\dot{s}_f$ is often represented as the difference between consecutive time steps, *i.e.* $\dot{s}_f(\mathbf{x}_n) = s_f(\mathbf{x}_n) - s_f(\mathbf{x}_{n-1})$ [3], [15], any derivative may be utilized.

Additionally, [1] introduces constrains to avoid trivial solutions and prevent information redundancy. The output signals of each mapping s_f, *i.e.* $\mathbf{s}_f = [s_f(\mathbf{x}_1) \cdots s_f(\mathbf{x}_N)]^\mathsf{T}$, are required to have zero mean, eq. (10), and unit variance, eq. (11). Moreover, all $\mathbf{s}_f$ are constrained to be uncorrelated, eq. (12). Finally, an ordering of the components according to their slowness is employed, eq. (13). We summarize:

$$\forall f \qquad \mathbf{s}_f^\mathsf{H}\mathbf{1}_{N\times 1} = 0 \tag{10}$$

$$\forall f \qquad \mathbf{s}_f^\mathsf{H}\mathbf{s}_f = 1 \tag{11}$$

$$\forall f \neq e \qquad \mathbf{s}_f^\mathsf{H}\mathbf{s}_e = 0 \tag{12}$$

$$\forall f < e \qquad \Delta(s_f) < \Delta(s_e) \tag{13}$$

where $\mathbf{1}_{a\times b}$ is an $a \times b$ matrix with all elements equal to 1.

Often, the input features $\mathbf{X} \in \mathbb{C}^{P\times N}$ are either assumed to be linear and directly taken from the input values $\mathbf{Z} = [\mathbf{z}_1 \cdots \mathbf{z}_N] \in \mathbb{C}^{P'\times N}$, *i.e.* $\mathbf{x}_n = \mathbf{z}_n$, $P = P'$, or a result of a nonlinear expansion, *e.g.* a quadratic expansion where $\mathbf{z} \in \mathbb{R}^{P'}$

$$\mathbf{x}_n = \begin{bmatrix} \mathbf{z}_n(1) \\ \vdots \\ \mathbf{z}_n(1)\mathbf{z}_n(P') \\ \vdots \\ \mathbf{z}_n(P')\mathbf{z}_n(P') \end{bmatrix},\ P = P' + P'\frac{P'+1}{2} \tag{14}$$

where $\mathbf{a}(b)$ denotes the b^{th} element of vector $\mathbf{a}$. Generally, we may utilize any mapping $\phi : \mathbb{C}^{P'} \rightarrow \mathcal{K}$, such that $\mathbf{x}_n = \phi(\mathbf{z}_n)$.

SFA can be solved by means of the generalized eigenvalue problem [2]. Let the scatter of the data be $\mathbf{S} = \sum_{n=1}^{N}(\mathbf{x}_n - \boldsymbol{\mu})(\mathbf{x}_n - \boldsymbol{\mu})^\mathsf{H}$, where $\boldsymbol{\mu}$ is the sample mean. The scatter of the data's derivative is $\dot{\mathbf{S}} = \sum_{n=1}^{N}(\dot{\mathbf{x}}_n - \dot{\boldsymbol{\mu}})(\dot{\mathbf{x}}_n - \dot{\boldsymbol{\mu}})^\mathsf{H}$, where $\dot{\mathbf{x}}_n$ is the derivative of $\mathbf{x}_n$, and $\dot{\boldsymbol{\mu}}$ is the derivatives' mean. Then, SFA finds a projection $\mathbf{V} = [\mathbf{v}_1 \cdots \mathbf{v}_F] \in \mathbb{C}^{P\times F}$ such that

$$\mathbf{V} = \arg\min_{\check{\mathbf{V}}} \operatorname{tr}\left(\left(\check{\mathbf{V}}^\mathsf{H}\mathbf{S}\check{\mathbf{V}}\right)^{-1}\check{\mathbf{V}}^\mathsf{H}\dot{\mathbf{S}}\check{\mathbf{V}}\right) \tag{15}$$

where $\operatorname{tr}(.)$ computes the trace of a matrix. After ordering, the functions s_f are then provided by $s_f(\mathbf{x}_n) = \mathbf{v}_f^\mathsf{H}(\mathbf{x}_n - \boldsymbol{\mu} - \dot{\boldsymbol{\mu}})$.

C. Solving Kernel SFA in Krein Space

Let $\phi : \mathbb{C}^{P'} \rightarrow \mathcal{K}$ be an unknown mapping into Krein space whose inner product is equivalent to a know kernel, $k : \mathbb{C}^{P'} \times \mathbb{C}^{P'} \rightarrow \mathbb{C}$. Although we notate ϕ to describe our method, we never require its evaluation, as k is employed.

We define $\mathbf{X} = [\phi(\mathbf{z}_1) \cdots \phi(\mathbf{z}_n)]$ as the implicitly given sample matrix of the original features in $\mathbf{Z}$. The samples' mean and centralized matrix can be respectively computed as

$$\boldsymbol{\mu} = \tfrac{1}{N}\mathbf{X}\mathbf{1}_{N\times 1} = \mathbf{X}\mathbf{M} \tag{16}$$

$$\tilde{\mathbf{X}} = \mathbf{X}\left(\mathbf{I}_N - \tfrac{1}{N}\mathbf{1}_{N\times N}\right) = \mathbf{X}\mathbf{C} \tag{17}$$

where $\mathbf{M} = \frac{1}{N}\mathbf{1}_{N\times 1}$ and $\mathbf{C} = \mathbf{I}_N - \frac{1}{N}\mathbf{1}_{N\times N}$ [10], [33]. Furthermore, we define the total scatter matrix

$$\begin{aligned}\mathbf{S} &\triangleq \textstyle\sum_{n=1}^{N}(\phi(\mathbf{z}_n) - \boldsymbol{\mu})(\phi(\mathbf{z}_n) - \boldsymbol{\mu})^* \\ &= \tilde{\mathbf{X}}\tilde{\mathbf{X}}^* = \tilde{\mathbf{X}}\tilde{\mathbf{X}}^\mathsf{H}\mathbf{J}\end{aligned} \tag{18}$$

for some fundamental symmetry matrix $\mathbf{J}$. Note, this matrix describes the scatter in the associated Hilbert space $|\mathcal{K}|$.

Similarly to [3], [15], the signal derivative is expressed as

$$\dot{\mathbf{X}} = \mathbf{X}\begin{bmatrix} 0 & -1 & & \\ 0 & 1 & -1 & \\ \vdots & & \ddots & \ddots \\ 0 & & & 1 \end{bmatrix} = \mathbf{X}\mathbf{D} \tag{19}$$

where $\mathbf{D} \in \mathbb{R}^{N\times N}$ describes the backward difference.[2] The derivatives' mean, $\dot{\boldsymbol{\mu}}$, and centralized matrix, $\dot{\tilde{\mathbf{X}}}$, are respectively found analogously to eq. (16) and eq. (17), and we define the scatter matrix of the derivatives as

$$\dot{\mathbf{S}} \triangleq \dot{\tilde{\mathbf{X}}}\dot{\tilde{\mathbf{X}}}^* = [\mathbf{X}\mathbf{D}\mathbf{C}]\left[\mathbf{C}^\mathsf{T}\mathbf{D}^\mathsf{T}\mathbf{X}^\mathsf{H}\right]\mathbf{J}. \tag{20}$$

Based on KSFA in Hilbert space [15] and KPCA in Krein space [12], we formulate the projection as linear combination $\mathbf{V} = \tilde{\mathbf{X}}\acute{\mathbf{V}}$, and define KSFA's optimization in Krein space as

$$\acute{\mathbf{V}} = \arg\min_{\check{\mathbf{V}}} \operatorname{tr}\left(\left(\check{\mathbf{V}}^\mathsf{H}\tilde{\mathbf{K}}^2\check{\mathbf{V}}\right)^{-1}\check{\mathbf{V}}^\mathsf{H}\mathbf{K}\mathbf{D}\mathbf{C}\mathbf{D}^\mathsf{T}\mathbf{K}^\mathsf{H}\check{\mathbf{V}}\right) \tag{21}$$

where $\mathbf{K} = \mathbf{X}^*\mathbf{X}$ is the kernel matrix, and $\tilde{\mathbf{K}} = \tilde{\mathbf{X}}^*\tilde{\mathbf{X}} = \mathbf{C}^\mathsf{T}\mathbf{K}\mathbf{C}$ its centralized version. The optimum of eq. (21) is found *via* a two step approach. First, we ensure that

$$\acute{\mathbf{V}}^\mathsf{H}\tilde{\mathbf{K}}^2\acute{\mathbf{V}} = \mathbf{I} \tag{22}$$

and then we find the optimum for

$$\acute{\mathbf{V}} = \arg\min_{\check{\mathbf{V}}} \operatorname{tr}\left(\check{\mathbf{V}}^\mathsf{H}\mathbf{K}\mathbf{D}\mathbf{C}\mathbf{D}^\mathsf{T}\mathbf{K}^\mathsf{H}\check{\mathbf{V}}\right). \tag{23}$$

We achieve eq. (22) by whitening the squared kernel matrix, $\tilde{\mathbf{K}}^2$. For this, let us compute the eigenvalue decomposition, $\tilde{\mathbf{K}}^2 = \boldsymbol{\Omega}\boldsymbol{\Lambda}^2\boldsymbol{\Omega}^\mathsf{H}$, and define $\acute{\mathbf{V}} = \mathbf{W}\boldsymbol{\Theta}$ such that $\mathbf{W} = \boldsymbol{\Omega}|\boldsymbol{\Lambda}|^{-1}$ and $\boldsymbol{\Theta}^\mathsf{H}\boldsymbol{\Theta} = \mathbf{I}$ holds (eigenvalues with zero magnitude and the corresponding eigenvectors are removed). We can now reformulate eq. (23) as

$$\begin{aligned}&\boldsymbol{\Theta} = \arg\min_{\check{\boldsymbol{\Theta}}} \operatorname{tr}\left(\check{\boldsymbol{\Theta}}^\mathsf{H}\mathbf{W}^\mathsf{H}\mathbf{K}\mathbf{D}\mathbf{C}\mathbf{D}^\mathsf{T}\mathbf{K}^\mathsf{H}\mathbf{W}\check{\boldsymbol{\Theta}}\right) \\ &\text{subject to } \check{\boldsymbol{\Theta}}^\mathsf{H}\check{\boldsymbol{\Theta}} = \mathbf{I}\end{aligned} \tag{24}$$

which is solved *via* the eigenvalue decomposition of $\mathbf{W}^\mathsf{H}\mathbf{K}\mathbf{D}\mathbf{C}\mathbf{D}^\mathsf{T}\mathbf{K}^\mathsf{H}\mathbf{W} = \boldsymbol{\Theta}\boldsymbol{\Sigma}\boldsymbol{\Theta}^\mathsf{H}$. Finally, the sought projection is provided by $\mathbf{V} = \tilde{\mathbf{X}}\acute{\mathbf{V}} = \tilde{\mathbf{X}}\mathbf{W}\boldsymbol{\Theta}$.[3] The ordering of the slow feature projections is given by the eigenvalues in $\boldsymbol{\Sigma}$. In particular, the columns in $\mathbf{V}$, or equivalently the columns in $\acute{\mathbf{V}}$, that correspond to the smallest non-zero eigenvalues in $\boldsymbol{\Sigma}$,

[2] If a more general derivative is desired, $\dot{\mathbf{X}}$ may be represented as the product of any two matrices $\dot{\mathbf{X}} = [\phi(\dot{\mathbf{z}}_1)\cdots\phi(\dot{\mathbf{z}}_{N'})]$ and $\mathbf{D} \in \mathbb{R}^{N'\times N}$.

[3] If $\dot{\mathbf{X}} \neq \mathbf{X}$, $\mathbf{W}$ seldom introduces a null-space to the projection of $\dot{\mathbf{S}}$. Here, an alternative optimization problem with total scatter matrix of samples and derived samples facilitates slow feature analysis. However, as this is rare and exceeds the scope of this work, we refer to [34], [35].

Algorithm 1 BATCH KSFA IN KREIN SPACE

Input: The training data $\mathbf{Z} \in \mathbb{C}^{P' \times N}$, the derivative matrix $\mathbf{D} \in \mathbb{R}^{N \times N}$, the number of components kept after whitening R, and the kernel function $k : \mathbb{C}^{P'} \times \mathbb{C}^{P'} \to \mathbb{C}$.

Output: The data projection $\acute{\mathbf{V}}$ with sorted components according to slowness.

1: Compute the kernel matrix $\mathbf{K} = \mathbf{X}^*\mathbf{X}$, utilizing k.
2: Find $\mathbf{C}^{\mathsf{H}}\mathbf{K}\mathbf{C} = \mathbf{\Omega}\mathbf{\Lambda}\mathbf{\Omega}^{\mathsf{H}}$.
3: Form the reduced set $\mathbf{\Omega}_R \in \mathbb{C}^{N \times R}$ and $\mathbf{\Lambda}_R \in \mathbb{R}^{R \times R}$ which is related to the R eigenvalues with largest magnitude in $|\mathbf{\Lambda}|$.
4: Set $\mathbf{W}_R = \mathbf{\Omega}_R |\mathbf{\Lambda}_R|^{-1}$.
5: Compute $\mathbf{W}_R^{\mathsf{H}}\mathbf{C}^{\mathsf{T}}\mathbf{K}\mathbf{D}\mathbf{C}\mathbf{D}^{\mathsf{T}}\mathbf{K}\mathbf{C}\mathbf{W}_R = \mathbf{\Theta}\mathbf{\Sigma}\mathbf{\Theta}^{\mathsf{H}}$.
6: Reorganize the components in $\mathbf{\Theta}$ in relation to the ascending eigenvalues in $\mathbf{\Sigma}$.
7: Set $\acute{\mathbf{V}} = \mathbf{W}_R\mathbf{\Theta}$.

