# Online Action Recognition Using Covariance of Shape and Motion

Igor Kviatkovsky[a], Ehud Rivlin[a], Ilan Shimshoni[b]

[a]*Department of Computer Science, Technion - Israel Institute of Technology, Technion City, Haifa 32000, Israel*
[b]*Department of Information Systems, University of Haifa, Carmel Mount, Rabin building, Haifa 31905, Israel*

**Abstract**

We propose a novel approach for online action recognition. The action is represented in a low dimensional (15D) space using a covariance descriptor of shape and motion features - spatio-temporal coordinates and optical flow of pixels belonging to extracted silhouettes. We analyze the applicability of the descriptor for online scenarios where action classification is performed based on incomplete spatio-temporal volumes. In order to enable our online action classification algorithm to be applied in real time, we introduce two modifications, namely the incremental covariance update and the on demand nearest neighbor classification. In our experiments we use quality measures, *e.g.*, latency, especially designed for the online scenario to report the algorithm's performance. We evaluate the performance of our descriptor on standard, publicly available datasets for gesture recognition, namely the Cambridge-Gestures dataset and the ChaLearn One-Shot-Learning dataset and show that its performance is comparable to the state-of-the-art despite its relative simplicity.

*Keywords:* Online Action Recognition, Gesture Recognition, Covariance Descriptor, Incremental Covariance Computation

## 1. Introduction

Automatic recognition of actions performed by humans is a challenging task with applicability to video surveillance, video summarization, natural user inter-

*Email address:* kviat@cs.technion.ac.il (Igor Kviatkovsky)

face and gaming. Typically, a distinction is made between "action" and "activity" [1]. Action is an atomic and self contained motion performed by a single subject, *e.g.*, jumping, hand waving, whereas an activity is a series of actions bearing a more complex meaning typically involving more than one subject, *e.g.*, a team scoring a goal in a soccer game. In this work we consider the problem of action recognition. Even more specifically, we are interested in interactive applications such as Natural User Interface (NUI) and gaming, where actions are to be detected and classified on the fly and the system response time is of great importance.

Action recognition has been extensively studied during the past two decades [1]. In the early works on action recognition the body or the hand appearance were learned and later recognized in video sequences. The action's temporal aspect was handled by training a parametric model such as the Hidden Markov Models (HMM) [2, 3]. Later on the community switched its attention to nonparametric approaches describing actions in terms of shape and motion patterns. Bobick and Davis [4] represent actions using Motion History Image (MHI) temporal templates, a weighted sum of binary silhouette masks, describing the action as a shape evolution in time. This representation was found to be discriminative enough for several simple action classes such as aerobics exercises. Blank *et al*. [5] use silhouette masks to represent an action as a space-time volume and extract features using the Poisson equation to compare these volumes. The major drawback of using shape is the assumption of availability of the silhouette masks, commonly extracted using background subtraction. To overcome this issue, Efros *et al*. [6] suggest motion descriptors based on optical flow computed over the entire bounding box in low resolution video sequences. Ali and Shah [7] derive kinematic features from optical flow and use them to represent spatio-temporal volumes in low dimensional space. Zelnik-Manor and Irani [8] use a nonparametric descriptor, based on normalized spatio-temporal gradient histograms extracted in different temporal scales and successfully apply it for action detection and clustering.

Following their success in still images, several extensions of local descriptors [9] to the spatio-temporal case were employed for action recognition. Laptev and Lindeberg [10] use an extension of Harris interest points to represent and recognize actions. Dollar *et al*. [11] learn a dictionary of spatio-temporal interest points and represent the action as a bag-of-words over the learned dictionary. Niebles *et al*. [12] use a similar approach, combining the bag-of-words approach with probabilistic graphical models.

Several recent works use tensors to represent space-time volumes. Kim and Cipolla [13] extend the Canonical Correlation Analysis (CCA) to tensors and use it to measure the similarity between two video volumes avoiding an explicit mo-

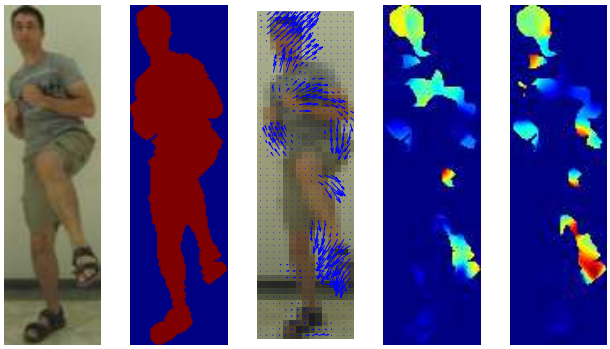
tion estimation. Lui *et al*. [14] represent the video volume tensor as a point on a product manifold and to classify actions using the nearest neighbor in this space. In a follow up work [15] the authors employ least squares regression achieving impressive results in application to gesture recognition including the challenging problem of one-shot-learning [16], where the action class has to be recognized based on a single action instance.

The covariance descriptor [17, 18], originally designed for object detection, was recently successfully employed for action recognition [19, 20, 21]. Guo *et al*. [19] use the covariance descriptor in kinematic feature space [7] to represent spatio-temporal volume. In their work on pedestrian detection, Tuzel *et al*. [18] show that covariance descriptors lie in a nonlinear Riemannian space. Harandi *et al*. [20] use a Riemannian Locality Preserving Projection (RLPP) technique to map the covariance descriptors into the euclidean space and to apply standard classification methods. In a follow up work by Sanin *et al*. [21] the RLPP was used with AdaBoost to learn a set of covariance descriptors giving impressive results in classification of a specific set of actions. A significant drawback of this approach is its learning phase making it inapplicable to the one-shot-learning scenario.

While classical action recognition is typically concerned with classifying actions using a full video volume, the goal of online action recognition is to classify actions on the fly. Albanese *et al*. [22] propose a system for online complex activity recognition using a specially designed logical language to describe the activity. In [23] the authors present a method for low latency online action recognition based on detecting distinctive canonical poses in the 3D motion of the subject's skeleton.

### *1.1. Our Approach*

In this work we present an efficient descriptor for action recognition and propose an algorithm for online action detection and recognition applicable in natural user interface and gaming scenarios. Motivated by the prior art [4, 5, 6, 7], we use the silhouette shape and motion properties to derive our action descriptor. Figures 1(a-c) show a frame from a video capturing an action, a corresponding silhouette and the computed optical flow vector field. The optical flow [24] is computed between the current and the consecutive frames assuming translation only. Figures 1(d,e) depict heat maps representing the optical flow's horizontal and vertical components. The hotter (more red) the pixel is, the more dynamic it is in terms of its horizontal or vertical velocity. Our action descriptor is based on the heat maps' evolution over time. We represent each pixel belonging to the



(a) (b) (c) (d) (e)

Figure 1: Kick action. (a) - Video frame bounding box. (b) - Silhouette obtained using background subtraction. (c) - Lucas-Kanade [24] optical flow. (d,e) - Horizontal/vertical optical flow components magnitude of pixels belonging to the silhouette.

silhouettes' series as a vector of its spatio-temporal coordinates and velocity. The action is represented as an empirical covariance matrix of such vectors [17]. Although the resulting descriptor is of low dimensionality (15 unique values only), we show experimentally that its discriminating capabilities are comparable and sometimes favorable to much more complex descriptors used by the state-of-the-art algorithms.

### *1.2. Contributions*

Our approach differs from previous works in several important aspects:

1. We use an extension of the covariance descriptor to the temporal domain to represent a spatio-temporal volume in an exceptionaly low dimensional (15D) space. We analyze the applicability of the descriptor to online scenario where action classification is performed based on incomplete spatio-temporal volumes.
2. We demonstrate that rough statistics of shape and motion (spatio-temporal coordinates and optical flow), computed on pixels belonging to regions of interest, provide a sufficiently rich representation for human body action and hand gesture classification. This is in contrast with [13] which uses a much richer method of Canonical Correlation Analysis (CCA), to represent the statistics of the volume and [7, 19] who use more complex kinematic features based on optical flow.
3. Unlike [14, 13] we do not assume a fixed number of frames in the spatio-temporal volume. Also, our approach does not rely on extensive learning

procedures applied, for example, in [21, 20, 13]. Thanks to this our approach may be applied in the challenging one-shot-learning scenario [16].

4. Our online action recognition algorithm uses an incremental covariance descriptor computation and on demand nearest neighbor classification, which significantly improve its performance and make it suitable for real time applications. Finally, the algorithm is relatively simple, easy to implement and computationally efficient.

Additional contributions of this work are the experimental validation of our approach on two standard, publicly available datasets for gesture recognition, on one publicly available dataset for action recognition and on our own private dataset for online action recognition. We propose several quality measures for action recognition in the online scenario, which differ from those usually used in the classical action categorization, and use them for the evaluation of our algorithm. We will make both our dataset and the code available for public use.

