Activities as Time Series of Human Postures

William Brendel and Sinisa Todorovic

Oregon State University,
Kelley Engineering Center, Corvallis, OR 97331, USA
brendelw@onid.orst.edu, sinisa@eecs.oregonstate.edu

Abstract. This paper presents an exemplar-based approach to detecting and localizing human actions, such as running, cycling, and swinging, in realistic videos with dynamic backgrounds. We show that such activities can be compactly represented as time series of a few snapshots of human-body parts in their most discriminative postures, relative to other activity classes. This enables our approach to efficiently store multiple diverse exemplars per activity class, and quickly retrieve exemplars that best match the query by aligning their short time-series representations. Given a set of example videos of all activity classes, we extract multiscale regions from all their frames, and then learn a sparse dictionary of most discriminative regions. The Viterbi algorithm is then used to track detections of the learned codewords across frames of each video, resulting in their compact time-series representations. Dictionary learning is cast within the large-margin framework, wherein we study the effects of ℓ_1 and ℓ_2 regularization on the sparseness of the resulting dictionaries. Our experiments demonstrate robustness and scalability of our approach on challenging YouTube videos.

1 Introduction

This paper is about efficient, robust, and scalable activity recognition. Our thesis is that certain human actions, such as cycling, diving, walking, and horseback riding, can be compactly represented as short time series of a few still snapshots. Such a discrete activity representation captures discriminative parts of the human body and participating objects (e.g., racquet in playing tennis) in moments when they also assume discriminative postures. Their discriminativeness is defined relative to other human postures and objects seen across different activity classes, so as to allow robust activity recognition. Since there may be only a few time instances in which a few human-body parts strike discriminative poses, the entire space-time volume of a video gets hugely compressed by representing activities as time series. This allows us to develop a robust and scalable, exemplar-based approach to activity recognition in realistic videos with dynamic backgrounds. Numerous video exemplars per activity class can be efficiently stored as time series for the purposes of representing diverse, natural, inter- and intra-class variations. Also, retrieval of exemplars that best match the query can be efficiently done by aligning their short time-series representations.

Our approach consists of the following four computational steps: (1) extracting useful video features, (2) learning a dictionary of discriminative features extracted from a given set of exemplar videos, (3) representing videos as temporal sequences of the learned codewords, and (4) detecting and locating activities in a query video by aligning the query and exemplar time series. In the following, we give an overview of our approach, and point out our main contributions.

Feature Extraction: To represent activities, we extract hybrid features from videos, where the hybrid consists of appearance and local motion cues. Our motivation for using static appearance features comes from the well-known capability of human perception to recognize human actions from still images of activity-characteristic body postures [6, 7]. In cases when different actions (e.g., walking and running) produce similar static features, motion cues that we also extract will help resolve any ambiguity about static features. Prior work also often combines local motion and static features [1, 2, 3, 4, 5], since their extraction is reportedly more robust than that of other types of features, such as 2D+t volumes, optical flow, etc. We segment each video frame by the standard hierarchical meanshift algorithm, as in [8]. Meanshift regions are described by the HOG descriptor [9], shown to be stable and discriminative under a certain amount of partial occlusion, and changes in object pose [10]. HOG is computed using the spatial derivative of pixel intensities in the frame. HOG's are invariant to similar camera motions (e.g., panning) across videos, which may produce similar motion features of distinct actions. We also compute the temporal derivative of pixel intensities between two frames, resulting in the 2D+t HOG descriptor associated with every meanshift region.

Dictionary Learning: Given a large set of 2D+t HOG's, extracted from all exemplar videos, we learn a sparse dictionary of codewords, each representing the most discriminative 2D+t HOG's in the set. Since the HOG's are anchored at meanshift regions of video frames, the learned codewords may correspond to the entire human body, or body part, as well as to an object taking part in the activity (e.g., horses head in horseback riding, or swing in swinging). Existing work typically clusters video features by K-means [1,11,12], yielding codewords that may not be relevant for discriminating among the action classes. There is very little work on dictionary leaning for activity recognition, with few exceptions [13, 4]. Their information-bottleneck formulation, however, is intractable and requires approximation, which may not learn the optimal dictionary. In contrast, we cast dictionary learning within the large-margin framework, and derive an efficient, linear-complexity algorithm, with strong theoretical guarantees of small generalization error. Another key difference from prior work is that our codewords may represent objects defining the activity, in addition to human-body parts. This is critical for differentiating between very similar activities in which the human body undergoes similar motions but interacts with different objects (e.g., eating a banana vs. answering the phone). Most existing methods, however, do not account for objects that people interact with while performing the activity. This is because they seek to crop out only people from the videos by various means of background subtraction [14], or by applying people detectors [2, 12]. Recent studies show that activity recognition may improve when co-occurring objects in the context are identified [11]. Unlike all previous work, we use only video labels, i.e., weak supervision, for our dictionary learning.