Algorithm 2 TESTING SFA IN KREIN SPACE

Input: The to-be-tested sample $\mathbf{z} \in \mathbb{C}^{P'}$, the data projection $\acute{\mathbf{V}}$ with sorted components, the number F of slow features to be used, the training data $\mathbf{Z} \in \mathbb{C}^{P' \times N}$, the derivative matrix $\mathbf{D} \in \mathbb{R}^{N \times N}$, and the kernel function $k : \mathbb{C}^{P'} \times \mathbb{C}^{P'} \to \mathbb{C}$.

Output: The output signal $\mathbf{o} \in \mathbb{C}^F$.

1: Compute $\mathbf{o}' = \tilde{\mathbf{X}}^*\mathbf{x} - \tilde{\mathbf{X}}^*\boldsymbol{\mu} - \tilde{\mathbf{X}}^*\dot{\boldsymbol{\mu}}$, where $\mathbf{x} = \phi(\mathbf{z})$, using the kernel k.
2: Find $\mathbf{o} = \acute{\mathbf{V}}_F^{\mathsf{H}}\mathbf{o}'$ where $\acute{\mathbf{V}}_F \in \mathbb{C}^{N \times F}$ consists of the first F rows of $\acute{\mathbf{V}}$.

represent the smallest changing features of the sample data.

Often, high dimensional data contains dimensions which virtually never vary. Unfortunately, such data points are likely to provide the slowest features in SFA. Therefore, a dimensionality reduction on the scatter matrix of the original data is often advantageous. The eigenvalue decomposition of $\tilde{\mathbf{K}}^2$ is ideal for this task, as we can extract the eigenvectors $\mathbf{\Omega}_R \in \mathbb{C}^{N \times R}$ related to the largest absolute eigenvalues $|\mathbf{\Lambda}_R| \in \mathbb{R}^{R \times R}$. The projection of the reduction is then given by $\mathbf{W}_R = \mathbf{\Omega}_R |\mathbf{\Lambda}_R|^{-1}$. All other parts of the calculation remain unchanged. In the following, the reduced set is implied when we write $\mathbf{\Lambda}$ and $\mathbf{\Omega}$ for ease of notation.

For convenience and clarity, we summarize the learning procedure of batch SFA in algorithm 1, and show how the slow feature functions are applied in algorithm 2.

III. A FRAMEWORK FOR ONLINE KSFA

We now formulate online KSFA which updates the slow features from novel data. In section III-A, online SFA with arbitrary Krein space kernels is introduced. A reduced set expansion is required to employ general kernels. In section III-B, we present a special kernel-type, which allows a representation which does not required a reduced set expansion.

A. General Online Kernel SFA

As seen in section II, SFA can be solved in two stages, the data whitening in eq. (22) and the decomposition in eq. (23). In figure 1, we illustrate our incremental setup. We employ the same notation as before, and indicate time steps and new data by subscripts (*e.g.* $\mathbf{X}_t$ is the implicitly mapped sample matrix at time t, and N_δ is the number of new samples).

1) Indefinite KPCA Update for Online Whitening: The update of the whitening projections can be performed with online KPCA. We extend the incremental KPCA in [10] to allow for indefinite kernels in Krein spaces.

Let $\mathbf{X}_\delta$ be the mapping of the new samples in $\mathbf{Z}_\delta$, such that $\mathbf{X}_t = \left[\begin{array}{cc} \mathbf{X}_{t-1} & \mathbf{X}_\delta \end{array}\right]$ provides the updated data set. We want to incrementally update the eigenvectors $\mathbf{\Omega}_{t-1}$ and the eigenvalue magnitudes $|\mathbf{\Lambda}_{t-1}|$ of the previous time step $t-1$, to find the new whitening projection $\mathbf{W}_t = \mathbf{\Omega}_t |\mathbf{\Lambda}_t|^{-1}$ which incorporates the additional information of the new samples.

Similarly to eq. (16) and eq. (17), we notate the new data's mean as $\boldsymbol{\mu}_\delta = \mathbf{X}_\delta \mathbf{M}_\delta$ and its centered sample matrix as $\tilde{\mathbf{X}}_\delta = \mathbf{X}_\delta \mathbf{C}_\delta$, where $\mathbf{M}_\delta = \frac{1}{N_\delta}\mathbf{1}_{N_\delta \times 1}$ and $\mathbf{C}_\delta = \mathbf{I}_{N_\delta} - \frac{1}{N_\delta}\mathbf{1}_{N_\delta \times N_\delta}$. For the sake of simplification, let us first assume the data mean unchanged (*i.e.* $\boldsymbol{\mu}_{t-1} = \boldsymbol{\mu}_\delta$). We define the kernel matrix of the new data $\tilde{\mathbf{K}}_\delta = \tilde{\mathbf{X}}_\delta^* \tilde{\mathbf{X}}_\delta$ and find the eigenvalue decomposition $\tilde{\mathbf{K}}_\delta^2 = \mathbf{\Omega}_\delta \mathbf{\Lambda}_\delta^2 \mathbf{\Omega}_\delta^{\mathsf{H}}$. Then, in respect to eq. (22), we seek to find the eigenvalue decomposition of

$$\begin{aligned} \tilde{\mathbf{K}}_t^2 &= \check{\mathbf{\Omega}}_t \begin{bmatrix} |\mathbf{\Lambda}_{t-1}| & \mathbf{\Omega}_{t-1}^{\mathsf{H}} \tilde{\mathbf{X}}_{t-1}^* \tilde{\mathbf{X}}_\delta \mathbf{\Omega}_\delta \\ \mathbf{\Omega}_\delta^{\mathsf{H}} \tilde{\mathbf{X}}_\delta^* \tilde{\mathbf{X}}_{t-1} \mathbf{\Omega}_{t-1} & |\mathbf{\Lambda}_\delta| \end{bmatrix}^2 \check{\mathbf{\Omega}}_t^{\mathsf{H}} \\ &= \check{\mathbf{\Omega}}_t \left[\hat{\mathbf{\Omega}}_t \mathbf{\Lambda}_t^2 \hat{\mathbf{\Omega}}_t^{\mathsf{H}}\right] \check{\mathbf{\Omega}}_t^{\mathsf{H}} = \mathbf{\Omega}_t \mathbf{\Lambda}_t^2 \mathbf{\Omega}_t^{\mathsf{H}} \end{aligned} \quad (25)$$

where $\check{\mathbf{\Omega}}_t = \begin{bmatrix} \mathbf{\Omega}_{t-1} & \mathbf{0} \\ \mathbf{0} & \mathbf{\Omega}_\delta \end{bmatrix}$ is an orthogonal matrix, composed of the eigenvectors in $\mathbf{\Omega}_{t-1}$ and $\mathbf{\Omega}_\delta$. Notice, eq. (25) can be solved utilizing the decomposition of the inner matrix, $\hat{\mathbf{\Omega}}_t \mathbf{\Lambda}_t^2 \hat{\mathbf{\Omega}}_t^{\mathsf{H}}$. The size of this matrix is independent of the sample number. The projected eigenvectors provide the solution

$$\mathbf{\Omega}_t = \begin{bmatrix} \mathbf{\Omega}_{t-1} & \mathbf{0} \\ \mathbf{0} & \mathbf{\Omega}_\delta \end{bmatrix} \hat{\mathbf{\Omega}}_t = \check{\mathbf{\Omega}}_t \hat{\mathbf{\Omega}}_t. \quad (26)$$

Thus, the new projection is given by $\mathbf{W}_t = \mathbf{\Omega}_t |\mathbf{\Lambda}_t|^{-1}$ (zero eigenvalues and their eigenvectors are removed implicitly). A dimensionality reduction may be applied as outlined in section II-C. It is important to note that $\mathbf{\Omega}_t$ may grow arbitrarily large over time. We refer to section III-A4, where we overcome this problem by introducing a reduced set representation that bounds the grows by means of approximation.

Thus far, only the case with unchanged data mean is considered. A simple modification of $\tilde{\mathbf{X}}_\delta$ allows for changing means [9], [10]. In particular, we include an additional data point to the new samples, whose sole purpose is the correction of the mean substraction

$$\begin{aligned} \tilde{\mathbf{X}}'_\delta &= \left[\begin{array}{cc} \tilde{\mathbf{X}}_\delta & \sqrt{\frac{N_{t-1}N_\delta}{N_{t-1}+N_\delta}}\left(\boldsymbol{\mu}_{t-1} - \boldsymbol{\mu}_\delta\right) \end{array}\right] \\ &= \mathbf{X}_t \left[\begin{array}{cc} \begin{bmatrix} \mathbf{0} \\ \mathbf{C}_\delta \end{bmatrix} & \sqrt{\frac{N_{t-1}N_\delta}{N_{t-1}+N_\delta}} \begin{bmatrix} \mathbf{M}_{t-1} \\ -\mathbf{M}_\delta \end{bmatrix} \end{array}\right]. \end{aligned} \quad (27)$$

Now, we compute eq. (25) with $\tilde{\mathbf{X}}'_\delta$ in place of $\tilde{\mathbf{X}}_\delta$. Finally, we update the mean of the overall data

$$\begin{aligned} \boldsymbol{\mu}_t &= \frac{N_{t-1}}{N_{t-1}+N_\delta}\mathbf{X}_{t-1}\mathbf{M}_{t-1} + \frac{N_\delta}{N_{t-1}+N_\delta}\mathbf{X}_\delta \mathbf{M}_\delta \\ &= \mathbf{X}_t \mathbf{M}_t \end{aligned} \quad (28)$$

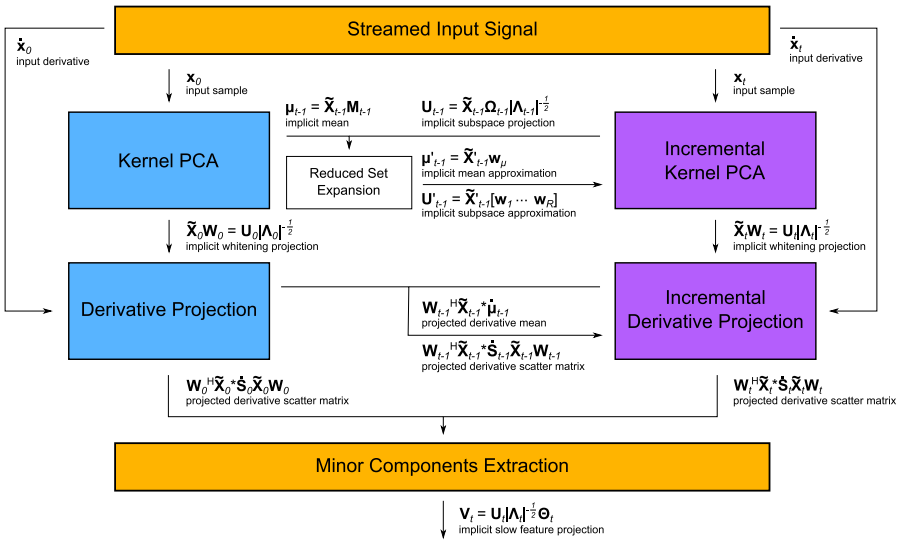

Fig. 1. Illustration of incremental SFA at initialization (time 0) and time-step t. We utilize incremental kernel PCA and our incremental derivative projection to find the slow features at each time-step. Details are provided in the text.

where $\mathbf{M}_t = \frac{1}{N_{t-1}+N_\delta} \begin{bmatrix} N_{t-1}\mathbf{M}_{t-1} \\ N_\delta \mathbf{M}_\delta \end{bmatrix}$.

It is important to note, that our complete incremental update never requires the explicit calculation of the unknown mapping ϕ, as we employ the kernel trick using the kernel k. Furthermore, ignoring the growth of the kernel support set, our framework for updating general KPCAs in Krein space computes in constant time and memory. Section III-A4 introduces a reduced set representation for constant execution of the whole algorithm. First, however, we complete the online KSFA framework with the second part (related to eq. (23)).

2) Slow Feature Update: After finding the whitening matrix, the slow features in $\dot{\mathbf{V}}_{t-1} = \mathbf{W}_{t-1}\mathbf{\Theta}_{t-1}$ require update. More precisely, the eigendecomposition of the projected scatter matrix, $\mathbf{W}_{t-1}^{\mathsf{H}}\tilde{\mathbf{X}}_{t-1}^{*}\dot{\mathbf{S}}_{t-1}\tilde{\mathbf{X}}_{t-1}\mathbf{W}_{t-1} = \mathbf{\Theta}_{t-1}\mathbf{\Sigma}_{t-1}\mathbf{\Theta}_{t-1}^{\mathsf{H}}$, needs to be renewed with regards to the new derivatives and the projection, which we denote $\dot{\mathbf{X}}_\delta$ and $\mathbf{W}_t$ respectively.

Let us first assume unchanged means (*i.e.* $\dot{\boldsymbol{\mu}}_{t-1} = \dot{\boldsymbol{\mu}}_\delta$). The new projection of the new scatter matrix is then provided as

$$\begin{aligned}
&\mathbf{W}_t^{\mathsf{H}}\tilde{\mathbf{X}}_t^{*}\dot{\mathbf{S}}_t\tilde{\mathbf{X}}_t\mathbf{W}_t \\
&= \mathbf{W}_t^{\mathsf{H}}\tilde{\mathbf{X}}_t^{*}\dot{\mathbf{S}}_{t-1}\tilde{\mathbf{X}}_t\mathbf{W}_t + \mathbf{W}_t^{\mathsf{H}}\tilde{\mathbf{X}}_t^{*}\dot{\mathbf{S}}_\delta\tilde{\mathbf{X}}_t\mathbf{W}_t \\
&= \mathbf{W}_t^{\mathsf{H}}\begin{bmatrix} \tilde{\mathbf{X}}_{t-1}^{*}\dot{\mathbf{X}}_{t-1}\mathbf{C}_{t-1} \\ \tilde{\mathbf{X}}_\delta^{*}\dot{\mathbf{X}}_{t-1}\mathbf{C}_{t-1} \end{bmatrix}\begin{bmatrix} \tilde{\mathbf{X}}_{t-1}^{*}\dot{\mathbf{X}}_{t-1}\mathbf{C}_{t-1} \\ \tilde{\mathbf{X}}_\delta^{*}\dot{\mathbf{X}}_{t-1}\mathbf{C}_{t-1} \end{bmatrix}^{\mathsf{H}}\mathbf{W}_t \\
&+ \mathbf{W}_t^{\mathsf{H}}\tilde{\mathbf{X}}_t^{*}\dot{\mathbf{X}}_\delta\mathbf{C}_\delta\dot{\mathbf{X}}_\delta^{*}\tilde{\mathbf{X}}_t\mathbf{W}_t \qquad (29)
\end{aligned}$$

where $\dot{\mathbf{S}}_\delta$ is the scatter of the new derivatives.