## 2. The *CovAct* Descriptor

We will now define the *CovAct*, the covariance descriptor for action recognition, and show its applicability for online action recognition. Let $V = \{I_t\}_{t=1}^{M}$ denote a sequence of $M$ video frames constituting an action and let $S = \{s_t\}_{t=1}^{M}$ denote a corresponding sequence of $M$ binary silhouette masks of the figure performing the action, *e.g.*, Figure 1(b). Let $(L_t, T_t, W_t, H_t)$ denote the $s_t$'s bounding box, where $(L_t, T_t)$ are its top left corner coordinates and $(W_t, H_t)$ are its width and height. We normalize the spatio-temporal coordinates $(x, y, t)$ of each point inside $s_t$ relative to the bounding box size and the action's duration $(x', y', t') = (\frac{x-L_t}{W_t}, \frac{y-T_t}{H_t}, \frac{t}{M})$. This normalization step provides partial invariance to object-camera distance and action duration. Besides the normalized spatio-temporal coordinates, for each point $(x, y)$ belonging to $s_t$, we compute its Lucas-Kanade [24] optical flow $(u_t(x, y), v_t(x, y))$ using $I_t$ and $I_{t+1}$, assuming translation only. Figures 2(a,b) show an example of normalized horizontal and vertical optical flow heat maps corresponding to the frame in Figure 1. Thus, each pixel $(x_i, y_i, t_i)$ belonging to the silhouette sequence $S$ of the spatio-temporal volume $V$ is represented as a 5-dimensional vector:

$$z_i = [x'_i, y'_i, t'_i, u_{t_i}(x_i, y_i), v_{t_i}(x_i, y_i)]\,, \tag{1}$$

where $(x'_i, y'_i, t'_i)$ are the normalized spatio-temporal coordinates and $(u_{t_i}(x_i, y_i), v_{t_i}(x_i, y_i))$ are the horizontal and vertical optical flow components. Only pixels belonging to

the silhouettes and having optical flow greater than zero are represented:

$$Z = \{z_i | (x_i, y_i) \in s_{t_i}, \|(u_{t_i}(x_i, y_i), v_{t_i}(x_i, y_i))\|_2 > 0\}. \quad (2)$$

We label the observations in $Z$ according to their normalized spatial origin - each observation $z_i$ is labeled with

$$l_i = \begin{cases} 1, & x'_i \leq \frac{1}{2}, y'_i \leq \frac{1}{2} \\ 2, & x'_i > \frac{1}{2}, y'_i \leq \frac{1}{2} \\ 3, & x'_i \leq \frac{1}{2}, y'_i > \frac{1}{2} \\ 4, & x'_i > \frac{1}{2}, y'_i > \frac{1}{2} \end{cases}, \quad (3)$$

differentiating between observations generated by the four parts of the silhouette (see Figure 2(d)).

### *2.1. Using Covariance Descriptor*

The purpose of region covariance descriptor, first introduced in [17], was for object recognition in still images. Since then, the covariance descriptor has proved useful also for representing spatio-temporal volumes [19, 21]. We employ the covariance descriptor for representing shape and motion statistics of a spatio-temporal buffer of video frames.

In the case of still image, each pixel is represented by a point in feature space. Possible features are the spatial coordinates of the pixel, its color and gradients, just to name a few. An image region consisting of $m$ pixels is described using the covariance matrix of their feature representations $Z = \{z_i\}_{i=0}^{m}$:

$$C_Z = \frac{1}{m-1} \sum_{i=1}^{m} (z_i - \mu)(z_i - \mu)^T, \quad (4)$$

where $n$ is the number of features and $\mu$ is the mean value of $z_i$s. An image is described by a set of covariance matrices, of size $n \times n$, corresponding to image the regions. Covariance matrices are compared using a metric for covariance matrices [25]:

$$d(C_1, C_2) = \sqrt{\sum_{i=1}^{n} \ln^2 \lambda_i(C_1, C_2)}, \quad (5)$$

where $\lambda_i(C_1, C_2)$ are the generalized eigenvalues of $C_1$ and $C_2$, both of size $n \times n$, computed from $\lambda_i C_1 u_i = C_2 u_i$, where $u_i \neq 0$ are the generalized eigenvectors.

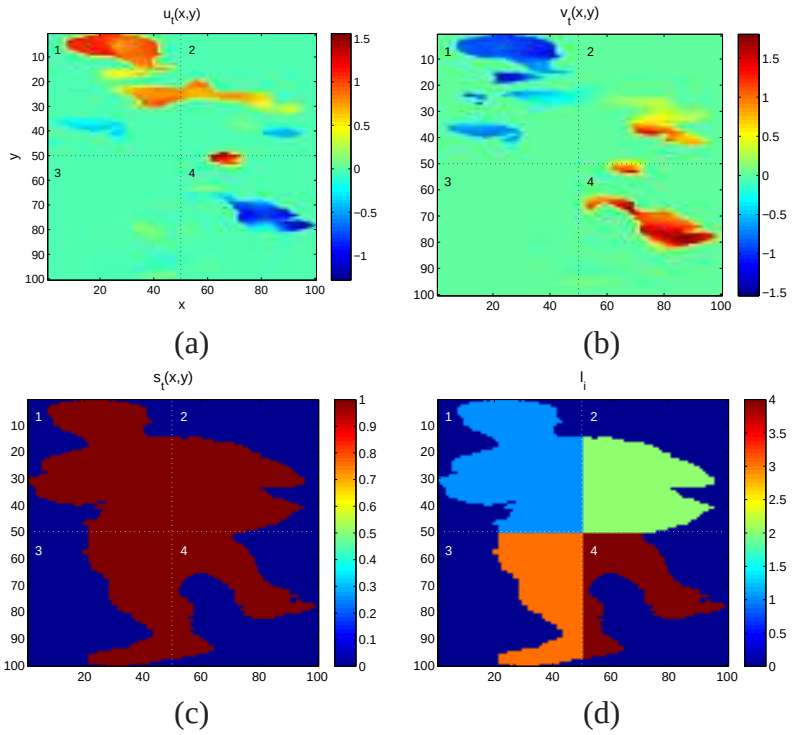



Figure 2: Horizontal and vertical optical flow heat maps in normalized coordinates, divided into four spatial regions using the silhouette $s_t(x, y)$. Each region is described using a covariance descriptors. (a) - $u_t(x, y)$ - horizontal optical flow component. (b) - $v_t(x, y)$ - vertical optical flow component. (c) - $s_t(x, y)$ - binary silhouette. (d) - $l_i$ - four labeled silhouette parts.

Intuitively, the metric for covariance matrices measures the extent to which the matrix $C_1^{-1}C_2$ deviates from the identity. For a pair of equal matrices $C_1$ and $C_2$, $d(C_1, C_2) = 0$ since the logarithms of all eigenvalues of $I$ are zero.

Our action descriptor is based on the region covariance descriptor extended to the spatio-temporal domain. Let $Z^k$ denote a set of observation in $Z$ generated by part $k$, $Z^k = \{z_i | z_i \in Z, l_i = k\}$. Similarly to [17] we represent the action with covariance matrices computed on observations generated by five overlapping regions:

$$A(Z) = \{C_Z, C_{Z^1 \cup Z^2}, C_{Z^3 \cup Z^4}, C_{Z^1 \cup Z^3}, C_{Z^2 \cup Z^4}\}. \quad (6)$$

The proposed representation encodes the correlation of measurements extracted from different regions.

*2.2. Riemannian Distance Metrics*

The group of covariance (symmetric positive definite) matrices may be represented as a connected Riemannian manifold $\mathcal{M}$. In such case, the distances between a pair of points $C_1$ and $C_2$ on the manifold is defined as a geodesic distance between these points, and is computed using the Affine-Invariant distance metric [26]:

$$d_{AI}(C_1, C_2) = \left\| \log\left(C_1^{-\frac{1}{2}} C_2 C_1^{-\frac{1}{2}}\right) \right\|_F = \sqrt{trace\left(\log^2\left(C_1^{-\frac{1}{2}} C_2 C_1^{-\frac{1}{2}}\right)\right)}, \quad (7)$$

where $\|\cdot\|_F$ is the matrix Frobenius norm and $log(\cdot)$ is the matrix logarithm operator. Let $C = U\Lambda U^T$ denote the eigenvalue decomposition of $C \in \mathcal{M}$, then the matrix logarithm operator maps $C$ to a point on a tangent (Euclidean) space, and is defined as:

$$log(C) = Ulog(\Lambda)U^T, \quad (8)$$

where $log(\Lambda)$ is a diagonal matrix with the logarithms of the eigenvalues of $C$. It turns out that the metric in Eq. 5, proposed by [25], is actually equivalent to the one from Eq. 7, proposed by [26].

One of the drawbacks of $d_{AI}$ is its relatively high computational burden, especially for high dimensional feature spaces. To overcome this issue, the Log-Euclidean metric was proposed in [27]:

$$d_{LE}(C_1, C_2) = \|\log(C_1) - \log(C_2)\|_F . \quad (9)$$

This metric maps both covariance matrices from the Riemannian manifold $\mathcal{M}$ to the tangent Euclidean space $T_I$ at point $I$ on the manifold using the logarithm operator. Besides the difference in several theoretical aspects (see [26, 27] for details), the $d_{LE}$ and the $d_{AI}$ metrics differ in some practical aspects as well. First, $d_{LE}$ is typically faster since it does not compute generalized eigenvalues as $d_{AI}$ does. Second, mapping the covariance descriptors to the Euclidean space opens possibilities to employ various general machine learning techniques, *e.g.*, [28, 29]. We have evaluated both metrics in terms of computational efficiency and recognition accuracy while comparing the proposed descriptors. Our experiments showed that $d_{AI}$ is roughly twice slower comparing to $d_{LE}$, but the speed comes with an expense of slight degradation in the recognition accuracy. Therefore, despite the advantages and considering the low dimensionality of our descriptor, we decided that the modest gain in the computational efficiency does not justify the preference of $d_{LE}$ over $d_{AI}$. We note that similar results were reported in [29] regarding the speed versus accuracy for these two metrics.

### 2.3. Comparing Actions

We suggest three possible distance measures for comparison between action descriptors. The first measure compares only the $C_Z$ of both actions, thus disregarding the observations' spatial origin while the second measure compares pairs of covariance matrices generated using observations of similar spatial origin. Let $A_1 = \{C_i^1\}_{i=1}^5$ and $A_2 = \{C_i^2\}_{i=1}^5$ denote two action descriptors as defined in Eq.6. The two measures are defined as:

$$D_1(A_1, A_2) = d(C_1^1, C_1^2), \tag{10}$$

and:

$$D_2(A_1, A_2) = \sum_{i=2}^{5} d(C_i^1, C_i^2), \tag{11}$$

where $d(\cdot, \cdot)$ is the metric for covariance matrices (Eq.5). The third distance measure uses the whole set of covariance matrices of $A$. It combines $D_1$ and $D_2$ in a way which gives more robustness to partial occlusions and to silhouette imperfections. Similarly to [17], we define the third measure as:

$$D(A_1, A_2) = D_1(A_1, A_2) + D_2(A_1, A_2) - \max_j d(C_j^1, C_j^2). \tag{12}$$

The purpose of the subtracted element is to compensate for an accidentally large distance computed between a certain pair of covariance descriptors of $A_1$ and $A_2$ which may be due to partial occlusion or a missing silhouette part in one of the descriptors. In our experiments we have found that the distance measure defined in Eq.12 gives the best performance in action categorization tasks. However, for online action recognition task, $D_1$ is preferable because of its lower computational cost.

### 2.4. Online Scenario

An important property of our approach is its extendability to the online scenario of action recognition. In addition to its computational efficiency this fact enables it to be successfully applied in real time interactive applications such as gaming and natural user interface. The challenge in online action recognition scenario is that the decision regarding whether a specific action is taking place at the moment has to be made based on an incomplete information available up until that moment. Making the decision as soon as possible is crucial because it immediately affects the system's behavior.

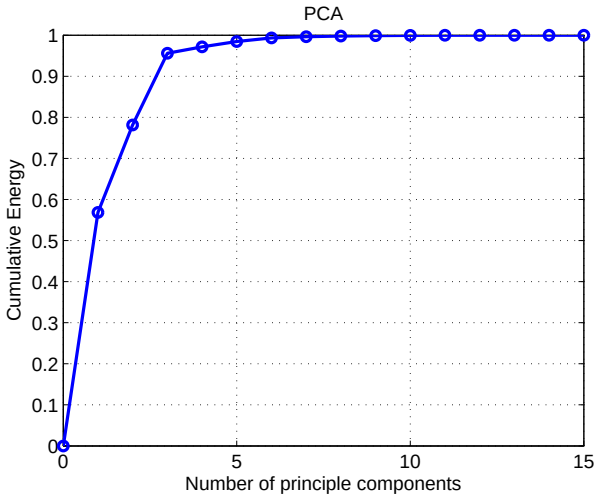



Figure 3: Cumulative energy as a function of a number of major principle components.

A useful property of the *CovAct* descriptor is that its dimension does not depend on the dimensions of the underlying spatio-temporal volume. That is to say, two descriptors, one extracted from a complete spatio-temporal volume describing an action and the second one extracted from only the first temporal half of this volume may be naturally compared. This is in contrast with methods reported in [13, 14] where the volumes have to be normalized to a fixed temporal size. This observation allows us to apply the nearest neighbor classifier on descriptors extracted from different spatio-temporal volumes. To see the covariance descriptors' behavior, we have used spatio-temporal volumes of gradually increasing temporal length (incremented one frame at a time), to extract a set of covariance matrices belonging to five different action types. The actions we used were punch, kick, bend forward, bend left and bend right and were performed by a single actor while playing an interactive video game. All covariance matrices in the set have $5 + (5 \times 4)/2 = 15$ unique values due to symmetry. We have applied principal component analysis (PCA) on these 15D vectors. Figure 3 depicts the cumulative energy as a function of the number of principal components. It can be seen from the graph that two major principal components contain around 80% of the overall energy. Figure 4 shows clusters of trajectories for each one of the five actions plotted in 2D using the two major principal components. Each trajectory represents the descriptor's transformation with the accumulation of frames in the underly-

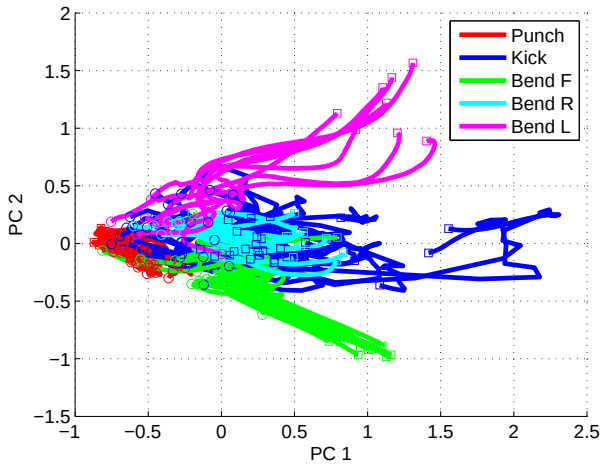



Figure 4: Actions trajectories illustration in 2D where the axes are the two major principle components. '$\circ$' marks the beginning of a trajectory and '$\square$' marks its end. Note that for each action type the descriptors tend to cluster, and diverge in different directions as actions develop.

ing spatio-temporal volume. Each trajectory begins with 'o' and ends with "$\square$" corresponding to action's complete covariance descriptor $C_Z$. It can be noticed from the 2D illustration that the trajectories belonging to each action class tend to cluster together, and diverge in different directions as more frames are used. Since adding more principle components will intensify this divergence we conclude that in the original 15D space the trajectories are clustered as well.

Another interesting observations is that the trajectories' intermediate points corresponding to descriptors extracted from incomplete information, at some point in time, tend to be closer to the trajectories' end points belonging to the same class than to those belonging to different classes. To support this intuition, we used Eq. 10 to compute the distances between covariance descriptors extracted from complete and incomplete spatio-temporal volumes. Let $C_i^{X\%}$ denote a covariance descriptor computed using the $X\%$ of the $i$'s spatio-temporal volume, and let $a(C_i)$ denote its corresponding action class. We define the distribution of distances between descriptors belonging to the same class, extracted from volumes of $X$'s degree of completeness and those extracted from the entire volumes (100%-complete), as $p^{X\%}_{same}(D_1(C_i^{X\%}, C_j^{100\%})|i \neq j, a(C_i) = a(C_j))$. Similarly, we define the distribution of distances between descriptors belonging to different classes as $p^{X\%}_{diff}(D_1(C_i^{X\%}, C_j^{100\%})|a(C_i) \neq a(C_j))$. Figure 5 shows the distri-

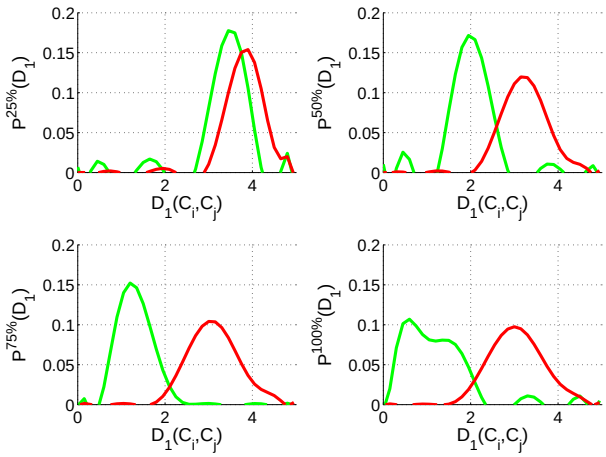



Figure 5: Distance distributions $p^{X\%}_{same}$ and $p^{X\%}_{diff}$ between the $X\%$-complete and the $100\%$-complete descriptors for the same (green) and different (red) actions, respectively. Note that even for the $50\%$-complete descriptors the modes look quite well separated.

butions $p^{X\%}_{same}$ and $p^{X\%}_{diff}$ in green and red respectively, for $X = 25, 50, 75, 100$. As expected the modes separation increases with the increased information completeness. Note that while it may be difficult to correctly predict the action based on 25% of the information, it is much easier based on 50%. Thus, the distance between a descriptor extracted online, based on an incomplete spatio-temporal volume, to those extracted during training phase, based on complete volumes, may serve as a predictor on what action is currently being performed. Thanks to this, we avoid the explicit modeling of actions' fragments, significantly reducing the number of exemplar descriptors required for the nearest neighbor classifier. On the other hand, if necessary, we may seamlessly augment the exemplars set with descriptors extracted from incomplete spatio-temporal volumes, at the price of runtime performance.