The second term of eq. (29) can be directly computed. More complicated, however, is the calculation of the first term when constant running time is required for online learning. Notice that $\tilde{\mathbf{X}}_t\mathbf{W}_t$ can be expressed through the samples' subspaces for $\tilde{\mathbf{X}}_{t-1}$ and $\tilde{\mathbf{X}}_\delta$, given by $\mathbf{U}_{t-1} = \tilde{\mathbf{X}}_{t-1}\mathbf{\Omega}_{t-1}|\mathbf{\Lambda}_{t-1}|^{-\frac{1}{2}}$ and $\mathbf{U}_\delta = \tilde{\mathbf{X}}_\delta\mathbf{\Omega}_\delta|\mathbf{\Lambda}_\delta|^{-\frac{1}{2}}$ respectively. We write

$$\tilde{\mathbf{X}}_t\mathbf{W}_t = \begin{bmatrix} \mathbf{U}_{t-1}|\mathbf{\Lambda}_{t-1}|^{\frac{1}{2}} & \mathbf{U}_\delta|\mathbf{\Lambda}_\delta|^{\frac{1}{2}} \end{bmatrix}\hat{\mathbf{\Omega}}_t|\mathbf{\Lambda}_t|^{-1}. \qquad (30)$$

It is reasonable to assume that $\dot{\mathbf{X}}_{t-1}$ is well represented by the subspace $\mathbf{U}_{t-1}$ alone, as the projection's components were selected to reproduce both $\tilde{\mathbf{X}}_{t-1}$ and $\dot{\mathbf{X}}_{t-1}$. The new components which are introduced by the new samples in $\tilde{\mathbf{X}}_\delta$ are unlikely to be of significance to $\dot{\mathbf{X}}_{t-1}$. Hence we omit their contribution,[4] and rewrite the first term of eq. (29) as

$$\mathbf{W}_t^{\mathsf{H}}\tilde{\mathbf{X}}_t^{*}\dot{\mathbf{S}}_{t-1}\tilde{\mathbf{X}}\mathbf{W}_t \approx \mathbf{P}_t^{\mathsf{H}}\mathbf{W}_{t-1}^{\mathsf{H}}\tilde{\mathbf{X}}_{t-1}^{*}\dot{\mathbf{S}}\tilde{\mathbf{X}}_{t-1}\mathbf{W}_{t-1}\mathbf{P}_t \qquad (31)$$

where $\mathbf{P}_t \in \mathbb{C}^{R_{t-1}\times R_t}$ is a correction matrix given by

$$\mathbf{P}_t = \begin{bmatrix} |\mathbf{\Lambda}_{t-1}| & \mathbf{0} \end{bmatrix}\hat{\mathbf{\Omega}}_t|\mathbf{\Lambda}_t|^{-1} \qquad (32)$$

where $\hat{\mathbf{\Omega}}_t$ corresponds to the non-zero eigenvalues in $\mathbf{\Lambda}_t$.

Let us represent the old data mean in its projected form, $\mathbf{W}_{t-1}^{\mathsf{H}}\tilde{\mathbf{X}}_{t-1}^{*}\dot{\boldsymbol{\mu}}_{t-1}$. Using $\mathbf{P}_t$, its update is approximated by

$$\mathbf{W}_t^{\mathsf{H}}\tilde{\mathbf{X}}_t^{*}\dot{\boldsymbol{\mu}}_{t-1} \approx \mathbf{P}_t^{\mathsf{H}}\mathbf{W}_{t-1}^{\mathsf{H}}\tilde{\mathbf{X}}_{t-1}^{*}\dot{\boldsymbol{\mu}}_{t-1}. \qquad (33)$$

Similarly to eq. (27), the step to correct the mean substraction is performed by adding a correction sample to the new data, or, equivalently, by adding the following to eq. (29)

$$\frac{N_{t-1}N_\delta}{N_{t-1}+N_\delta}\mathbf{W}_t^{\mathsf{H}}\tilde{\mathbf{X}}_t^{*}(\dot{\boldsymbol{\mu}}_{t-1}-\dot{\boldsymbol{\mu}}_\delta)(\dot{\boldsymbol{\mu}}_{t-1}-\dot{\boldsymbol{\mu}}_\delta)^{*}\tilde{\mathbf{X}}_t\mathbf{W}_t \qquad (34)$$

and the new mean is provided by

$$\mathbf{W}_t^{\mathsf{H}}\tilde{\mathbf{X}}_t^{*}\dot{\boldsymbol{\mu}}_t = \frac{N_{t-1}\mathbf{W}_t^{\mathsf{H}}\tilde{\mathbf{X}}_t^{*}\dot{\boldsymbol{\mu}}_{t-1} + N_\delta\mathbf{W}_t^{\mathsf{H}}\tilde{\mathbf{X}}_t^{*}\dot{\boldsymbol{\mu}}_\delta}{N_{t-1}+N_\delta}. \qquad (35)$$

Finally, after computing the new scatter matrix $\mathbf{W}_t^{\mathsf{H}}\tilde{\mathbf{X}}_t^{*}\dot{\mathbf{S}}_t\tilde{\mathbf{X}}_t\mathbf{W}_t$, we obtain the slow features through eigenvalue decomposition as described in section II-C. Note, the size of this matrix is bound by the number of components in $\mathbf{W}_t$. Hence, the computation time and memory requirements remain constant.

3) Forgetting Factor: In many online systems it is beneficial to attach a higher weight to recent data. A common approach to moderate the balance between old and new data is a *forgetting factor* [8], [9]. This value acts as a weight on the known data, and reduces their impact by a factor $w \in [0,1]$ (usually $0 \ll w < 1$). In the following, we describe how w is applied to both steps of the incremental update.

[4]The forgetting factor, introduced in section III-A3, reduces the effect of errors even further.

Essentially, we want to reduce the effect of the values in $\mathbf{X}_{t-1}$ by w. The forgetting factor can be directly applied to the eigenvalues of the old data, as we find

$$\left(w\tilde{\mathbf{X}}_{t-1}^{*}\tilde{\mathbf{X}}_{t-1}w\right)^{2} = \mathbf{\Omega}_{t-1}\left(w^{2}\mathbf{\Lambda}_{t-1}\right)^{2}\mathbf{\Omega}_{t-1}^{\mathsf{H}}. \tag{36}$$

Hence, we reflect this in the computation of the inner matrix of eq. (25), as we compute the decomposition of

$$\begin{bmatrix} w^{2}|\mathbf{\Lambda}_{t-1}| & \mathbf{\Omega}_{t-1}^{\mathsf{H}}\tilde{\mathbf{X}}_{t-1}^{*}\tilde{\mathbf{X}}_{\delta}\mathbf{\Omega}_{\delta} \\ \mathbf{\Omega}_{\delta}^{\mathsf{H}}\tilde{\mathbf{X}}_{\delta}^{*}\tilde{\mathbf{X}}_{t-1}\mathbf{\Omega}_{t-1} & |\mathbf{\Lambda}_{\delta}| \end{bmatrix}^{2} = \hat{\mathbf{\Omega}}_{t}\mathbf{\Lambda}_{t}^{2}\hat{\mathbf{\Omega}}_{t}^{\mathsf{H}}. \tag{37}$$

The forgetting factor also effects the old data's mean $\boldsymbol{\mu}_{t-1}$, which is changed by multiplying w. Analogously to [9], we update the mean using

$$\mathbf{M}_{t} = \frac{1}{wN_{t-1}+N_{\delta}}\begin{bmatrix} wN_{t-1}\mathbf{M}_{t-1} \\ N_{\delta}\mathbf{M}_{\delta} \end{bmatrix}. \tag{38}$$

Finally, the correction term for the mean substraction in $\tilde{\mathbf{X}}'_{\delta}$ (in eq. (27)) is also modified to reflect w:

$$\mathbf{X}_{t}\left[\begin{bmatrix} \mathbf{0} \\ \mathbf{C}_{\delta} \end{bmatrix} \frac{\sqrt{w^{2}N_{t-1}N_{\delta}(N_{t-1}+N_{\delta})}}{fN_{t-1}+N_{\delta}}\begin{bmatrix} \mathbf{M}_{t-1} \\ -\mathbf{M}_{\delta} \end{bmatrix}\right]. \tag{39}$$

Similar modifications are employed for the second stage of the update. With analogous derivations to above, we find

$$\begin{aligned} &\mathbf{W}_{t}^{\mathsf{H}}\tilde{\mathbf{X}}_{t}^{*}\dot{\mathbf{S}}_{t}\tilde{\mathbf{X}}_{t}\mathbf{W}_{t} \\ &= \frac{\sqrt{w^{2}N_{t-1}N_{\delta}(N_{t-1}+N_{\delta})}}{wN_{t-1}+N_{\delta}}\mathbf{W}_{t}^{\mathsf{H}}\tilde{\mathbf{X}}_{t}^{*}(\dot{\boldsymbol{\mu}}_{t-1}-\dot{\boldsymbol{\mu}}_{\delta}) \\ &\quad (\dot{\boldsymbol{\mu}}_{t-1}-\dot{\boldsymbol{\mu}}_{\delta})^{*}\tilde{\mathbf{X}}_{t}\mathbf{W}_{t}\frac{\sqrt{w^{2}N_{t-1}N_{\delta}(N_{t-1}+N_{\delta})}}{wN_{t-1}+N_{\delta}} \\ &+ \; w^{2}\mathbf{W}_{t}^{\mathsf{H}}\tilde{\mathbf{X}}_{t}^{*}\dot{\mathbf{S}}_{t-1}\tilde{\mathbf{X}}_{t}\mathbf{W}_{t} + \mathbf{W}_{t}^{\mathsf{H}}\tilde{\mathbf{X}}_{t}^{*}\dot{\mathbf{S}}_{\delta}\tilde{\mathbf{X}}_{t}\mathbf{W}_{t} \end{aligned} \tag{40}$$

which can be computed as described in section III-A2. The mean update is now given by

$$\mathbf{W}_{t}^{\mathsf{H}}\tilde{\mathbf{X}}_{t}^{*}\dot{\boldsymbol{\mu}}_{t} = \frac{wN_{t-1}\mathbf{W}_{t}^{\mathsf{H}}\tilde{\mathbf{X}}_{t}^{*}\dot{\boldsymbol{\mu}}_{t-1}+N_{\delta}\mathbf{W}_{t}^{\mathsf{H}}\tilde{\mathbf{X}}_{t}^{*}\dot{\boldsymbol{\mu}}_{\delta}}{wN_{t-1}+N_{\delta}}. \tag{41}$$

Finally, the number of elements at time t is $N_{t} = wN_{t-1}+N_{\delta}$.

4) Introducing Constant Running Time: Like most learning systems which employ the kernel trick, our online SFA with arbitrary Krein kernel depends on a support set of the previously encountered data. One of the major challenges is that this set may grow to become arbitrarily large over time [10], [16], [17]. For example, whenever we wish to apply the slow features $\mathbf{V}$ to a sample $\mathbf{x}$ the kernel matrix $\tilde{\mathbf{X}}_{t}^{*}\mathbf{x}$ is computed, which requires all data points in $\mathbf{Z}_{t}$. This, however, violates the online learning requirements of bound running speed, as $\mathbf{Z}_{t}$ grows at every update.

Although alternatives exist [20], [21], [23], [24], we adopt the reduced set expansion of [10] to ensure constant running time. While we describe the main steps in the following, we refer to [10] for details.

Our algorithm uses the support set $\mathbf{Z}_{t}$ in combination with the eigenvectors $\mathbf{\Omega}_{t}$ in the form of $\tilde{\mathbf{X}}_{t}\mathbf{\Omega}_{t}$. Therefore, in the following, as we exploit properties of orthogonality, we estimate the reduced set expansion based on the subspace $\mathbf{U}_{t} = \tilde{\mathbf{X}}_{t}\mathbf{\Omega}_{t}|\mathbf{\Lambda}_{t}|^{-\frac{1}{2}}$ of $\tilde{\mathbf{X}}_{t}$. Each principal component in $\mathbf{U}_{t} = [\mathbf{u}_{1}\cdots\mathbf{u}_{R}]$ depends on the complete set of previously encountered data in $\mathbf{Z}_{t}$. In particular, each component is realized by computing $\mathbf{u}_{r} = \tilde{\mathbf{X}}_{t}\boldsymbol{\omega}_{r}|\lambda_{r}|^{-\frac{1}{2}}$, where $\boldsymbol{\omega}_{r}$ is the component of $\mathbf{\Omega}_{t}$ related to the r^{th} eigenvalue, denoted λ_{r}, at time t. The basic idea behind the reduced set expansion is depicted in figure 2 [10]. For each component $\mathbf{u}_{r}$, we want to find a set of at most Q pre-images $\hat{\mathbf{Z}}_{r} = [\hat{\mathbf{z}}_{r1}\cdots\hat{\mathbf{z}}_{rQ}]$ whose implicit mappings, denoted $\hat{\mathbf{X}}_{r} = [\phi(\hat{\mathbf{z}}_{r1})\cdots\phi(\hat{\mathbf{z}}_{rQ})]$, best approximate $\mathbf{u}_{r} \approx \hat{\mathbf{X}}_{r}\mathbf{w}_{r}$. Here $\mathbf{w}_{r}$ is an optimized multiplier which controls the weighting of the samples.

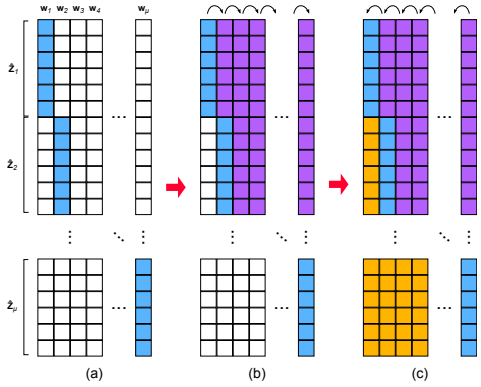

Fig. 2. The mean and each component in $\mathbf{U}_{t}$ is approximated by Q pre-images, denoted $\hat{\mathbf{Z}}_{r}$, and a weight vector $\mathbf{w}_{r}$ as detail in the text and shown in (a). Improved optimization can be archived through propagating the pre-images to other components, as shown in (b) and (c), and detailed in [10].

A two stage greedy search finds the components [10]. Intuitively, we find the next pre-image which carries most information to describe the remainder of $\mathbf{u}_{r}$. For convenience, let us denote the mapped set of the first q samples as $\hat{\mathbf{X}}_{rq} = [\phi(\hat{\mathbf{z}}_{r1})\cdots\phi(\hat{\mathbf{z}}_{rq})]$ and the weighting of the q elements as $\mathbf{w}_{rq} \in \mathbb{R}^{q}$. The $(q+1)^{\text{th}}$ element is found as

$$\hat{\mathbf{z}}_{r(q+1)} = \arg\max_{\check{\mathbf{z}}} \frac{\left(\left(\mathbf{u}_{r}^{*}-\mathbf{w}_{rq}^{\mathsf{H}}\hat{\mathbf{X}}_{rq}^{*}\right)\phi(\check{\mathbf{z}})\right)^{2}}{\phi(\check{\mathbf{z}})^{*}\phi(\check{\mathbf{z}})} \tag{42}$$

and the optimal weighting as [10], [36]

$$\mathbf{w}_{r(q+1)} = \left(\hat{\mathbf{X}}_{r(q+1)}^{*}\hat{\mathbf{X}}_{r(q+1)}\right)^{-1}\hat{\mathbf{X}}_{r(q+1)}^{*}\mathbf{u}_{r}. \tag{43}$$

Analogously, an additional set of at most Q pre-images, denoted $\hat{\mathbf{Z}}_{\mu}$ with mappings $\hat{\mathbf{X}}_{\mu}$ and weights $\mathbf{w}_{\mu}$ approximates the mean $\boldsymbol{\mu}_{t} \approx \hat{\mathbf{X}}_{\mu}\mathbf{w}_{\mu}$. One approach to solve the optimization is gradient decent, but other algorithms exist [37].

Once all pre-images are computed, we could set

$$\mathbf{Z}'_{t} \triangleq \left[\hat{\mathbf{Z}}_{1}\cdots\hat{\mathbf{Z}}_{R} \quad \hat{\mathbf{Z}}_{\mu}\right] \tag{44}$$

$$\mathbf{X}'_{t} \triangleq \left[\hat{\mathbf{X}}_{1}\cdots\hat{\mathbf{X}}_{R} \quad \hat{\mathbf{X}}_{\mu}\right] \tag{45}$$

$$\tilde{\mathbf{U}}'_{t} \triangleq \mathbf{X}'_{t}\begin{bmatrix} \mathbf{w}_{1} & \cdots & \mathbf{0}_{Q\times 1} \\ \vdots & \ddots & \vdots \\ \mathbf{0}_{Q\times 1} & \cdots & \mathbf{w}_{R} \\ \mathbf{0}_{Q\times 1} & \cdots & \mathbf{0}_{Q\times 1} \end{bmatrix} \tag{46}$$

$$\mathbf{M}'_{t} \triangleq \begin{bmatrix} \mathbf{0}_{(RQ)\times 1} \\ \mathbf{w}_{\mu} \end{bmatrix}. \tag{47}$$

However, we apply the method shown in figure 2 and detailed in [10], which extracts a better approximation, by using all elements in $\mathbf{Z}'_{t}$ for each component in $\tilde{\mathbf{U}}'_{t}$ and $\mathbf{M}'_{t}$, effectively making these less sparse. See [10] for details.

Finally, we enforce orthogonality in the new found $\tilde{\mathbf{U}}'_{t}$ subspace. Note, $\tilde{\mathbf{U}}'_{t}$ is only an approximation of the orig-

inal $\mathbf{U}_t$ subspace, and thus the unit norm is no longer guaranteed. We extend [10] to work with complex kernels in Krein space. In particular, using the decomposition of $(\tilde{\mathbf{U}'}_t{}^*\tilde{\mathbf{U}'}_t)^2 = \mathbf{\Pi}\mathbf{\Upsilon}^2\mathbf{\Pi}^\mathsf{H}$, we whiten the estimated projection to find orthogonal components, *i.e.*

$$\mathbf{U'}_t \triangleq \tilde{\mathbf{U}'}_t\mathbf{\Pi}|\mathbf{\Upsilon}|^{-\frac{1}{2}}\mathbf{\Pi}^\mathsf{H}. \tag{48}$$

The new eigenvalues' magnitudes are given by

$$|\mathbf{\Lambda'}_t| = \mathrm{diag}(\mathbf{U'}_t^*\mathbf{U}_t|\mathbf{\Lambda}_t|) \tag{49}$$

where diag(.) discards the values, not located on the diagonal.

After substituting the new representation with the original set, we continue the update through the usual process. Note however, the variable for the number of previous data points N_t remains unchanged. In total, the reduced set contains $Q(R+1)$ preimages, *i.e.* Q for each component and an additional set of Q data values to represent the mean $\boldsymbol{\mu}_t$. Therefore, we compute a reduced set whenever the number of samples in $\mathbf{X}_t$ exceeds $Q(R+1)$. For clarity, the code is provided in the supplementary material.

B. Direct Online SFA with Special Kernels

We now present a special kernel family allows for a direct computation of online KSFA, coined direct online KSFA. Direct online KSFA does not require a reduced set expansion, making the update computationally faster and more accurate.

1) Motivation: Eq. (6) defines a regular Krein space's kernel representation as the dot-product between the implicitly defined mappings $\phi(\mathbf{x}), \phi(\mathbf{y}) \in \mathcal{K}$, where $\mathbf{x}, \mathbf{y} \in \mathbb{C}^{P'}$ are samples in the original feature space. In traditional systems, the implicity of the mapping, however, makes the reduced set expansion necessary. In our previous work [12], we introduce a special kernel which does not require such set. In particular, the kernel can be expressed exactly by two explicit mappings:

$$a(\mathbf{x}) = \begin{bmatrix} \frac{\mathbf{R}_\mathbf{x} \odot e^{i\boldsymbol{\theta}_\mathbf{x}}}{2\sqrt{\sum_{p=1}^{P'} \mathbf{R}_\mathbf{x}^2(p)P'}} \\ e^{i\boldsymbol{\theta}_\mathbf{x}} \end{bmatrix},\ b(\mathbf{x}) = \begin{bmatrix} e^{i\boldsymbol{\theta}_\mathbf{x}} \\ \frac{\mathbf{R}_\mathbf{x} \odot e^{i\boldsymbol{\theta}_\mathbf{x}}}{2\sqrt{\sum_{p=1}^{P'} \mathbf{R}_\mathbf{x}^2(p)P'}} \end{bmatrix} \tag{50}$$

where $\mathbf{x}$ corresponds to an image with vectorized gradient magnitudes $\mathbf{R}_\mathbf{x} \in \mathbb{R}^{+P'}$ and gradient angles $\boldsymbol{\theta}_\mathbf{x} \in [-\pi, \pi)^{P'}$. Here, the operator $\odot$ is shorthand for a componentwise multiplication, and $e^{i\boldsymbol{\theta}_\mathbf{x}} \triangleq \begin{bmatrix} e^{i\boldsymbol{\theta}_\mathbf{x}(1)} & \cdots & e^{i\boldsymbol{\theta}_\mathbf{x}(P')} \end{bmatrix}^\mathsf{T}$. As this kernel has been shown to achieve state-of-the-art performance when utilized for tracking and recognition [12], We now desire to employ this kernel for online KSFA.

For generality, we consider a special kernel family. In particular, we assume that our kernels are expressed by two explicit functions $a : \mathbb{C}^{P'} \to \mathbb{C}^P$ and $b : \mathbb{C}^{P'} \to \mathbb{C}^P$ such that

$$k(\mathbf{x}, \mathbf{y}) = a(\mathbf{x})^\mathsf{H} b(\mathbf{y}) = a(\mathbf{y})^\mathsf{H} b(\mathbf{x}). \tag{51}$$

It is important to note here that not our mapping is not only finite dimensional like [24], but it is also exactly equivalent to the computation of the kernel function.

2) Direct Online Learning: Let us develop a direct online KSFA algorithm for any kernel that satisfies eq. (51). The main benefits of the two mappings is the capability of computing an equivalence of the implicit projection explicitly. Given the sample matrices of mappings a and b, denoted $\mathbf{X}^{(a)} = \begin{bmatrix} a(\mathbf{z}_1) & \cdots & a(\mathbf{z}_N) \end{bmatrix}$ and $\mathbf{X}^{(b)}$, we replace the implicit linear combination $\tilde{\mathbf{X}}\mathbf{\Omega}$ with two explicit matrices $\mathbf{A} = \tilde{\mathbf{X}}^{(a)}\mathbf{\Omega} = \mathbf{X}^{(a)}\mathbf{C}\mathbf{\Omega}$ and $\mathbf{B} = \tilde{\mathbf{X}}^{(b)}\mathbf{\Omega}$. Then, the projection of a new sample $\mathbf{z}$ is exactly computed by utilizing either of its mapping, such that $\mathbf{\Omega}\tilde{\mathbf{X}}^*\phi(\mathbf{z}) = \mathbf{A}^\mathsf{H} b(\mathbf{z}) = \mathbf{B}^\mathsf{H} a(\mathbf{z})$. We reformulate the online learning process using this setup.

We exploit the explicit mappings to find the decomposition of the inner matrix in eq. (25) *via*

$$\begin{bmatrix} |\mathbf{\Lambda}_{t-1}| & \mathbf{A}_{t-1}^\mathsf{H}\mathbf{B}_\delta \\ \mathbf{B}_\delta^\mathsf{H}\mathbf{A}_{t-1} & |\mathbf{\Lambda}_\delta| \end{bmatrix}^2 = \hat{\mathbf{\Omega}}_t\mathbf{\Lambda}_t^2\hat{\mathbf{\Omega}}_t \tag{52}$$

where $\mathbf{A}_\delta$ and $\mathbf{B}_\delta$ is related to the new data in $\tilde{\mathbf{X}}_\delta^{(a)}$ and $\tilde{\mathbf{X}}_\delta^{(b)}$ respectively. The update of $\mathbf{A}_t$ and $\mathbf{B}_t$ is analogous to eq. (26):

$$\mathbf{A}_t = \begin{bmatrix} \mathbf{A}_{t-1} & \mathbf{A}_\delta \end{bmatrix}\hat{\mathbf{\Omega}}_t \tag{53}$$

$$\mathbf{B}_t = \begin{bmatrix} \mathbf{B}_{t-1} & \mathbf{B}_\delta \end{bmatrix}\hat{\mathbf{\Omega}}_t. \tag{54}$$

Additionally, we substitute the data mean $\boldsymbol{\mu}$ with the explicit versions, $\boldsymbol{\mu}^{(a)} = \mathbf{X}^{(a)}\mathbf{M}$ and $\boldsymbol{\mu}^{(b)}$, making the correction in eq. (27) and the mean update in eq. (28) trivial.

All that is left to do, is the formulation of the slow feature extraction in section III-A2 using the explicit mappings. We rewrite the projection in eq. (29) as follows

$$\begin{aligned} &\mathbf{W}_t^\mathsf{H}\tilde{\mathbf{X}}_t^*\dot{\mathbf{S}}_t\tilde{\mathbf{X}}_t\mathbf{W}_t \\ &\approx\ \mathbf{P}_t|\mathbf{\Lambda}_{t-1}|^{-1}\mathbf{A}_{t-1}^\mathsf{H}\dot{\mathbf{X}}_{t-1}^{(b)}\mathbf{C}_{t-1}\dot{\mathbf{X}}_{t-1}^{(a)\,\mathsf{H}}\mathbf{B}_{t-1}|\mathbf{\Lambda}_{t-1}|^{-1}\mathbf{P}_t \\ &+\ |\mathbf{\Lambda}_t|^{-1}\mathbf{A}_t^\mathsf{H}\dot{\mathbf{X}}_\delta^{(b)}\mathbf{C}_\delta\dot{\mathbf{X}}_\delta^{(a)\mathsf{H}}\mathbf{B}_t|\mathbf{\Lambda}_t|^{-1} \end{aligned} \tag{55}$$

where $\dot{\mathbf{X}}^{(a)}$ and $\dot{\mathbf{X}}^{(b)}$ are the explicit versions of the derived samples in $\dot{\mathbf{X}}$. Notice that $\tilde{\mathbf{X}}\mathbf{W}$ is expressed through $\mathbf{A}|\mathbf{\Lambda}|^{-1}$ and $\mathbf{B}|\mathbf{\Lambda}|^{-1}$. Analogously, incorporating explicit mean representations $\dot{\boldsymbol{\mu}}^{(a)} = \dot{\mathbf{X}}^{(a)}\mathbf{C}$ and $\dot{\boldsymbol{\mu}}^{(b)}$, we apply the mean correction factor in eq. (34) and update the projected mean in eq. (35) using $\mathbf{A}|\mathbf{\Lambda}|^{-1}$ and $\mathbf{B}|\mathbf{\Lambda}|^{-1}$.

After eigenvalue decomposition of the whitened scatter matrix of the derived samples $\mathbf{W}_t^\mathsf{H}\tilde{\mathbf{X}}_t^*\dot{\mathbf{S}}_t\tilde{\mathbf{X}}_t\mathbf{W}_t = \mathbf{\Theta}_t\mathbf{\Sigma}_t\mathbf{\Theta}_t^\mathsf{H}$ we represent the slow features $\mathbf{V}_t$ by the explicit matrices $\mathbf{A}_t|\mathbf{\Lambda}_t|^{-1}\mathbf{\Theta}_t$ and $\mathbf{B}_t|\mathbf{\Lambda}_t|^{-1}\mathbf{\Theta}_t$, and we apply our projections on either mapping a or b of novel incoming data points. A full implementation can be found in the supplementary material.