### *2.4.1. Online Action Recognition Algorithm*

Based on the above derivations we propose an algorithm (summarized in Algorithm 1) for online action recognition. The algorithm uses a temporal sliding window of $\mathcal{L}_w$ frames to buffer measurements for covariance descriptor computation. For each new frame the data buffer is updated, a new covariance matrix $\mathcal{C}_t$ is calculated and compared with exemplar matrices collected during training. The distance to the nearest neighbor is used to detect whether or not the action of

the nearest neighbor's class took place. Rather than applying a threshold on the absolute distance to the nearest neighbor we have found it useful to apply it on the ratio between the nearest and the farthest neighbor. Let $\{(C^i, a_i)\}_{i=1}^N$ denote the descriptors stored in the database together with their class labels taking values from $\mathcal{A} = \{a_1, ..., a_{N_a}\}$. Let $d_t^{max}$ denote the $\mathcal{C}_t$'s distance to the farthest neighbor - $d_t^{max} = \max_{i=1...N} d(\mathcal{C}_t, C_i)$. Let $\gamma_a$ denote the threshold on the nearest-to-farthest distance ratio for each class $a \in \mathcal{A}$. Thus, a covariance descriptor $\mathcal{C}_t$ is classified as belonging to class $a$ if $C^* = \operatorname{argmin}_{i=1...N} d(\mathcal{C}_t, C_i)$ belongs to class $a$ and:

$$d(\mathcal{C}_t, C^*) \leq th_t^a, \tag{13}$$

where $th_t^a = d_t^{max}\gamma_a$ and $d(\cdot, \cdot)$ is the covariance distance defined in Eq. 5.

**Algorithm 1** Online Action Recognition (CovActOL)

**Input:** Video frames $V = \{I_t\}_{t=0}^M$, maximum buffer size $\mathcal{L}_w$,
action descriptors database $DB = \{(C^i, a_i)\}_{i=1}^N$ ($a_i \in \mathcal{A}$),
nearest-to-farthest distance ratio threshold for each action class $\{\gamma_a | a \in \mathcal{A}\}$

1. Initialize $\mathcal{Z} = \emptyset$.
2. For t=1:M
2.1. Extract silhouette mask $s_t$.
2.2. Compute LK optical flow $(u_t, v_t)$ using frames $I_t, I_{t-1}$.
2.3. Build $Z_t$ as described in Eq. 2.
2.4. $\mathcal{Z} = \mathcal{Z} \cup Z_t$.
2.5. If $t \geq \mathcal{L}_w$
2.5.1. $\mathcal{Z} = \mathcal{Z} \setminus Z_{t-\mathcal{L}_w}$.
2.5. Compute covariance matrix $\mathcal{C}_t$ of $\mathcal{Z}$.
2.6. For i=1:N
2.6.1. Compute $d(\mathcal{C}_t, C^i)$ using Eq. 5.
2.7. $i^* = \arg\min_i d(\mathcal{C}_t, C^i)$.
2.8. If $d(\mathcal{C}_t, C^{i^*}) < th_{a_{i^*}}^t$, Output "Action $a_{i^*}$".
2.9. Else, Output "No Action".

### *2.5. Runtime Optimization*

One of the deficiencies of the *CovActOL* algorithm as described in Algorithm 1, is that its runtime depends on the buffer size $\mathcal{L}_w$ and on the database size $N$.

This dependency is because for each frame we need to recompute the covariance descriptor and to compute its distance to each one of the descriptors in the database. In this section we outline two modifications to the *CovActOL* algorithm dealing with each one of these deficiencies independently, with no impact caused to the algorithm's correctness. In the first modification we show that rather than recomputing the descriptor for each frame we can use a simple and efficient update rule. In the second modification we use the fact that the descriptors computed in the consecutive frames are not likely to be very different one from each other, and thus the distances between them and the descriptors in the database are not likely to change much as well. Using this observation and the properties of the metric for covariance matrices, Eq. 5, we derive a criterion which allows us to skip over the distance computations which are certainly not going to affect the classifier's final decision.

*2.5.1. Efficient Incremental Computation of the Covariance Descriptor*

In the online scenario the temporal sliding window is of a fixed size, and so for each new frame the data buffer has to be updated, namely observations extracted from the new frame should be appended while observations of the oldest frame should be dropped. However, the requirement of recomputing the covariance matrix at each iteration induces a significant computational burden, especially if a large number of pixels are extracted from each frame. To overcome this issue we propose a simple and efficient update rule for incrementally computing the covariance descriptor instead of recomputing it entirely after each frame.

Let $\mathcal{D}_{t-1} = \cup_{i=1}^{L_w} Z_i$ denote a set of observations, currently in the buffer, extracted from the $\mathcal{L}_w$ frames preceding frame $t-1$, where $\mathcal{L}_w$ is the buffer size and $Z_i = \left\{x_j^i\right\}_{j=1}^{N_i}$ denotes a set of observations extracted from frame $t-1-i$. Let $N_i$ and $\mathcal{N}_{t-1}$ denote the number of samples in $Z_i$ and $\mathcal{D}_{t-1}$ respectively. Let $\mu_i$ and $C_i$ denote the mean and the covariance of $Z_i$, and $\mathcal{M}_{t-1}$ and $\mathcal{C}_{t-1}$ denote the mean and the covariance of $\mathcal{D}_{t-1}$. As a new frame, $t$, arrives, its observations are appended to the buffer: $\mathcal{D}_t = \mathcal{D}_{t-1} \cup Z_t$. If the buffer is full[1], the least recent frame's observations are dropped: $\mathcal{D}_t = \mathcal{D}_t \setminus Z_1$. Thus, the number of observations in $\mathcal{D}_t$ is $\mathcal{N}_t = \mathcal{N}_{t-1} - N_1 + N_{t+1}$, where $N_1 = 0$ when the buffer is not full. Our goal, therefore, is to efficiently compute the new mean $\mathcal{M}_t$ and the covariance $\mathcal{C}_t$ of samples in $\mathcal{D}_t$ using $\mathcal{M}_{t-1}$ and $\mathcal{C}_{t-1}$, without explicitly recomputing them from the data $\mathcal{D}_t$. In Section Appendix A we derive these efficient update rules for $\mathcal{M}_t$

[1]By full buffer we mean that it contains observations extracted from $\mathcal{L}_w$ last frames.

and $\mathcal{C}_t$, which are:

$$\mathcal{M}_t = \frac{1}{\mathcal{N}_t}\left(\mathcal{N}_{t-1}\mathcal{M}_{t-1} - N_1\mu_1 + N_t\mu_t\right), \tag{14}$$

and

$$\mathcal{C}_t = \frac{1}{\mathcal{N}_t - 1}\left(\left(\mathcal{N}_{t-1} - 1\right)\mathcal{C}_{t-1} + \Delta\mathcal{C}_t\right), \tag{15}$$

where

$$\Delta\mathcal{C}_t = -(N_1-1)C_1+(N_t-1)C_t+\mathcal{N}_{t-1}\mathcal{M}_{t-1}\mathcal{M}_{t-1}^T-N_1\mu_1\mu_1^T+N_t\mu_t\mu_t^T-\mathcal{N}_t\mathcal{M}_t\mathcal{M}_t^T. \tag{16}$$

Thus the update takes $O(N_t^2)$ operations instead of the $O(\mathcal{N}_t^2)$ operations, which would be the case had we explicitly recomputed the covariance from the data for each frame. For example, if $\mathcal{L}_w$=30, then we improve the descriptor computation speed by a factor of $\mathcal{L}_w^2 = 900$. We note that a similar approach was used in [30] to improve the performance of real time tracking using the covariance descriptor. The authors used an efficient update rule to build an adaptive appearance model of the tracked target.

Using $m$ recursive expansions of expression Eq. 15, we get $\mathcal{C}_t$ as a function of $\mathcal{M}_{t-m}$ and $\mathcal{C}_{t-m}$, and $M_{t-i}$ and $C_{t-i}$ computed from $\{Z_{t-i}\}_{i=0}^{m-1}$. (see Section Appendix A for details):

$$\mathcal{C}_t = \frac{1}{\mathcal{N}_t - 1}\left(\left(\mathcal{N}_{t-m} - 1\right)\mathcal{C}_{t-m} + \sum_{i=0}^{m-1}\Delta\mathcal{C}_{t-i}\right). \tag{17}$$

*2.5.2. On Demand Nearest Neighbor Classification*

Let $d(\mathcal{C}_{t-m}, C')$ denote the distance between the covariance descriptor $\mathcal{C}_{t-m}$, computed in $(t-m)$'s iteration of the *CovActOL* algorithm ($m < t$), and some covariance descriptor in the database, $C'$. Let us also assume that $C'$ belongs to class $a$ and $d(\mathcal{C}_{t-m}, C') - th_{t-m}^a \geq 0$, where $th_{t-m}^a$ is as defined in Section 2.4.1. Thus, action $a$ was certainly not detected based on the $\mathcal{C}_{t-m}$'s proximity to $C'$ in iteration $t-m$. We would like to define a condition which, once held, guarantees that $d(\mathcal{C}_t, C') - th_t^a \geq 0$, meaning that action $a$ is not detected based on the $\mathcal{C}_t$'s proximity to $C'$ in iteration $t$, without explicitly computing the distance. Note that the thresholds $th_{t-m}^a$ and $th_t^a$ depend on the distances to the farthest neighbor in iterations $t-m$ and $t$ respectively. However, since the farthest neighbor's distance in iteration $t$ could not shrink by more than $d(\mathcal{C}_t, \mathcal{C}_{t-m})$ relative to the farthest neighbor's distance in iteration $t-m$, we can say that $d_t^{max} \geq d_{t-m}^{max} - d(\mathcal{C}_t, \mathcal{C}_{t-m})$, and thus:

$$th_t^a = d_t^{max}\gamma_a \geq \gamma_a(d_{t-m}^{max} - d(\mathcal{C}_t, \mathcal{C}_{t-m})) = th_{t-m}^a - \gamma_a d(\mathcal{C}_t, \mathcal{C}_{t-m}). \tag{18}$$

The triangle inequality holds for the covariance distance and therefore:

$$d(\mathcal{C}_t, C') \geq d(\mathcal{C}_{t-m}, C') - d(\mathcal{C}_t, \mathcal{C}_{t-m}). \tag{19}$$

Using the inequalities 18 and 19, we obtain the following inequality:

$$\begin{aligned} d(\mathcal{C}_t, C') - th_t^a &\geq d(\mathcal{C}_t, C') - th_{t-m}^a + \gamma_a d(\mathcal{C}_t, \mathcal{C}_{t-m}) \\ &\geq d(\mathcal{C}_{t-m}, C') - d(\mathcal{C}_t, \mathcal{C}_{t-m}) - th_{t-m}^a + \gamma_a d(\mathcal{C}_t, \mathcal{C}_{t-m}) \\ &\geq d(\mathcal{C}_{t-m}, C') - th_{t-m}^a - (1-\gamma_a) d(\mathcal{C}_t, \mathcal{C}_{t-m}). \end{aligned} \tag{20}$$

This lower bound allows us to derive a criterion to whether or not we should recompute the distance $d(\mathcal{C}_t, C')$ in iteration $t$ for some $C'$ in the database, or we can say for sure that it is above the threshold. Also, we define $m_{max}$ indicating the maximum number of frames allowed to skip the distance computation. Thus, in iteration $t$ we require a computations of $d(\mathcal{C}_t, C')$ only if

$$m > m_{max} \; Or \; d(\mathcal{C}_{t-m}, C') - th_{t-m}^a - (1-\gamma_a) d(\mathcal{C}_t, \mathcal{C}_{t-m}) < 0, \tag{21}$$

otherwise we conclude that $d(\mathcal{C}_t, C') > th_t^a$ without the actual computation of $d(\mathcal{C}_t, C')$. In our experiments we have found that $m_{max} = \mathcal{L}_w/2$, where $\mathcal{L}_w$ is the sliding window length, provides a good choice in terms of performance.

In iteration $t$ we evaluate the criterion from Eq. 21 for each one of the $N$ descriptors in the database. The criterion computation involves at most $m_{max}$ covariance distance computations to compare $\mathcal{C}_t$ with each one of the $\mathcal{C}_{t-m},\ m = 1, \ldots, m_{max}$. Also once in a while, when the criterion does not hold, the distance computation comparing $\mathcal{C}_t$ with $C'$, some descriptor from the database, should be performed. Thus, the overall number of covariance distance computations per frame is reduced from $N$, as in the *CovActOL* algorithm, to $m_{max} + pN$, where $p$ is the probability that the condition in Eq. 21 does not hold. According to our experiments, $p \approx 0.1$ when $\mathcal{L}_w = 30$ is used. This is much better considering that typically $N >> m_{max}$. Experimental validation showed that about 70%-90% of the covariance distance computations are saved when using the criterion.

### *2.6. Fast Online Action Recognition*

We have modified the *CovActOL* algorithm based on the improvements described in sections 2.5.1 and 2.5.2. The resulting algorithm, *Fast-CovActOL*, is summarized in Algorithm 2. The algorithm uses a temporal sliding window of $\mathcal{L}_w$ frames to extract the *CovAct* descriptor. For each new frame $t$ the *CovAct* descriptor $\mathcal{C}_t$ is computed using the updated rule in Eq. 15, and is compared with the exemplar descriptors meeting the criterion (inequality 21), using the covariance distance Eq. 5. Action $a$ is detected if $C_t$'s nearest neighbor is of class $a$, and its

distance is smaller than the threshold, $th_t^a$. We present an experimental validation of the algorithm in Section 3.1.

**Algorithm 2** Fast Online Action Recognition (Fast-CovActOL)

**Input:** Video frames $V = \{I_t\}_{t=0}^{M}$, maximum buffer size $\mathcal{L}_w$,
action descriptors database $DB = \{(C^i, a_i)\}_{i=1}^{N}$ ($a_i \in \mathcal{A}$),
nearest-to-farthest distance ratio threshold for each action class $\{\gamma_a | a \in \mathcal{A}\}$

1. Initialize $m_i = 0, \forall i = 1, \ldots, N$
2. For t=1:M
2.1. Extract silhouette mask $s_t$.
2.2. Compute LK optical flow $(u_t, v_t)$ using frames $I_t, I_{t-1}$.
2.3. Build $Z$ as described in Eq. 2 with $t'_i = t$.
2.4. Compute the mean $\mu_t$ and the covariance matrix $C_t$ of $Z$ ($N_t = |Z|$).
2.5. Update $\mathcal{C}_t$ using Eq 15.
2.6. For i=1:N
2.6.1. If $t = 1$ Or $m_i > m_{max}$ Or $d(\mathcal{C}_{t-m_i}, C^i) - th^a_{t-m_i} - (1 - \gamma_{a_i}) d(\mathcal{C}_t, \mathcal{C}_{t-m_i}) < 0$
2.6.1.1. Compute $d(C_t, C^i)$ using Eq. 5.
2.6.1.2. $m_i = 0$
2.6.2. Else
2.6.2.1. $d(C_t, C^i) = d(C_{t-m_i}, C^i)$
2.6.2.2. $m_i = m_i + 1$
2.7. $i^* = \text{argmin}_i \, d(C_t, C^i)$.
2.8. If $d(C_t, C^{i^*}) < th^t_{a_{i^*}}$, Output "Action $a_{i^*}$".
2.9. Else, Output "No Action".

## 3. Experimental Results

Our experimental validation includes two parts. In the first part we validate our online action recognition approach on privately collected video sequences. In the second part we present experimental validation of our *CovAct* descriptor in gesture recognition on two publicly available datasets - the Cambridge-Gesture [13] and ChaLearn [16] datasets. In the third part we evaluate the *CovAct* descriptor in action recognition scenario on a subset of the challenging UTF-101 [31] dataset.

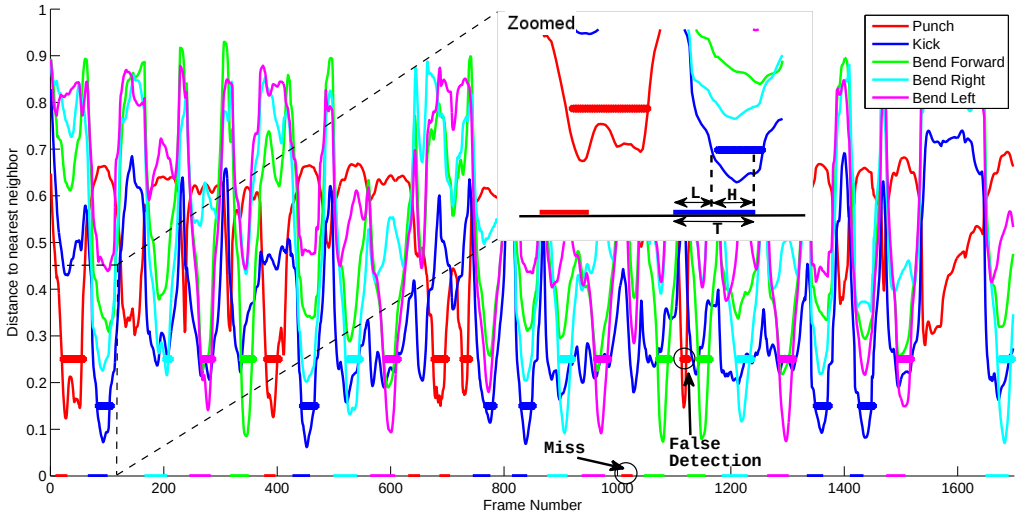



Figure 6: The output of the *CovActOL* algorithm for the first 1,700 frames of the test sequence. The colored horizontal intervals on and above the X axis indicate the true and the detected actions, respectively. The magnified region highlights the performance measures used to evaluate the algorithm performance.

### *3.1. Online Action Recognition*

We have evaluated our online action recognition approach described in Section 2.4 on a privately collected dataset of a player performing five game actions— "Punch", "Kick", "Bend Forward", "Bend left" and "Bend right" — while playing an interactive video game. We have used background subtraction to extract the player's silhouette in each frame. In the training phase we have built the action descriptors database, $DB = \{(C^i, a_i)\}_{i=1}^{N}$, and the set of thresholds on the nearest-to-farthest distance ratio for each action, $\{\gamma_a | a \in \{1, \ldots, 5\}\}$. The database of $N = 88$ descriptors was extracted from a manually labeled video sequence of the player performing the actions. The set of thresholds was obtained using a validation video sequence with 10 action instances (two for each action class). In the test phase we have applied the *CovActOL* and the *Fast-CovActOL* algorithms on a test video sequence containing 3,821 frames. We have used the descriptors database and the thresholds set obtained during the learning phase and $\mathcal{L}_w = 30$ as the input parameters to the algorithms. We remark that all of the actions in our experiments were performed by the same actor.