C. Running Time Analysis

We analyze the complexity of our setups thus far. For this, we assume the dimensionality of P to e larger than the number of samples N. Table II summarizes our findings.

TABLE II
COMPLEXITY OF EACH FRAME IN SEQUENTIAL DATA.

Batch SFA	$\mathcal{O}(N^3 + FN^2)$
Online SFA	$\mathcal{O}(N_\delta^3 + N_\delta Q + R(R+N_\delta)^2 + FR^2 + Q^3)$
Direct Online SFA	$\mathcal{O}(N_\delta^3 + R(R+N_\delta)^2 + FR^2)$

First, let us consider the complexity of the batch algorithm for SFA. We compute the eigenvalue decomposition of the kernel matrix to solve the PCA stage of SFA. A linear kernel function requires $\mathcal{O}(N^2)$ to compute the kernel matrix for N samples. The eigenvalue decomposition can be evaluation in $\mathcal{O}(N^3)$. The final stage, which finds the F slow feature functions, is an eigenvalue decomposition in $\mathcal{O}(FN^2)$.

The online method with reduced set expansion requires an incremental update at each time-step. In particular, we compute the eigenvalue decomposition of the kernel matrix for the novel data samples in $\mathcal{O}(N_\delta^3)$. We then combine the new data with the support set of size Q using the kernel, costing $\mathcal{O}(N_\delta Q)$, to produce a matrix of size $(R+N_\delta)\times(R+N_\delta)$ (eq. (25)), which eigenvalue decomposition we compute in $\mathcal{O}(R(R+N_\delta)^2)$. Here, R represents the number of components in the PCA. The final eigenvalue decomposition to find the slow features costs $\mathcal{O}(FR^2)$. However, not only does the complexity of the update need to be accounted for, we also require the optimization problem for the reduced set expansion. This is commonly done in $\mathcal{O}(Q^3)$ for linear kernel functions.

In the direct online SFA we benefit from the known mapping functions. Hence, the computation of the PCA stage is composed of two eigenvalue decompositions in $\mathcal{O}(N_\delta Q^3)$ and $\mathcal{O}(R(R+N_\delta)^2)$ respectively. The final state is again computed in $\mathcal{O}(FR^2)$. Notice, no optimization step is required as pre-images are not needed.

IV. Applications of Online SFA

We introduce the first SFA-based change detection which we apply to temporal video segmentation and tracking with multiple appearance models.

A. Temporal Video Segmentation

One application of incremental SFA is temporal video segmentation through change detection. In particular, given a video with multiple activities, the segmentation of these activities is closely related to finding consecutive frames which have large differences in their slow features [3].

SFA natively minimizes the slowness of a signal, see eq. (9). In video, the slowness uses the squared magnitude of the derivative signal, extracted from a sample frame. Analogously, we define the *change* of a signal $\mathbf{x}_n \in \mathcal{K}$, at time t, as the magnitude of its projected derivative, *i.e.*

$$c_t(\mathbf{x}_n) = \dot{\mathbf{x}}_n^* \mathbf{V}_t \mathbf{V}_t^* \dot{\mathbf{x}}_n = \mathbf{D}_n^\mathsf{T} \mathbf{X}_n^* \mathbf{V}_t \mathbf{V}_t^* \mathbf{X}_n \mathbf{D}_n \qquad (56)$$

where $\mathbf{X}_n\mathbf{D}_n$ is a product of matrices, which describes the derivative of $\mathbf{x}_n$, using the notation from section II-C.

When a new activity starts, the change is expected to be "unusually" large. We measure the importance of a change as a *change ratio* between the new data point $\mathbf{x}_{N_{t+1}}$ and the average change of previous data, given by

$$r_t(\mathbf{x}_{N_{t+1}}) = \frac{N_t c_t(\mathbf{x}_{N_{t+1}})}{\sum_n^{N_t} c_t(\mathbf{x}_n)}. \qquad (57)$$

Notice however, a trivial update of the mean is not possible, as c_t changes at each time interval. If we stored the whole signal in memory to compute the average at each time-step, the requirements of an online system would be violated. A sliding window could alleviate this problem, but it does not take the dynamics of the whole video into account.

Let us now present an alternative approach. Considering eq. (22) and the related update in eq. (39), we already know the eigenvalue decomposition $\mathbf{V}_t^* \mathbf{X}_t \mathbf{D}_t \mathbf{C}_t \mathbf{X}_t^\mathsf{T} \acute{\mathbf{X}}_t^* \mathbf{V}_t = \mathbf{\Theta}_t \mathbf{\Sigma}_t \mathbf{\Theta}_t^\mathsf{H}$. The eigenvalues are much related to the sum of previous changes, and we can compute eq. (57) as follows

$$r_t(\mathbf{x}_{N_{t+1}}) = \frac{N_t c_t(\mathbf{x}_{N_{t+1}})}{\mathrm{tr}(\mathbf{\Sigma}_t) + \dot{\boldsymbol{\mu}}_t^* \mathbf{V}_t \mathbf{V}_t^* \dot{\boldsymbol{\mu}}_t} \qquad (58)$$

where $\dot{\boldsymbol{\mu}}_t^* \mathbf{V}_t \mathbf{V}_t^* \dot{\boldsymbol{\mu}}_t$ handle the mean substraction by $\dot{\boldsymbol{\mu}}_t$.

With the change ratio r_t it is now possible to identify significant changes in data streams. At each time-step, we first analyze the significance of variation, and then update the SFA with the new data point. A threshold is applied to find the frames with large amount of change. These frames provide the split position in the temporal segments of the video stream. An optional median filter may be applied to smoothen the change detection. However, if immediate output is required this process may be skipped.

B. Multi Appearance Model Tracking

Building on change detection, we now propose a tracking framework that detects likely areas of drift. Although many tracking applications benefit from online learning [9], [11], [12], susceptibility to drift is a challenging problem [11]. One of the reasons drift can occur is the prolonged exposure to corrupted data, *e.g.* caused by occlusions, appearance changes or pose variations. Typically these instances harm the tracking system over time, as learned appearances are forgotten about in the online appearance model. One technique to suppress drift in tacking is the combination of multiple trackers n a unifying framework [38], [39]. In [40] an event-driven tracking system is introduced. These methods are, however, outside the scope of this paper. In our work, we want to understand the benefit of incremental SFA and its change detection algorithm to improve upon a simple tracking framework.

In [12] we introduce the direct incremental KPCA tracker (DIKT) which builds on online PCA learning as observation model. In particular, DIKT's PCA subspace of the target is updated incrementally after every 5^{th} frame. The update is composed of the tracked particles. We refer to [12] for details. We now incorporate change detection to DIKT. Figure 3 summarizes our setup. We start with one initial online model that is used for the tracking mechanism. Once a change is detected, the current version of the knowledge base is copied to create an active online model, and a dormant offline model. The online model now performs the tracking and receives updates, while the offline model is unchanged. After further changes, we create an offline and online model for each version of the appearance description and tracking is done *via* a combination of all online methods. The most similar models are merged, to satisfy the online requirements.

The following summarizes DIKT's tracking procedure and the proposed tracking with multiple appearance models. Our change detection and merging algorithm are then presented.

1) Existing Tracking Procedure: We explain DIKT's existing tracking procedure [12] as a probabilistic framework. First, let us consider the framework under linear features that do not require a kernel representation. Let $\mathcal{A}_t = \left\{\mathcal{A}_t^1, \ldots, \mathcal{A}_t^P\right\}$ be the set of P affine transformations extracted by a particle filter at time t, and $\mathcal{I}_t = \left\{\mathcal{I}_t^1, \ldots, \mathcal{I}_t^P\right\}$ be the

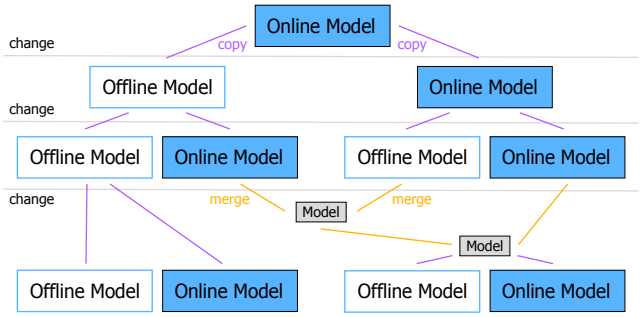

Fig. 3. Tracking with multiple appearance models, using change detection. The models are copied into an offline and online model each time a change is detected. The online models are used to generate the tracking results, while the offline models remain dormant until further changes are detected. Most similar models are merged if necessary to maintain a budget.

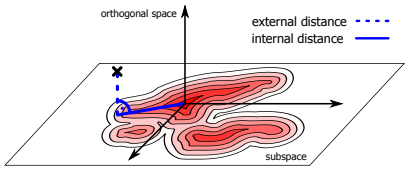

Fig. 4. Visualization of the distance between a data point and the mean of the subspace. The external distance is the length between the sample and the subspace, while the internal distance is the displacement between the sample's projection and the mean.

corresponding observations. We find the probability of $\mathcal{A}_t^p$ given $\mathcal{I}_t$ as

$$p(\mathcal{A}_t^p|\mathcal{I}_t) = p(\mathcal{A}_t^p|\mathcal{I}_t^p) \propto p(\mathcal{I}_t^p|\mathcal{A}_t^p)p(\mathcal{A}_t^p). \tag{59}$$

Hence, the system is composed out of an observation model $p(\mathcal{I}_t^p|\mathcal{A}_t^p)$ and a transformation model $p(\mathcal{A}_t^p|\mathcal{A}_{t-1}^p)$.

The transformation model is a mixture of Gaussian

$$\sum_{p'=1}^{P'} p(\mathcal{A}_{t-1}^{p'}|\mathcal{I}_{t-1}^{p'})\mathcal{N}(\mathcal{A}_{t-1}^{p'}, \Xi)|_{\mathcal{A}_t^p} \tag{60}$$

where the likelihood of previous time-steps, $p(\mathcal{A}_{t-1}^{p'}|\mathcal{I}_{t-1}^{p'})$, acts as weight, and Ξ is an independent covariance matrix, which represents the variance in horizontal and vertical displacement, rotation, scale, ratio and shew [9], [12]. Notice, the transformation model remains unchanged in this paper.

Let us consider the observation model. Probabilistic PCA [41] allows us to formulate the likelihood of a sample as [9]

$$\begin{aligned} p(\mathbf{x}) &= \left((2\pi)^d \|\mathbf{U}(\mathbf{\Lambda}^{\frac{1}{2}} - \sigma^2\mathbf{I})\mathbf{U}^\mathsf{T} + \sigma^2\mathbf{I}\|\right)^{-\frac{1}{2}} \\ & e^{-\frac{1}{2\sigma^2}(\mathbf{x}-\boldsymbol{\mu})^\mathsf{T}(\mathbf{I}-\mathbf{U}\mathbf{U}^\mathsf{T})(\mathbf{x}-\boldsymbol{\mu})} \qquad (61) \\ & e^{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^\mathsf{T}\mathbf{U}\mathbf{\Lambda}^{-\frac{1}{2}}\mathbf{U}^\mathsf{T}(\mathbf{x}-\boldsymbol{\mu})} \qquad (62) \end{aligned}$$

where $\mathbf{x}$ corresponds to the features in $\mathcal{I}_t^p$, $\mathbf{U}$ is the PCA's subspace, $\mathbf{\Lambda}^{\frac{1}{2}}$ contains the PCA's eigenvalues, $\boldsymbol{\mu}$ represents the mean of the training data, d is the dimensionality of $\mathbf{x}$, $\|.\|$ computes the determinant of a matrix, and σ^2 controls the spread. Hence, the probability of a sample being from a single PCA is explained by the reconstruction error (eq. (61)) and the inner subspace distance (eq. (62)). The PCA for the appearance model is update after every 5^{th} frame [12].

With kernelized data, the model is commonly not a well defined probability distribution. Nonetheless, as illustrated in figure 4 and in [42], we may still base our cost function on the internal and external distance of the PCA. Hence, the internal distance is given, as before, by

$$(\mathbf{x}-\boldsymbol{\mu})^*\mathbf{U}|\mathbf{\Lambda}|^{-\frac{1}{2}}\mathbf{U}^*(\mathbf{x}-\boldsymbol{\mu}) \tag{63}$$

and the external distance is given by the kernel (eq. (2))) as

$$(\mathbf{x}-\mathbf{U}\mathbf{U}^*\mathbf{x})^*(\mathbf{x}-\mathbf{U}\mathbf{U}^*\mathbf{x}). \tag{64}$$

The most probable sample is chosen and used as update.

2) Tracking with Multiple Models: As multiple active tracking models are produced, their knowledge is to be combined.

Let us assume a model with N PCAs. We take the probability of a sample as the average of multiple Gaussians. Therefore, the probability for linear data is written as

$$\begin{aligned} p(\mathbf{x}) &= \frac{1}{N(2\pi)^{\frac{d}{2}}} \sum_n^N \left(\sigma_n^{R_n-d} \prod_{r=1}^{R_n} \lambda_{nr}^{-\frac{1}{4}} \right. \qquad (65) \\ & e^{-\frac{1}{2\sigma_n^2}(\mathbf{x}-\boldsymbol{\mu}_n)^\mathsf{T}(\mathbf{I}-\mathbf{U}_n\mathbf{U}_n^\mathsf{T})(\mathbf{x}-\boldsymbol{\mu}_n)} \qquad (66) \\ & \left. e^{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_n)^\mathsf{T}\mathbf{U}_n\mathbf{\Lambda}_n^{-\frac{1}{2}}\mathbf{U}_n^\mathsf{T}(\mathbf{x}-\boldsymbol{\mu}_n)} \right) \qquad (67) \end{aligned}$$

where $\mathbf{U}_n$, $\mathbf{\Lambda}_n$, $\boldsymbol{\mu}_n$ and R_n correspond to the n^{th} PCA, λ_{nr} is the r^{th} eigenvalue of the n^{th} PCA, and the determinant is expressed as the product of the matrix's eigenvalues (eq. (65)).