Figure 6 shows the distances of descriptors, extracted from frames in the slid-

ing window, to the nearest neighbor of each action class as a function of the frame number. For each frame the distances were normalized with respect to the maximal distance, thus the Y axis is in the [0,1] range. The color-coded intervals overlaid on top of the X axis are the ground truth indicating the frame ranges where the actions (each action is marked with a specific color) actually took place. We call these intervals action instances. The horizontal color-coded intervals appearing on different levels above X axis mark those frames which were classified as belonging to a certain action. We call these intervals action detections. The interval's vertical level indicates the threshold ($\gamma_a$) which was used to detect the corresponding action class. For example, for detecting a kick action a threshold of 0.15 was used. Any frame which is not color-coded belongs to an additional class we call "No Action".

We use three types of measures to evaluate the performance of our algorithm. First we define the *miss rate* as a portion of missed instances of a specific class out of the total number of instances belonging to this class. An action instance is considered as missed if its intersection with any action detection of the same class is empty. See an example of a missed punch in Figure 6. For all action instances which are not missed we define the *latency* and the *hold* measures. The latency measure is defined as the portion of the action instance duration which was required to detect that instance for the first time. The hold measure is defined as a correctly classified portion of the instance duration from the moment it was detected for the first time. The magnified section of Figure 6 focuses on a sequence of two actions. For the blue (kick) action, for example, the latency equals to $\frac{L}{T}$ and the hold equals to $\frac{H}{T-L}$. Table 1 summarizes the per class average measures obtained for the test video sequence. Note that the punch action is missed in 33% of the time, other actions are never missed. Note that the latency measure indicates the portion of the original spatio-temporal volume used for correct classification. Besides punch, which is typically detected based on only one third of its original spatio-temporal volume, all other actions are detected based on two thirds of theirs. Also note that the average hold measure is significantly lower for the punch action than for other actions. The reason for this is the descriptor's contamination with irrelevant frames caused by the relatively short punch action duration (18 frames on average) compared to the sliding window (30 frames).

We note that our performance measures were used to determine the set of thresholds on the nearest-to-farthest distance ratio for each action ($\gamma_a$). In particular, we have performed a grid search to find the set optimizing a value function combining the miss rate, the latency and the hold measures, computed over the validation video sequence. In other types of applications, for example, in action

| Action Names | Actions Num. | Miss Rate(%) | Avg. Latency(%) | Avg. Hold(%) |
|---|---|---|---|---|
| No Action | 61 | 0 | 20 | 97 |
| Punch | 21 | 33 | 30 | 75 |
| Kick | 13 | 0 | 66 | 100 |
| Bend Fwd. | 7 | 0 | 69 | 100 |
| Bend Right | 9 | 0 | 70 | 100 |
| Bend Left | 10 | 0 | 67 | 100 |
| Total | 121 | 6 | 37 | 94 |

Table 1: Performance measures evaluated on the test video sequence. Note that the miss rate of the "Punch" action is 33% while all other actions are never missed, and therefore the only action responsible for the 6% total miss rate is the "Punch" action.

detection where there is no requirement to detect the action's interval, but only its approximate point in time, another value function could be used.

The processing speed of our non-optimized Matlab implementation of the *CovActOL* and the *Fast-CovActOL* algorithms applied on the test video sequence was 13 and 16 fps respectively on a 1.70GHz Intel Core i5 machine. The processing speed of the *CovActOL* algorithm drops as the temporal window size and the database size increase. The *Fast-CovActOL* algorithm's speed, on the other hand, does not depend on the window size and is much less affected by an increase in the database size. We attach the test video with online action detections to the supplementary material. We also attach a video example of our algorithm's performance on gesture recognition task running in real time.

### *3.2. Gesture Recognition*

In this section we describe the experimental validation of our *CovAct* descriptor in a standard gesture recognition scenario. In our experiments we have used exactly the same descriptor for categorization of palm gestures (Cambridge dataset) and general gestures involving both body and hands (ChaLearn). Given a gallery set of action descriptors we have used a nearest neighbor, with distance measure defined by Eq. 12, to classify a query descriptor.

#### *3.2.1. Cambridge-Gesture database*

Cambridge-Gesture database [13] contains 900 video sequences of nine types of hand gestures performed by three individuals. The sequences are divided into

| Method | Set1 | Set2 | Set3 | Set4 | Total |
|---|---|---|---|---|---|
| pLSA [12] | 70 | 57 | 68 | 71 | $66\pm6.1$ |
| TCCA [13] | 81 | 81 | 78 | 86 | $82\pm3.5$ |
| RLPP [20] | 86 | 86 | 85 | 88 | $86\pm1.3$ |
| PM [14] | 89 | 86 | 89 | 87 | $88\pm2.1$ |
| CovAct | 89 | 86 | 83 | 87 | $86\pm2.3$ |

Table 2: Correct classification rates for Cambridge-Gesture database. Each column reports the performance on different set. The average performance is reported in the "Total" column.

five sets - one set for each one of five types of illumination. Each set contains twenty sequences (ten for each one of two individuals) for each one of nine gestures. The dataset does not contain hand masks, so we have evaluated our approach without using silhouette masks at all, taking the measurements from the entire frame. In our experiments we have followed the protocol described in [13]. Sequences from Set5 which were shot under standard illumination were used as the training set and sequences from Set1, Set2, Set3 and Set4 were used as testing sets. Since our approach does not employ learning, we have used the whole Set5 for training rather than splitting it by half into training and validation sets as done in [13]. Unlike in [13, 14] where the action spatio-temporal volume is resized to a fixed size we do not need to employ this step since our approach inherently copes with actions of different durations. We have used nearest neighbor with distance metric from Eq. 12 for classification. Table 2 shows the recognition rates of our *CovAct* approach, the pLSA [12] method, TCCA [13] method, the Product Manifold [14] method and the RLPP [20] method. Despite its simplicity our method has a much better performance than the TCCA and the pLSA methods and the PM method slightly outperforms ours. Figure 7(a) shows the recognition confusion matrices for *CovAct*. Note that two typical classification errors occurred - the "Flat Right"(FR) gesture was confused with the "Spread Right"(SR) gesture and the "V Left"(VL) gesture was confused with the "Flat Left"(FL) gesture. Looking at Figure 7(b), depicting the recognition confusion matrix of the PM method, note that for about half of the gesture types our recognition rates are higher than those of PM.

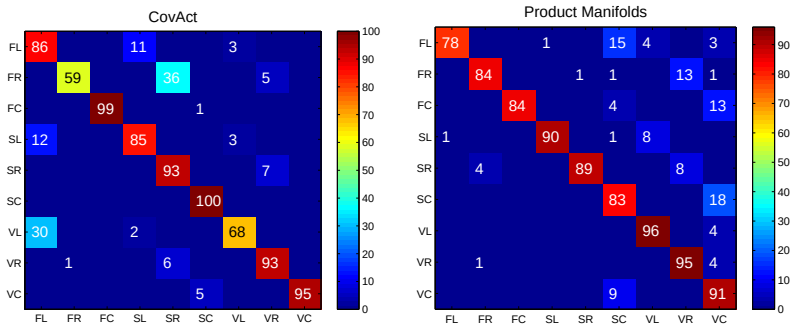



Figure 7: Confusion matrices for Cambridge-Gesture database, see text. (a) - CovAct method. (b) - Product Manifolds (PM) [14] method.

### *3.2.2. ChaLearn One-Shot-Learning Dataset*

ChaLearn dataset [2] is a large set of 50,000 gestures of various types, recorded using the Microsoft Kinect camera. The dataset contains short RGB video clips featuring a single actor performing a sequence of gestures. Each RGB video clip is coupled with a corresponding depth video captured using Kinect (see Figure 8). One-Shot-Learning gesture challenge [16], was to recognize a specific gesture from a vocabulary of 8-15 gestures, based on a single example for each gesture from the vocabulary. This is a challenging task since extensive learning procedures providing good generalization capabilities may not be applied simply due to the lack of training data. We have evaluated our descriptor on the twenty provided development batches, each including 47 video clips, following the methodology described in [16]. Depth frames were used to extract the actors' silhouettes. Table 3 shows error rates in terms of an average Levenshtein distance [32], also known as the edit distance. We compare our method with the baseline method reported in [16] and the method reported in [15] which is an extension of PM [14] specifically tuned for this challenge. Our results are superior to the baseline in all batches. Results marked in green are better or comparable to those obtained in [15], and those marked in red are worse. Note that despite the relative richness and complexity of the PM descriptor compared to ours, we report superior results for some of the batches, *e.g.*, devel01, devel05. That being said, our approach did not perform well on several batches. After inspecting the video clips

[2] ChaLearn Gesture Dataset (CGD2011), ChaLearn, California, 2011

(a) (b)

Figure 8: Example of a frame from one of the ChaLearn One-Shot-Learning dataset video clips. (a) - RGB frame of video clip where a subject performs a gesture. (b) - Depth frame corresponding to the RGB frame (darker color means closer to the camera).

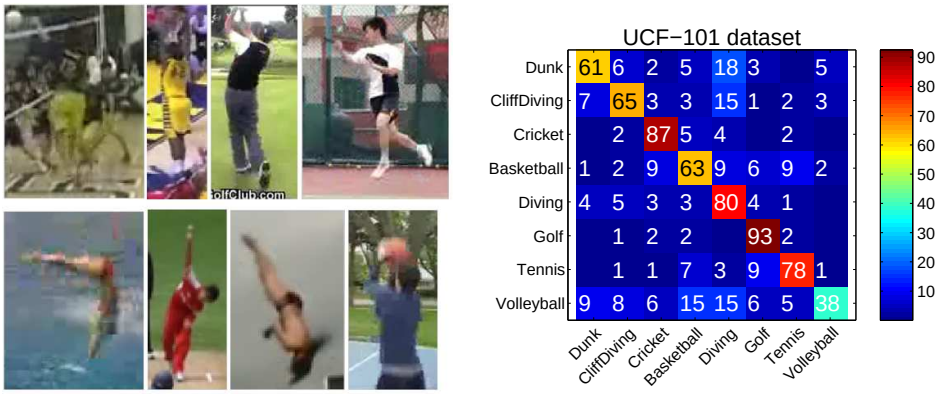
we have found that the gestures in batches where we did well are based primarily on limbs' motion and those batches where we did not do well are based on subtle palm gestures overridden by more dominant, but not informative, limbs' motion.

### *3.3. UCF-101 Action Recognition Dataset*

The UCF-101 dataset [31] is a large collection of video clips belonging to 101 action classes, downloaded from YouTube. The clips were recorded in unconstrained environment and contain large variability in viewing angles and illumination conditions, partial occlusions, and a significant amount of background clutter. Video clips belonging to 24 sport action classes come with bounding box annotations. These sports action classes may be roughly divided into two categories — continuous activities, *e.g.*, fencing, horse riding, ice dancing, and short actions, *e.g.*, a golf swing, a basketball dunk, a dive. In our experiments we have used the following set of 8 actions belonging to the second category — "Basketball Dunk", "Cliff Diving", "Cricket Bowling", "Basketball Shooting", "Diving", "Golf Swing", "Tennis Swing", "Volleyball Spiking" (see Figure 9(a)). In total we have extracted 1,209 action instances from the labeled sequences of the UCF-101 dataset. We note that in this experiment we did not use subjects' silhouettes to derive the *CovAct* descriptors, but rather used the whole bounding boxes. Figure 9(b) shows the confusion matrix of the 8 action classes we used. Note the relatively high confusion rates of the "Cliff Diving" vs. "Diving", and the "Basketball Shooting" vs. "Volleyball Spike" action pairs, which are due to the high inherent similarity of these actions. As expected, we can see that actions involving

| Batch | Base | PM [15] | CovAct |
|---|---|---|---|
| devel01 | 53.33 | 13.33 | 4.44 |
| devel02 | 68.89 | 35.56 | 28.40 |
| devel03 | 77.17 | 71.74 | 61.95 |
| devel04 | 52.22 | 10.00 | 11.11 |
| devel05 | 43.48 | 9.78 | 4.34 |
| devel06 | 66.67 | 37.78 | 34.44 |
| devel07 | 81.32 | 18.68 | 19.78 |
| devel08 | 58.43 | 8.99 | 26.96 |
| devel09 | 38.46 | 13.19 | 18.68 |
| devel10 | 75.82 | 50.55 | 65.93 |
| devel11 | 67.39 | 35.87 | 47.82 |
| devel12 | 52.81 | 22.47 | 30.33 |
| devel13 | 50.00 | 9.09 | 34.09 |
| devel14 | 73.91 | 28.26 | 39.13 |
| devel15 | 50.00 | 21.74 | 48.91 |
| devel16 | 57.47 | 31.03 | 45.97 |
| devel17 | 66.30 | 30.43 | 40.21 |
| devel18 | 70.00 | 40.00 | 55.55 |
| devel19 | 71.43 | 49.45 | 61.53 |
| devel20 | 70.33 | 35.16 | 31.03 |
| Average | 62.32 | 28.73 | 35.53 |

Table 3: Sum of Levenshtein distances divided by the total number of actions in each batch of the ChaLearn gestures dataset [16].




(a) (b)

Figure 9: (a) - Examples of bounding boxes extracted from video sequences for each one of the 8 classes from the UCF-101 dataset. (b) - Confusion matrix between 8 actions from the UCF-101 dataset.

less clutter, *i.e.*, "Cricket Bowling", "Diving", "Golf Swing", "Tennis Swing" are recognized with higher rates than other action involving larger amounts of clutter.

## 4. Discussion and Conclusions

We have presented a method for action recognition for the online scenario. The method recognizes actions on the fly by applying a nearest neighbor classifier on descriptors extracted from an incomplete spatio-temporal volume available at the moment. The proposed descriptor encodes the statistics of temporal shape and motion changes in a low dimensional space using the covariance descriptor. The method was successfully validated on a privately collected dataset, using quality measures, e.g. latency, especially designed for the online scenario. The descriptor was evaluated on standard gesture recognition datasets showing performance comparable to the state of the art, despite its relative simplicity and compactness. Our fast online action recognition algorithm exploits two computational speed-ups, namely the incremental covariance update and the on demand nearest neighbor computation, to meet real time requirements.

In our method we use the Affine-Invariant distance metric to compare covariance matrices. As noted in Section 2.2, this metric gave us better results than the Log-Euclidean metric, in terms of accuracy. That said, the advantage of the Log-Euclidean metric is that it can be seen as a projection of points on the manifold to

the Euclidean space, followed by the regular Euclidean metric computation. This may improve the accuracy beyond what we have obtained with the nearest neighbor classifier combined with the Affine-Invariant metric, using standard machine learning techniques. One of the possible extensions of this work could be deriving a more suitable classifier for our problem using either the Riemannian kernel [28] or the Riemannian pseudo-kernel [20].

Considering group actions is another possible directions for future work. The required adjustments depend on the type of the action. If the action is a collective motion of a group of people in some region of interest such as the case with the crowd actions, *e.g.*, demonstrations, parades, concerts, the *CovAct* descriptor may be employed as is. For more complex scenarios involving a structured interaction between several actors, the method has to be adapted according to the the available information. For example, if we are given the bounding boxes of each actor we may describe the motion of each one of them separately, and represent the action as a collection of such descriptors. Otherwise, if we are given a probability distribution of each actor's position inside the region of interest, we may still model actors separately by employing soft association of samples to actors based on the given distribution. Finally, if no additional information is given we may represent the complex action as a collective motion, and for moderately complex actions, *e.g.*, a mutual handshake, this may be just enough.

## Appendix A. Mean Update Rule

The derivation of the mean update rule is as follows:

$$\begin{aligned}\mathcal{M}_t &= \frac{1}{\mathcal{N}_t}\sum_{i=2}^{t}\sum_{j=1}^{N_i} x_j^i = \frac{1}{\mathcal{N}_t}\left(\sum_{i=1}^{t-1}\sum_{j=1}^{N_i} x_j^i - \sum_{j=1}^{N_1} x_j^1 + \sum_{j=1}^{N_t} x_j^t\right) \\ &= \frac{1}{\mathcal{N}_t}\left(\mathcal{N}_{t-1}\mathcal{M}_{t-1} - N_1\mu_1 + N_t\mu_t\right)\end{aligned} \tag{A.1}$$

## Appendix B. Covariance Update Rule

We would like to introduce the following lemma which will help us in the derivation of the covariance update rule.

**Lemma Appendix B.1.** *Let* $\{x_i\}_{i=1}^N$ *denote a sample of points in* $\mathbb{R}^n$*, and let* $\mu$ *and* $C$ *denote the sample mean and covariance matrix* i.e., $\mu = \sum_{i=1}^N x_i$ *and*

$C = \frac{1}{N-1}\sum_{i=1}^{N}(x_i-\mu)(x_i-\mu)^T$. *Then,*

$$\sum_{i=1}^{N}(x_i-\hat{\mu})(x_i-\hat{\mu})^T = (N-1)C + N(\mu-\hat{\mu})(\mu-\hat{\mu})^T, \tag{B.1}$$

*for any* $\hat{\mu} \in \mathbb{R}^n$.

**Proof**

$$\begin{aligned}
&\sum_{i=1}^{N}(x_i-\hat{\mu})(x_i-\hat{\mu})^T = \\
&= \sum_{i=1}^{N}\left((x_i-\mu)(x_i-\mu)^T + x_i\mu^T + \mu x_i^T - \mu\mu^T - x_i\hat{\mu}^T - \hat{\mu}x_i^T + \hat{\mu}\hat{\mu}^T\right) \\
&= (N-1)C + N(\mu-\hat{\mu})(\mu-\hat{\mu})^T.
\end{aligned} \tag{B.2}$$