For the kernel version, we employ eq. (63) and eq. (64) as before. Again, the most probable sample is used for the update of all active PCA models. Offline models are unchanged.

3) Change Detection for Tracking: We utilize the change ratio r_t from eq. (58) to detect likely areas of sudden variations, which we believe are indications of appearance changes.

In particular, a single instance of online SFA is run along side the tracking system. Its input data consists of the best tracked warped image particles as provided by the observation model (the same features as for tracking are employed). DIKT computes its PCA update after each 5^{th} frame. We update SFA at the same time. The delay not only enables more stable learning, it also allows us to impose a median filter to improve the change detection. Finally, we enforce nonconsecutive detection of changes to prevent multiple detections at a single change over several video frames.

4) Merging Appearance Models: At each change detection, we copy all previously generated PCA models to create one offline and one online version each. Corruptions only alter half of the appearance models, allowing the other half to stay uncorrupted. In such framework, however, the number of PCA models doubles at each change. Thus appearance models need to be merged for such setup to satisfy a memory budget. We detail this procedure in the following.

Let there be two data-sets $\mathbf{X}_1 = [\phi(\mathbf{z}_{11}) \cdots \phi(\mathbf{z}_{1N_1})]$ and $\mathbf{X}_2 = [\phi(\mathbf{z}_{21}) \cdots \phi(\mathbf{z}_{2N_2})]$ that make up two different PCAs. Their eigendecomposition of the kernel matrices are given by

$$\begin{aligned} \tilde{\mathbf{K}}_1^2 &= \left(\frac{1}{N_1}\tilde{\mathbf{X}}_1^*\tilde{\mathbf{X}}_1\right)^2 = \mathbf{\Omega}_1\mathbf{\Lambda}_1^2\mathbf{\Omega}_1^\mathsf{H} \qquad (68) \\ \tilde{\mathbf{K}}_2^2 &= \left(\frac{1}{N_2}\tilde{\mathbf{X}}_2^*\tilde{\mathbf{X}}_2\right)^2 = \mathbf{\Omega}_2\mathbf{\Lambda}_2^2\mathbf{\Omega}_2^\mathsf{H}. \qquad (69) \end{aligned}$$

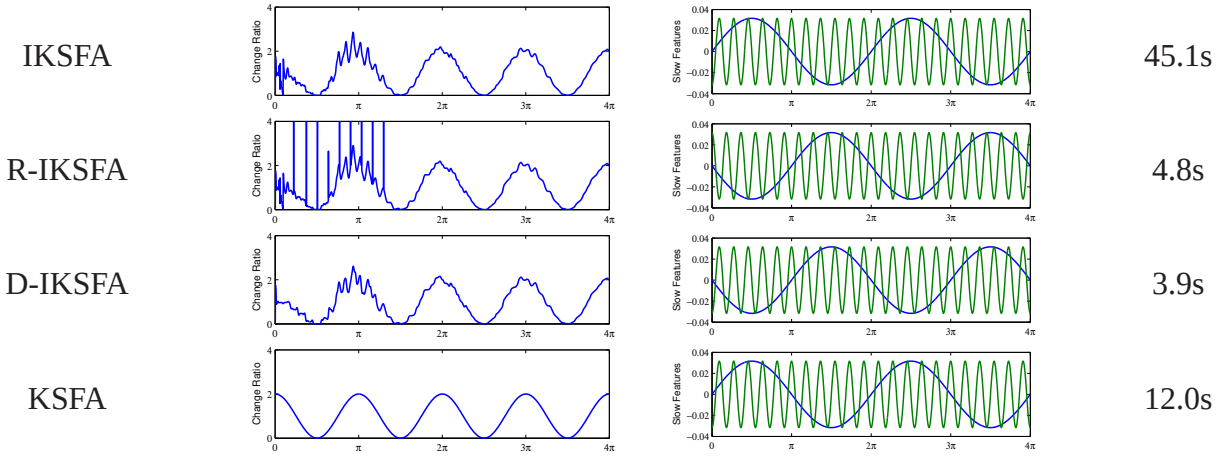

Fig. 5. Change ratio (column 2), final slow features (column 3), and learning time (column 4) are shown for online SFA (IKSFA), SFA with a reduced set of 60 elements ($Q = 10$) (R-IKSFA), direct online SFA (D-IKSFA), and batch KSFA.

The PCAs to be merged are the most similar PCAs, as given by the smallest angle difference [43]

$$e(\mathbf{U}_1, \mathbf{U}_2) = \max(R_1, R_2) - \sum_{r_1=1}^{R_1} \sum_{r_2=1}^{R_2} |\mathbf{u}_{1_{r_1}}^* \mathbf{u}_{2_{r_2}}| \qquad (70)$$

where $\mathbf{U}_1 = \begin{bmatrix} \mathbf{u}_{1_1} & \cdots & \mathbf{u}_{1_{R_1}} \end{bmatrix}$ and $\mathbf{U}_2 = \begin{bmatrix} \mathbf{u}_{2_1} & \cdots & \mathbf{u}_{2_{R_2}} \end{bmatrix}$ are the tested PCAs' subspaces.

The combination of both PCAs is quite similar to the update of the whitening projection for the PCA part of SFA in eq. (25). Thus, we find the new, combined eigenspectrum using the decomposition of

$$\tilde{\mathbf{K}}^2 = \check{\mathbf{\Omega}} \begin{bmatrix} \frac{1}{2}|\mathbf{\Lambda}_1| & \mathbf{\Omega}_1^{\mathsf{H}} \tilde{\mathbf{X}}_1^* \tilde{\mathbf{X}}_2 \mathbf{\Omega}_2 \\ \mathbf{\Omega}_2^{\mathsf{H}} \tilde{\mathbf{X}}_2^* \tilde{\mathbf{X}}_1 \mathbf{\Omega}_1 & \frac{1}{2}|\mathbf{\Lambda}_2| \end{bmatrix} \check{\mathbf{\Omega}}. \qquad (71)$$

Notice, we introduce weighted eigenvalues to give both PCAs similar importance, especially in conjunction with further future updates. The number of samples in the combined PCA is $N = \frac{N_1+N_2}{2}$ and the mean is updated accordingly.

After combining two PCAs, a single PCA model is produced which contains the knowledge base of both instances. An online and an offline version is stored to continue the tracking in conjunction with all other active models.

V. Evaluation

As a proof of concept, we apply the proposed incremental SFA to temporal video segmentation and tracking.

A. Change Detection with Synthetic Data

We test the general properties of incremental SFA for the case of change detection with synthetic data. In particular, we compare batch KSFA, which incorporates all data points at once, with incremental KSFA which learns at each time-step. As in [1], we assume an input signal $\mathbf{x}_n = \begin{bmatrix} \sin(\mathbf{y}_n) + \cos(11\mathbf{y}_n)^2 & \cos(11\mathbf{y}_n) \end{bmatrix}^{\mathsf{T}}$ where $\mathbf{y}_n$ is taken from 2000 equally distributed values in the range $[0, 4\pi]$. The corresponding slow features are to be found. With a quadratic kernel, the solution is $\mathbf{o}_n = \begin{bmatrix} \sin(\mathbf{y}_n) & \cos(11\mathbf{y}_n) \end{bmatrix}^{\mathsf{T}}$ [1].

Three versions of incremental KSFA are tested (figure 5). IKSFA uses the full data set as support – it grows larger over time. R-IKSFA employs a reduced set expansion for learning with constant memory. D-IKSFA exploits the direct equivalent mappings of the quadratic kernel (*i.e.* $a(\mathbf{x}_n) = b(\mathbf{x}_n) = \begin{bmatrix} \mathbf{x}_n(1) & \mathbf{x}_n(2) & \mathbf{x}_n(1)\mathbf{x}_n(2) & \mathbf{x}_n(1)^2 & \mathbf{x}_n(2)^2 \end{bmatrix}^{\mathsf{T}}$). Finally, we consider KSFA as ground truth, as in this setup, the complete sequence is known *a priori*.

All methods converge towards the same slow features.[5] However, they differ in their execution. IKSFA is most stable for change ratio estimation. With a reduced set, noisy results are encountered at the beginning of the sequence, when learned by R-IKSFA. D-IKSFA performs similarly to IKSFA. In terms of running time, IKSFA performs worst as the complete set of samples is required for the kernel trick at each time-step. R-IKSFA's reduced support set improves execution by a factor of 9, while D-IKSFA squeezes the learning time to 3.9s – more than 11 times faster than IKSFA, and 3 times faster than the batch version of KSFA. Notice, as preimages are easily computed for the quadratic kernel, R-IKSFA is also fast.

B. Change Detection with Real Data

We evaluate the quality of online SFA and change detection with real data. Our data set consists of the expressions in the MMI Facial Expression Database (MMI) [44]. We concatenate all videos and employ our tracker in [12] to extract aligned images (40×40 pixel) of 60 activities. The resulting sequence consists of 4182 frames, cropped to the subject's face.

We analyze the proposed framework with different kernels. The original image data input are lexicographical pixel intensities in $[0, 1]$. We utilize the direct input features (Linear), the quadratic kernel (Quadratic), the standard Gaussian kernel (Gaussian) and our Krein space kernel in [12] (Krein). The deviation of the Gaussian is fixed to $\frac{1}{N(N-1)} \sum_{n=1}^{N} \sum_{n'=1}^{N} ||\mathbf{x}_n - \mathbf{x}_{n'}||^2$, where N is the number of samples [45], and the parameters of Krein follow [12].

1) Incremental Learning Behavior: To quantitatively measure the learning behavior of online SFA, we compare it to its offline equivalent. In particular, we compute the angle

[5] As in standard PCA, the sign of the components is irrelevant.

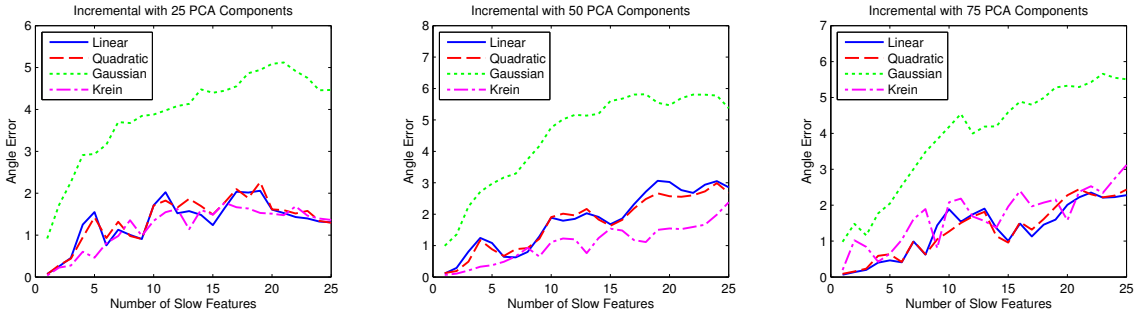

Fig. 6. Angle error between the batch learned and incrementally learned subspaces for different number of components in the PCA (R) with varying number of slow features. No reduced set, or forgetting factor is applied.

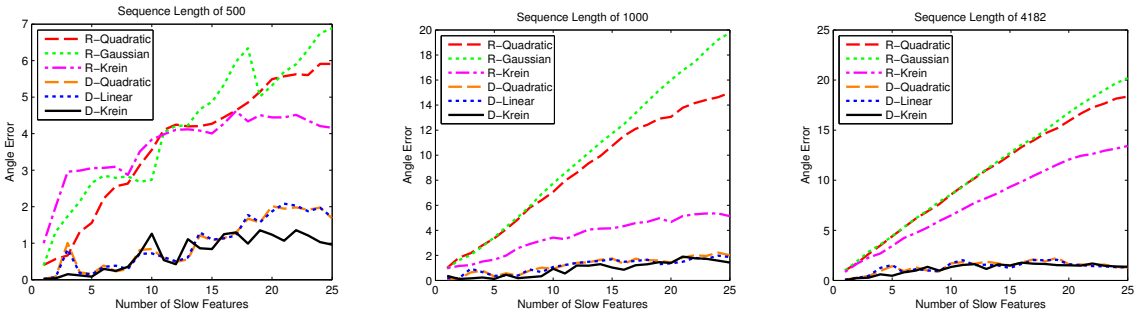

Fig. 7. Angle error between the batch learned and incrementally learned subspaces for reduced set representation of 260 preimages (prefix R) and direct mappings (prefix D). Results are shown for different sequence sizes and a PCA of 25 components.

difference between the offline and online subspaces [43]

$$e(F) = F - \sum_{f_1=1}^{F} \sum_{f_2=1}^{F} |\mathbf{v}_{1_{f_1}}^{*} \mathbf{v}_{2_{f_2}}| \qquad (72)$$

where F is the number of slow features evaluated, and $\mathbf{v}_{1_{f_1}}$ and $\mathbf{v}_{2_{f_2}}$ are the individual slow feature projections of the compared SFAs.

Figure 6 shows the angle error with different numbers of components in the PCA and varying number of slow features. Here, the kernel versions with complete support set are employed, and the forgetting factor is set to 1 (no forgetting) to facilitate comparison. The Linear, Quadratic and Krein kernel perform equally well. The Gaussian kernel has a slightly increased error rate. In general, all perform with low errors and they are independent of the number of components in the PCA stage of SFA.

Next, we evaluate the incremental SFA with reduced sets (prefix R) or direct mapping (prefix D). We fix the number of components in the PCA to $R = 25$ and request a budget of 260 preimages in the reduced set, *i.e.* $Q = 10$. In figure 7 we show the angle error when learning is performed on different sequence lengths. The algorithms work well for short videos (about 500 frames). Nonetheless, approximation errors in the reduced set representation seem unavoidable and introduce lower performance in longer sequences. Here the algorithms with direct mapping are more suitable, as low errors are achieved for any sequence length.