□

The derivation of the covariance update rule is as follows:

$$\begin{aligned}
\mathcal{C}_t &= \frac{1}{\mathcal{N}_t-1}\sum_{i=2}^{t}\sum_{j=1}^{N_i}(x_j^i-\mathcal{M}_t)(x_j^i-\mathcal{M}_t)^T \\
&= \frac{1}{\mathcal{N}_t-1}(\sum_{i=1}^{t-1}\sum_{j=1}^{N_i}(x_j^i-\mathcal{M}_t)(x_j^i-\mathcal{M}_t)^T - \sum_{j=1}^{N_1}(x_j^1-\mathcal{M}_t)(x_j^1-\mathcal{M}_t)^T + \sum_{j=1}^{N_t}(x_j^t-\mathcal{M}_t)(x_j^t-\mathcal{M}_t)^T) \\
&= \frac{1}{\mathcal{N}_t-1}((\mathcal{N}_{t-1}-1)C + N(\mathcal{M}_{t-1}-\mathcal{M}_t)(\mathcal{M}_{t-1}-\mathcal{M}_t)^T - (N_1-1)C_1 - N_1(\mu_1-\mathcal{M}_t)(\mu_1-\mathcal{M}_t)^T + \\
&\qquad + (N_t-1)C_t + N_t(\mu_t-\mathcal{M}_t)(\mu_t-\mathcal{M}_t)^T) \\
&= \frac{1}{\mathcal{N}_t-1}((N-1)C - (N_1-1)C_1 + (N_t-1)C_t + \mathcal{N}_{t-1}\mathcal{M}_{t-1}\mathcal{M}_{t-1}^T - N_1\mu_1\mu_1^T + N_t\mu_t\mu_t^T - \\
&\qquad - \mathcal{M}_t(\mathcal{N}_{t-1}\mathcal{M}_{t-1} - N_1\mu_1 + N_t\mu_t)^T - (\mathcal{N}_{t-1}\mathcal{M}_{t-1} - N_1\mu_1 + N_t\mu_t)\mathcal{M}_t^T + \\
&\qquad + (N - N_1 + N_t)\mathcal{M}_t\mathcal{M}_t^T) \\
&= \frac{1}{\mathcal{N}_t-1}((\mathcal{N}_{t-1}-1)C - (N_1-1)C_1 + (N_t-1)C_t + \mathcal{N}_{t-1}\mathcal{M}_{t-1}\mathcal{M}_{t-1}^T - N_1\mu_1\mu_1^T + N_t\mu_t\mu_t^T - \\
&\qquad - \mathcal{N}_t\mathcal{M}_t\mathcal{M}_t^T).
\end{aligned} \tag{B.3}$$

The above update rule may be rewriten in the following way:

$$\mathcal{C}_t = \frac{1}{\mathcal{N}_t-1}\left((\mathcal{N}_{t-1}-1)\,\mathcal{C}_{t-1} + \Delta\mathcal{C}_t\right), \tag{B.4}$$

where

$$\Delta\mathcal{C}_t = -(N_1-1)C_1 + (N_t-1)C_t + \mathcal{N}_{t-1}\mathcal{M}_{t-1}\mathcal{M}_{t-1}^T - N_1\mu_1\mu_1^T + N_t\mu_t\mu_t^T - \mathcal{N}_t\mathcal{M}_t\mathcal{M}_t^T. \tag{B.5}$$

Expanding $\mathcal{C}_{t-1}$ we obtain:

$$\begin{aligned}\mathcal{C}_t &= \frac{1}{\mathcal{N}_t - 1}\left(\left(\mathcal{N}_{t-2} - 1\right)\mathcal{C}_{t-2} + \Delta\mathcal{C}_{t-1} + \Delta\mathcal{C}_t\right) \\ &= \dots \\ &= \mathcal{C}_t = \frac{1}{\mathcal{N}_t - 1}\left(\left(\mathcal{N}_{t-m} - 1\right)\mathcal{C}_{t-m} + \sum_{i=0}^{m-1}\Delta\mathcal{C}_{t-i}\right),\end{aligned} \tag{B.6}$$

for $m = 0, \dots, t$.

## References


[1] P. Turaga, R. Chellappa, V. Subrahmanian, O. Udrea, Machine Recognition of Human Activities: A Survey, IEEE Trans. Circuits and Systems for Video Technology (CSVT) 18 (11) (2008) 1473–1488.

[2] J. Yamato, J. Ohya, K. Ishii, Recognizing human action in time-sequential images using hidden Markov model, in: Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR), 379–385, 1992.

[3] T. Starner, A. Pentland, Visual Recognition of American Sign Language Using Hidden Markov Models, in: International Workshop on Automatic Face and Gesture Recognition (AFGR), 1995.

[4] A. Bobick, J. Davis, The Recognition of Human Movement Using Temporal Templates, IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI) 23 (3) (2001) 257–267.

[5] M. Blank, L. Gorelick, E. Shechtman, M. Irani, R. Basri, Actions as Space-Time Shapes, in: Proc. IEEE Int. Conf. on Computer Vision (ICCV), II: 1395–1402, 2005.

[6] A. Efros, A. Berg, G. Mori, J. Malik, Recognizing action at a distance, in: Proc. IEEE Int. Conf. on Computer Vision (ICCV), 726–733, 2003.

[7] S. Ali, M. Shah, Human Action Recognition in Videos Using Kinematic Features and Multiple Instance Learning, IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI) 32 (2) (2010) 288–303.

[8] L. Zelnik Manor, M. Irani, Statistical Analysis of Dynamic Actions, IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI) 28 (9) (2006) 1530–1535.

[9] D. Lowe, Distinctive Image Features from Scale-Invariant Keypoints, International Journal of Computer Vision (IJCV) 60 (2) (2004) 91–110.

[10] I. Laptev, T. Lindeberg, Space-Time Interest Points, in: Proc. IEEE Int. Conf. on Computer Vision (ICCV), 432–439, 2003.

[11] P. Dollar, V. Rabaud, G. Cottrell, S. Belongie, Behavior recognition via sparse spatio-temporal features, in: Performance Evaluation of Tracking and Surveillance (PETS), 65–72, 2005.

[12] J. Niebles, H. Wang, L. Fei Fei, Unsupervised Learning of Human Action Categories Using Spatial-Temporal Words, in: The British Machine Vision Conference (BMVC), 2006.

[13] T. Kim, R. Cipolla, Canonical Correlation Analysis of Video Volume Tensors for Action Categorization and Detection, IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI) 31 (8) (2009) 1415–1428.

[14] Y. Lui, J. Beveridge, M. Kirby, Action classification on product manifolds, in: Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR), 833–839, 2010.

[15] Y. Lui, A least squares regression framework on manifolds and its application to gesture recognition, in: Computer Vision and Pattern Recognition Workshops (CVPRW), 13–18, 2012.

[16] I. Guyon, V. Athitsos, P. Jangyodsuk, B. Hamner, H. Escalante, ChaLearn gesture challenge: Design and first results, in: Computer Vision and Pattern Recognition Workshops (CVPRW), 1–6, 2012.

[17] O. Tuzel, F. Porikli, P. Meer, Region Covariance: A Fast Descriptor for Detection and Classification, in: Proc. Eurpoean Conf. on Computer Vision (ECCV), II: 589–600, 2006.

[18] O. Tuzel, F. Porikli, P. Meer, Pedestrian Detection via Classification on Riemannian Manifolds, IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI) 30 (10) (2008) 1713–1727.

[19] K. Guo, P. Ishwar, J. Konrad, Action Recognition Using Sparse Representation on Covariance Manifolds of Optical Flow, in: IEEE Conf. Advanced Video and Signal-Based Surveillance (AVSS), 188–195, 2010.

[20] M. Harandi, C. Sanderson, A. Wiliem, B. Lovell, Kernel analysis over Riemannian manifolds for visual recognition of actions, pedestrians and textures, in: IEEE Workshop on the Applications of Computer Vision (WACV), 433–439, 2012.

[21] A. Sanin, C. Sanderson, M. Harandi, B. Lovell, Spatio-Temporal Covariance Descriptors for Action and Gesture Recognition, in: IEEE Workshop on the Applications of Computer Vision (WACV), 433–439, 2013.

[22] M. Albanese, R. Chellappa, N. Cuntoor, V. Moscato, A. Picariello, V. Subrahmanian, O. Udrea, PADS: A Probabilistic Activity Detection Framework for Video Data, IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI) 32 (12) (2010) 2246–2261.

[23] C. Ellis, S. Z. Masood, M. F. Tappen, J. J. Laviola, Jr., R. Sukthankar, Exploring the Trade-off Between Accuracy and Observational Latency in Action Recognition, International Journal of Computer Vision (IJCV) 101 (3) (2013) 420–436.

[24] T. Kanade, B. Lucas, An Iterative Image Registration Technique with an Application to Stereo Vision, in: International Joint Conference on Artificial Intelligence (IJCAI), 674–679, 1981.

[25] W. Forstner, M. B., A metric for covariance matrices, Tech. Rep., Dept. of Geodesy and Geoinformation, Stuttgart University, 1999.

[26] X. Pennec, P. Fillard, N. Ayache, A Riemannian Framework for Tensor Computing, International Journal of Computer Vision (IJCV) 66 (2006) 41–66.

[27] V. Arsigny, P. Fillard, X. Pennec, N. Ayache, Geometric Means in A Novel Vector Space Structure on Symmetric Positive-definite Matrices, SIAM J. Matrix Analysis and Applications 29 (2007) 328–347.

[28] Q. Dai, L. S. Davis, H. Guo, R. Wang, Covariance discriminative learning: A natural and efficient approach to image set classification, in: Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR), 2496–2503, 2012.

[29] A. Cherian, S. Sra, A. Banerjee, N. Papanikolopoulos, Efficient Similarity Search for Covariance Matrices via the Jensen-Bregman LogDet Divergence, in: Proc. IEEE Int. Conf. on Computer Vision (ICCV), 2399–2406, 2011.

[30] Y. Wu, J. Cheng, J. Wang, H. Lu, Real-time Visual Tracking via Incremental Covariance Tensor Learning, in: Proc. IEEE Int. Conf. on Computer Vision (ICCV), 1631–1638, 2009.

[31] K. Soomro, A. Roshan Zamir, M. Shah, UCF101: A Dataset of 101 Human Actions Classes From Videos in The Wild, in: CRCV-TR-12-01, 2012.

[32] A. Levenshtein, Binary Codes Capable of Correcting Deletions, Insertions and Reversals, in: Soviet Physics Doklady, vol. 10, 1966.