2) Change Detection: We now apply the change detection algorithm of section IV-A to our dataset. MMI consists of facial expressions, labeled by onset, apex and offset. The onset and offset indicate the start and end of an expression

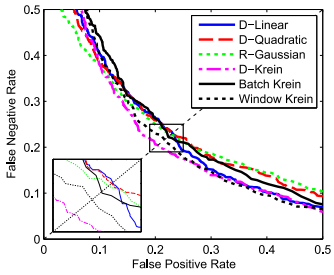

Fig. 8. False positive over false negative rate of different kernel setups. The batch version of Krein learns from the whole sequence *a priori*, while Window Krein performs the lerning as batch in a window.

respectively. We utilize these labels as ground truth as they mark the frames in which the activity in the video changes. The equal error rate of false negatives over false positives then reveal the performance of individual setups.

Linear, Quadratic, Gaussian and Krein are compared. We optimize with respect to the number of PCA components R, the forgetting factor w and the number of slow features F. A median filter of 8 frames is applied to the output and direct mappings are used where possible. The Gaussian kernel function receives a fixed budget of $Q = 10$. All methods perform best with $w = 0.996$ (≈ 250 frames), and $F = 3$. However, the ideal number of PCA components varies for each method ($R = 15$ for Linear, $R = 30$ for Quadratic, $R = 50$ for Gaussian, $R = 10$ for Krein). We also include the batch version of Krein (Batch Krein), for which we compute the change ratio with *all samples known a priori*– i.e. we learn the slow features from the complete sequence. Finally, a batch algorithm from a sliding window of 250 frames (equivalent to the forgetting factor) is compared against (Window Krein).

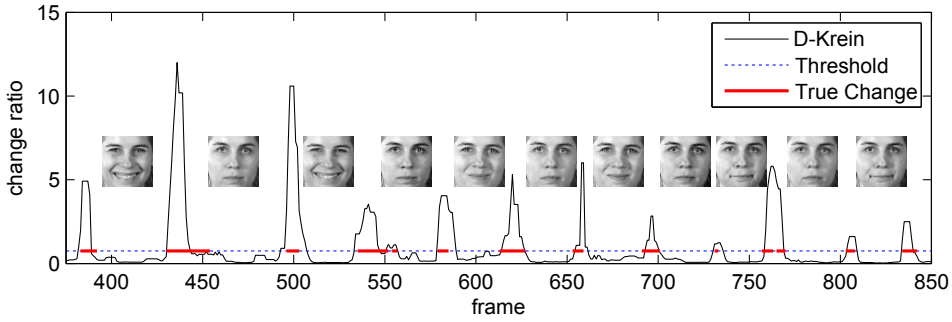

Fig. 9. Change detection for Krein with direct mapping, forgetting factor $w = 0.996$ and 10 PCA components.

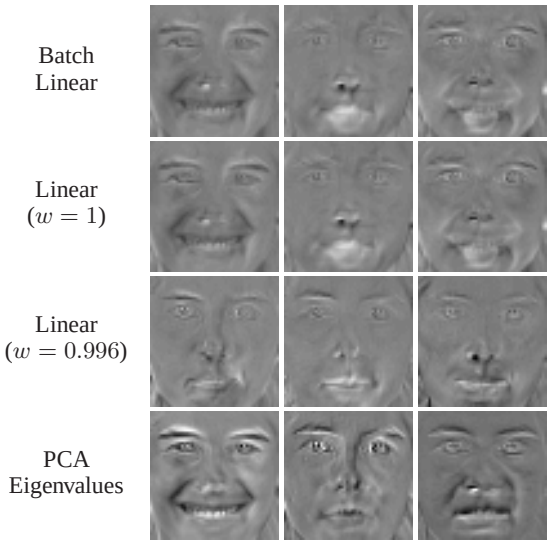

Fig. 10. The 3 slowest features after 900 frames using the linear kernel. Batch learning is compared to online learning with and without forgetting. The eigenvalues after PCA learning are visualized in the bottom row.

The false negative over false positive rates are shown in figure 8. The quadratic expansion has a slightly better equal error rate than linear features, while the Gaussian kernel is the most performant positive definite setup. The best results are achieved by the direct version of incremental SFA with Krein kernel (D-Krein). The advantage of our Krein kernel stem not only from its domain-specific design, its direct mappings are fast to compute (in linear complexity) and less PCA components are required to outperform other systems. For example, as comparison, the direct mapping of the quadratic expansion is polynomial, and 30 PCA components are employed. Figure 9 shows an excerpt from the results for Krein.

The batch version of Krein (Batch Krein) performs with reduced results to its online or windowed equivalent. The online version is advantaged as the forgetting factor allows the system to adapt to different parts of the video. Similarly, with a sliding window, only recent frames are used. To visualize the difference, we show the top 3 projections after frame 900 for the Linear kernel (i) as batch setup, (ii) with $w = 1$ (no forgetting), and (iii) with $w = 0.996$ (the best forgetting value) in figure 10. Here Linear is chosen to aid visualization, as the projections remain in the original space. We compare (i) to (ii). The resulting projections are virtually equivalent, which validates our update procedure. With forgetting, the effect of w becomes apparent, as the projections are most relevant to later

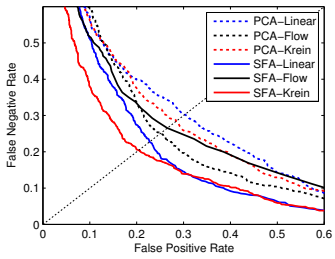

Fig. 11. False positive over false negative rate comparison for optical flow (Flow), intensity values (Linear) and our kernel (Krein) as PCA and SFA.

expressions. Notice, the smile around frame 530 and visible for (i) and (ii) is no longer in the subspace.

In contrast to Window Krein, our proposed D-Krein uses a forgetting factor and incremental learning. Hence, our approach is much faster, as it uses the previously learned model from the frames before. On the other hand, Window Krein resets the complete optimization at each time-step. Notice also, the forgetting factor only reduces the weight of previous frames – it does not remove them – allowing for significant components to be retained.

We conclude this part of the evaluation with a selection of videos from different scenarios. Figure 12 shows the temporal segmentation of 2 yoga sequences, taken from YouTube (http://www.youtube.com/watch?v=ziVctQnyvwE) and 2 examples from the ballet dataset in [46]. Note, our algorithm finds the visually distinct partitions.

3) Comparison to Optical Flow and PCA: In the final part of the experiments for change detection, we analyze our framework in comparison to alternative methods. A typical approach to finding drift in many tracking frameworks is the distance between the current frame and the learned eigenspace. Similarly, this can be applied to change detection *via* the reconstruction error given by

$$e(R) = \|\mathbf{x} - \mathbf{U}_R \mathbf{U}_R^{\mathsf{H}} \mathbf{x}\|^2 \tag{73}$$

where $\mathbf{U}_R$ is the reduced eigenspace of previous samples, and $\mathbf{x}$ denotes the tested input. The eigenvectors can be incrementally learned by the methods in [9], [12].

Additionally, let us consider another feature input, known as optical flow. Optical flow has proven advantageous for describing the dynamics of a video sequence [47]. While the computation of such features is expensive (sub-realtime with the source code of [47]), we will compare it to our SFA with Krein space kernel in this section.

We use the following setups in our experiment: PCA learn-

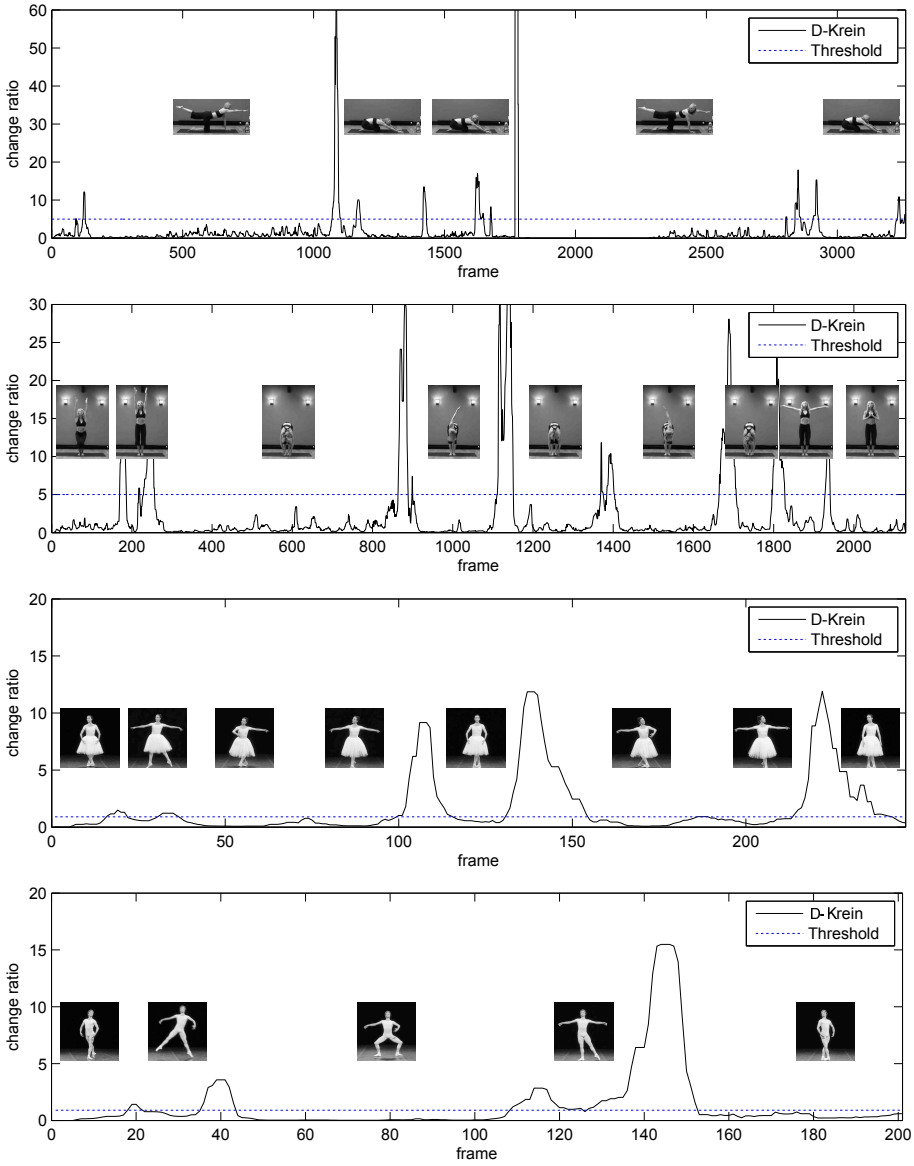

Fig. 12. Frames of the segmentation for yoga and ballet scenes. SFA's change detection finds the visually distinct parts of the videos. Our Krein kernel with direct mapping is used with the same parameters as before.

ing with intensity values (PCA-Linear), optical flow (PCA-Flow) and our kernel (PCA-Krein), which are compared to SFA with intensity values (SFA-Linear), optical flow (SFA-Flow) and our kernel (SFA-Kernel). Figure 11 illustrates the results. Notice, change detection solely based on PCA without the added temporal information (*i.e.* PCA-Linear and PCA-Krein) is inferior to other methods. Less efficient PCA-Flow on the other hand is competitive in terms of equal error rate. SFA inherently incorporates temporal information as the derivative scatter matrix is considered in its optimization. Indicative of this is the good performance of SFA-Linear and SFA-Krein. SFA-Flow is less performant, as optical flow is less valuable to SFA.

Figure 10 visualizes the learned subspaces of SFA and PCA. Notice, while SFA finds the important features, which change slowest over time. In contrast, PCA computes the features that best describe the samples, without the temporal information. For instances, the eyebrows and contours around the nose of the subject are more prominently modeled with PCA.

C. Tracking with Change Detection

In our final experiment, we evaluate our tracking framework with change detection (CD-DIKT), as proposed in section IV-B. Here, we focus on the analysis of the gain, that change detection using SFA brings to adaptive tracking. Hence, we use DIKT [12] as base line comparison.

The performance is tested on the 9 videos also used in [12]. These videos contain drastic changes of the targets appearance, including pose variation, occlusions, and nonuniform illumination. All videos have 3-7 fiducial points which allow for quantitative performance evaluation. We use the root mean square (RMS) errors between the true and the estimated locations of these points as performance indication. Our choice of parameters follows [12]. For all videos, we fix the translation model of the particle filter tracking framework to the values provided by DIKT's source code at http://www.doc.ic.ac.uk/~sl609/dikt.

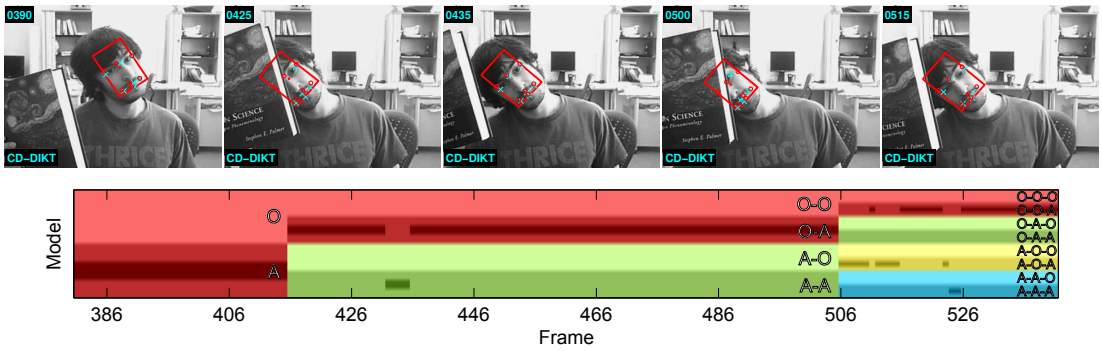

Fig. 13. Tracking results and models of appearances. We encode offline and active models with a sequence of Os and As to highlight their active learning periods (*e.g.* A-O-A learned with data before and after the occlusion, but not during). The model that provides the highest score is highlighted.

The parameters of the change detection in CD-DIKT are set for each video specifically, as the thresholds of the change ratio vary video-dependently. All other parts of DIKT and CD-DIKT are identical. See videos in supplement.

TABLE III
MEAN RMS ERRORS FOR DIKT AND DIKT WITH CHANGE DETECTION.

	Vid1	Vid2	Vid3	Vid4	Vid5	Vid6	Vid7	Vid8	Vid9
DIKT	4.44	2.77	2.58	(lost)	(lost)	3.79	2.19	2.75	(lost)
CD-DIKT	4.15	2.60	2.34	5.95	(lost)	3.68	2.18	2.68	6.70

Table III lists the mean RMS errors for each video. Consistently, CD-DIKT improves upon DIKT. While the RMS error is only marginally better, CD-DIKT is more robust. In particular, CD-DIKT is capable of tracking Vid4 and Vid9 successfully as it utilizes previously seen appearances. Finally we note that CD-DIKT is highly parallelizable due to its independent set of appearance modules and the additional online SFA. Hence, a similar execution time to DKT can be achieved, utilizing multiple threads.

Figure 13 shows an example of Vid4. Not visible in the figure, the first change splits the initial model into an active model (A), and a dormant offline model (O). In frame 415 a change is detected, which reactivates the dormant model (O-A) and the already active model is split into offline and online versions, A-O and A-A respectively. The tracker consults both active models, *i.e.* O-A and A-A, during the occlusion. Notice however, the more naïve O-A is favored during the occlusion as it knows less about the face. Around frame 435 the face is fully visible again, and model A-A kicks in for a short period of time. Finally, after the second change detection in frame 505, initially model A-O-A provides most confidence. This model consists of data before the occlusion, and was idle during the input of corrupted data. Finally O-O-A is favored thereafter, as this model is significantly trained from data after the corruptions in frame 415 to 505.

VI. CONCLUSION

In is paper, we proposed an exact kernel slow feature analysis (KSFA) framework for arbitrary Krein space kernels. We formulated general online KSFA which employs a reduced set expansion to fulfill budget requirements. Finally, by utilizing a special kind of kernel family, we formulated an exact online KSFA for which no reduced set is required.

We apply our online SFA and develop the first SFA-based change detection algorithm for stream data. This framework is employed for temporal video segmentation and tracking. When applied to synthetic and real data streams, our method successfully segments the input using change detection. Combined with an online learning tracking system, change detection improves upon systems without such detection, in our evaluation.

REFERENCES

[1] L. Wiskott and T. Sejnowski, "Slow Feature Analysis: Unsupervised Learning of Invariances," *Neural Comput.*, vol. 14, no. 4, pp. 715 – 770, Apr 2002.
[2] P. Berkes and L. Wiskott, "Slow Feature Analysis Yields a Rich Repertoire of Complex Cell Properties," *J. Vis.*, vol. 5, no. 6, pp. 579 – 602, Jul 2005.
[3] A. Nater, H. Grabner, and L. Van Gool, "Temporal Relations in Videos for Unsupervised Activity Analysis," in *Machine Learning*, 2011, pp. 78–86.
[4] V. Kompella, M. Luciw, and J. Schmidhuber, "Incremental Slow Feature Analysis: Adaptive Low-Complexity Slow Feature Updating from High-Dimensional Input Streams," *Neural Comput.*, vol. 24, no. 11, pp. 2994 – 3024, Nov 2012.
[5] M. Franzius, N. Wilbert, and L. Wiskott, "Invariant Object Recognition with Slow Feature Analysis," in *Int. Conf. Artificial Neural Networks, ICANN'08*, 2008, pp. 961 – 970.
[6] L. Zafeiriou, M. Nicolaou, S. Zafeiriou, S. Nikitidis, and P. M., "Learning Slow Features for Behaviour Analysis," in *IEEE Int. Conf. Computer Vision, ICCV'13*, 2013.
[7] P. Földiák, "Learning invariance from transformation sequences," *Neural Comput.*, vol. 3, pp. 194–200, 1991.
[8] A. Levy and M. Lindenbaum, "Sequential Karhunen-Loeve Basis Extraction and its Application to Images," *IEEE Trans. Image Process.*, vol. 9, no. 8, pp. 1371 – 1374, Aug 2000.
[9] D. Ross, J. Lim, R. Lin, and M. Yang, "Incremental Learning for Robust Visual Tracking," *Int. J. Comput. Vis.*, vol. 77, no. 1, pp. 125 – 141, 2008.
[10] T. Chin and D. Suter, "Incremental Kernel Principal Component Analysis," *IEEE Trans. Image Process.*, vol. 16, no. 6, pp. 1662 – 1674, Jun 2007.
[11] B. Babenko, M. Yang, and S. Belongie, "Robust Object Tracking with Online Multiple Instance Learning," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 33, no. 8, pp. 1619 – 1632, Aug 2011.
[12] S. Liwicki, S. Zafeiriou, G. Tzimiropoulos, and M. Pantic, "Efficient Online Subspace Learning with an Indefinite Kernel for Visual Tracking and Recognition," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 23, no. 10, pp. 1624–1636, Oct 2012.
[13] J. Weng, Y. Zhang, and W. Hwang, "Candid Covariance-Free Incremental Principal Component Analysis," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 25, no. 8, pp. 1034 – 1040, Aug 2003.
[14] T. Chen, S. Amari, and N. Murata, "Sequential Extraction of Minor Components," *Neural Processing Letters*, vol. 13, no. 3, pp. 195 – 201, Jun 2001.
[15] W. Böhmer, S. Grünewälder, H. Nickisch, and K. Obermayer, "Regularized Sparse Kernel Slow Feature Analysis," in *European Conf. Machine Learning, ECML'11*, 2011, pp. 235 – 248.

[16] K. Kim, M. Franz, and B. Schölkopf, "Iterative kernel principal component analysis for image modeling," vol. 27, no. 9, pp. 1351 – 1366, Sep 2005.
[17] S. Günter, N. Schraudolph, and S. Vishwanathan, "Fast Iterative Kernel Principal Component Analysis," *J. Mach. Learn. Res.*, vol. 8, pp. 1893 – 1918, Aug 2007.
[18] P. Pokharel, W. Liu, and J. Principe, "Kernel least mean square algorithm with constrained growth," *Signal Processing*, vol. 89, no. 3, pp. 257–265, Mar 2009.
[19] W. Liu, I. Park, and J. Príncipe, "An Information Theoretic Approach of Designing Sparse Kernel Adaptive Filters," vol. 20, no. 12, pp. 1950 – 1961, Dec 2009.
[20] S. van Vaerenbergh, M. Lazaro-Gredilla, and I. Santamaria, "Kernel Recursive Least-Squares Tracker for Time-Varying Regression," vol. 23, no. 8, pp. 1313 – 1326, Aug 2012.
[21] W. Gao, J. Chen, C. Richard, and J. Huang, "Online Dictionary Learning for Kernel LMS," *IEEE Trans. Signal. Process.*, vol. 62, no. 11, pp. 2765–2777, Jun 2014.
[22] J. Kivinen, A. Smola, and R. Williamson, "Online Learning with Kernels," *IEEE Trans. Signal. Process.*, vol. 52, no. 8, pp. 2165 – 2176, Aug 2004.
[23] C. Richard, J. Bermudez, and P. Honeine, "Online Prediction of Time Series Data with Kernels," *IEEE Trans. Signal. Process.*, vol. 57, no. 3, pp. 1058 – 1067, Mar 2009.
[24] A. Singh, N. Ahuja, and P. Moulin, "Online learning with kernels: Overcoming the growing sum problem," in *IEEE Int. Workshop Machine Learning, MLSP'12*, 2012, pp. 1 – 6.
[25] W. Chu, F. Zhou, and F. De la Torre, "Unsupervised Temporal Commonality Discovery," in *European Conf. Computer Vision, ECCV'12*, 2012, pp. 373–387.
[26] P. Turaga, A. Veeraraghavan, and R. Chellappa, "Unsupervised View and Rate Invariant Clustering of Videosequences," *Comput. Vis. Image Underst.*, vol. 113, no. 3, pp. 353 – 371, Mar 2009.
[27] M. Hoai, Z. Lan, and F. De la Torre, "Joint Segmentation and Classification of Human Actions in Video," in *IEEE Conf. Computer Vision and Pattern Recognition, CVPR'11*, 2011, p. 3265 3272.
[28] F. Zhou, F. De la Torre, and J. Cohn, "Unsupervised Discovery of Facial Events," in *IEEE Conf. Computer Vision and Pattern Recognition, CVPR'10*, 2010, pp. 2574 – 2581.
[29] S. Liwicki, S. Zafeiriou, and M. Pantic, "Incremental Slow Feature Analysis with Indefinite Kernel for Online Temporal Video Segmentation," in *Asian Conf. Computer Vision, ACCV'12*, 2012, pp. 162–176.
[30] B. Schölkopf, A. Smola, and K.-R. Müller, "Kernel Principal Component Analysis," in *Int. Conf. Artificial Neural Networks, ICANN'97*, 1997, pp. 583 – 588.
[31] J. Shawe-Taylor and N. Cristianini, *Kernel Methods for Pattern Analysis*. Cambridge University Press, 2004.
[32] B. Hassibi, A. Sayed, and T. Kailath, "Linear Estimation in Krein Spaces. I. Theory," *IEEE Trans. Automat. Contr.*, vol. 41, no. 1, pp. 18 –33, Jan 1996.
[33] E. Pękalska and B. Haasdonk, "Kernel Discriminant Analysis for Positive Definite and Indefinite Kernels," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 31, no. 6, pp. 1017 – 1032, Jun 2009.
[34] T. Kim, S. Wong, B. Stenger, J. Kittler, and R. Cipolla, "Incremental Linear Discriminant Analysis using Sufficient Spanning Set Approximations," in *IEEE Conf. Computer Vision and Pattern Recognition, CVPR'07*, 2007, pp. 124 – 131.
[35] K. Fukunaga, *Introduction to Statistical Pattern Recognition*. Academic Press, 1990.
[36] B. Schölkopf, S. Mika, C. Burges, P. Knirsch, K.-R. Müller, G. Rätsch, and A. Smola, "Input Space Versus Feature Space in Kernel-Based Methods," *IEEE Trans. Neural Netw.*, vol. 10, no. 5, pp. 1000 – 1017, Sep 1999.
[37] C. Burges, "Simplified Support Vector Decision Rules," in *Int. Conf. Machine Learning, ICML'99*, 1996, pp. 71 – 77.
[38] B. Stenger, T. Woodley, and R. Cipolla, "Learning to Track with Multiple Observers," in *IEEE Conf. Computer Vision and Pattern Recognition, CVPR'09*, 2009, pp. 2647 – 2654.
[39] Y. Gao, R. Ji, L. Zhang, and A. Hauptmann, "Symbiotic Tracker Ensamble toward a Unified Tracking Framework," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 24, no. 7, pp. 1122 – 1131, Jun 2014.
[40] A. Doulamis, "Event-driven video adaptation: A powerful tool for industral video supervision," vol. 69, no. 2, pp. 339 – 358, Mar 2014.
[41] M. Tipping and C. Bishop, "Probabilistic Principal Component Analysis," *J. R. Stat. Soc. Series B Stat. Methodol.*, vol. 61, no. 3, pp. 611 – 622, 1999.
[42] B. Moghaddam and A. Pentland, "Probabilistic Visual Learning for Object Representation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 19, no. 7, p. 696, Jul 1997.
[43] H. Gunawan, O. Neswan, and W. Setya-Budhi, "A Formula for Angles between Subspaces of Inner Product Spaces," *Beitr. Algebra Geom.*, vol. 46, no. 2, pp. 311 – 320, 2005.
[44] M. Valstar and M. Pantic, "Induced Disgust, Happiness and Surprise: an Addition to the MMI Facial Expression Database," in *Int. Conf. Language Resources and Evaluation, LREC'10*, 2010, pp. 65 – 70.
[45] J. Kwok and I. Tsang, "The Pre-Image Problem in Kernel Methods," *IEEE Trans. Neural Netw.*, vol. 15, no. 6, pp. 1517 – 1525, Nov 2004.
[46] A. Fathi and G. Mori, "Action Recognition by Learning Mid-level Motion Features," in *IEEE Conf. Computer Vision and Pattern Recognition, CVPR'08*, 2008.
[47] D. Sun, S. Roth, and M. Black, "Secrets of optical flow estimation and their principles," in *IEEE Conf. Computer Vision and Pattern Recognition, CVPR'10*, 2010, pp. 2432 – 2439.

Stephan Liwicki (S'11) is now a post-doctoral research scientiest in 3D reconstruction and online learning for computer vision with the Department of Engineering, University of Oxford, UK. He is alumnus of two of the UK's most respected universities, as he completed his M.Phil. with the University of Cambridge, and his Ph.D. with the Imperial College London. During his studies, Dr Liwicki has been recognised on multiple occasions for his abilities, and he received several scholarships and fellowships during his research career. Most notably, he was awarded the prestigious Intel Fellowship which honours the most promising computing and engineering Ph.D. students throughout Europe.

Stefanos P. Zafeiriou (M'09) is a senior lecturer (associate professor) in pattern recognition/statistical machine learning for computer vision with the Department of Computing, Imperial College London, UK. In 2011, he was a recipient of the prestigious Junior Research Fellowships from Imperial College London to start his own independent research group. He received various awards during his doctoral and post-doctoral studies. Currently, he serves as an associate editor of the IEEE Trans. on Cybernetics and the Image and Vision Computing Journal. He has been a guest editor of over five journal special issues and co-organized over five workshops/special sessions in top venues, such as CVPR/FG/ICCV/ECCV. He has co-authored over 40 journal papers mainly on novel statistical machine learning methodologies, such as 2D/3D face analysis, deformable object fitting and tracking, shape from shading, and human behaviour analysis, published in the most prestigious journals in his field of research, and many papers in top conferences. His students are frequent recipients of highly competitive fellowships, such as the Google Fellowship, the Intel Fellowship, and the Qualcomm Fellowship.

Maja Pantic (M98SM06-F12) is a professor in affective and behavioral computing in the Department of Computing at Imperial College London, UK, and in the Department of Computer Science at the University of Twente, the Netherlands. She currently serves as the editor in chief of Image and Vision Computing Journal and as an associate editor for both the IEEE Transactions on Pattern Analysis and Machine Intelligence and the IEEE Transactions on Affective Computing. She has received various awards for her work on automatic analysis of human behavior, including the European Research Council Starting Grant Fellowship 2008 and the Roger Needham Award 2011. She is a fellow of the IEEE.