Time-Series Representation: We represent videos as temporal sequences of codewords of the dictionary, learned in the previous step. Given a video, its time series

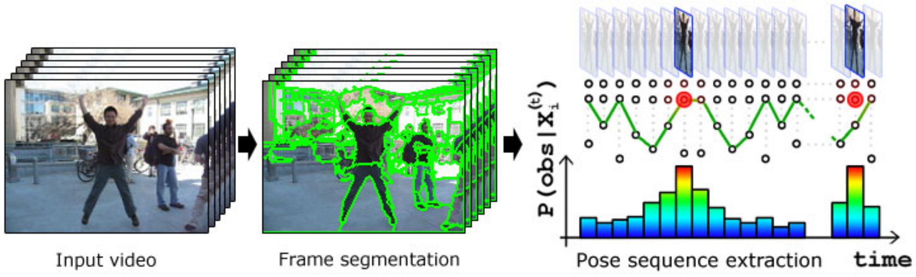


Fig. 1. Compact video representation: Meanshift regions, extracted from video frames, are matched with the codewords representing discriminative human-body parts and activity-defining objects. Best matching candidates are tracked across the frames by the Viterbi algorithm, resulting in a short time series of a few discriminative, still snapshots (marked red).

is computed by tracking candidate detections of the codewords in each frame, as illustrated in Fig. 1. For this tracking, we use the Viterbi algorithm which sequentially pursues the bast track at any given state, defined by a product of all codewords and meanshift segments in the visited frame. The codewords carry the information about their relative time locations in the exemplar videos from which they have been extracted. This allows the Viterbi algorithm to enforce the activity-characteristic temporal consistency of the resulting time-series representation. Prior work also seeks to represent videos as sequences of shape-motion prototypes [12]. However, they detect the prototypes in each frame, and thus generate long sequences of prototypes spanning all frames. Also, their prototypes represent the entire human body, giving our part-based codewords advantage in the presence of partial occlusions.

Recognition: Given a query video, and its time-series representation, it is aligned with the exemplar sequences by the cyclic dynamic time warping (CDTW) [15]. The activity label of the best aligned exemplar is transferred to the query, where their CDTW alignment also localizes the activity's occurrence in the space-time volume of the query. As shown in Sec. 5, we achieve the average recognition rate of 77.8% on challenging YouTube videos, outperforming the state-of-the-art result of 71.2% from [4].

Our Contributions include: (i) Four alternative, weakly supervised methods for learning a sparse dictionary of video features, formulated within the large-margin framework, using ℓ_1 or ℓ_2 regularizations; (ii) Proofs that the four methods converge to their respective globally optimal solutions, subject to the four distinct objective functions considered; (iii) Accounting for the co-occurrence statistics of objects and human actions in the scene, and thus extracting discriminative objects, which participate in the activity, along with discriminative human postures; and (iii) Robust and scalable exemplar-based approach to activity detection and localization in videos.

In the following, Sec. 2 explains the video features we use, Sec. 3 presents the four algorithms for dictionary learning and proofs of their convergence, Sec. 4 describes how to extract and align the time-series representations of videos for activity recognition, and Sec. 5 presents our experimental evaluation.

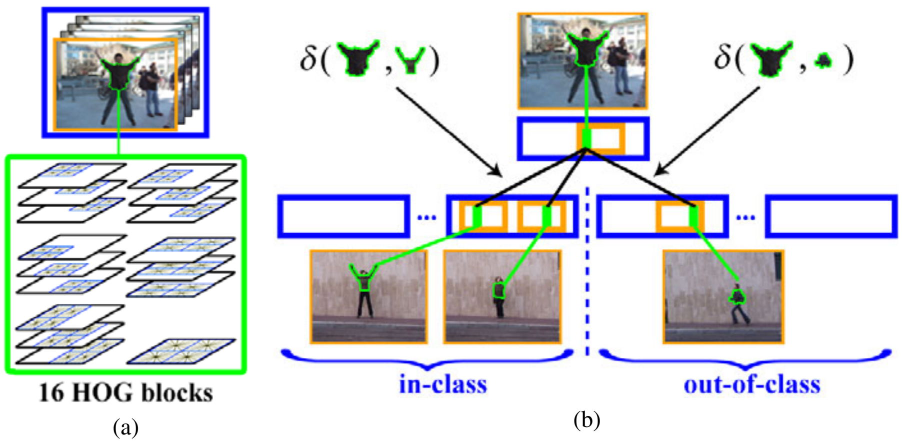


Fig. 2. (a) The meanshift regions (green) of all frames (orange) in a video (blue) are characterized by the 2D+t HOG descriptors, called hybrid features as they combine static appearance and motion cues. The 2D+t HOG of a meanshift region uses orientations of spatial and temporal gradients of pixel intensities, extracted from 16 overlapping windows covering the region. (b) Computing distances between in-class and out-of-class videos. (best viewed in color).

2 Feature Extraction

This section specifies appearance and local motion features that we use in this paper. Each frame is first partitioned into segments using the standard hierarchical meanshift algorithm, as in [8]. The segments provide static appearance features, at multiple scales. Each meanshift region is then described using a 2D+t HOG descriptor, which additionally incorporates local motion cues. The 2D+t HOG extends the standard HOG [9], which has been shown to exhibit invariance to partial occlusion and object deformations [10]. We first use the difference operators in time and space to compute the 2D+t gradient vectors at every pixel of the meanshift region. Then, we project these 3D vectors onto the *x*-*y*, *x*-*t*, and *y*-*t* planes. Next, each projection is covered by 16 overlapping blocks, as shown Fig. 2a. From each block we extract a 36-dimensional histogram of oriented gradients (9 bins for 4 cells within one block). By concatenating the three 36D histograms from *x*-*y*, *x*-*t*, and *y*-*t* planes, we obtain the 2D+t HOG with 108 dimensions.

3 Learning the Dictionary of Activity Codewords

In this section, we specify four alternative algorithms for learning the dictionary of discriminative activity features, and present their convergence analysis. We begin by introducing some notation and basic definitions. Suppose that we are given a set of annotated exemplar videos $\mathbb{D} = \{\boldsymbol{x}_i, y_i\}$, where $\boldsymbol{x}_i = [\boldsymbol{x}_{i1}, \ldots, \boldsymbol{x}_{ik}, \ldots]^{\mathrm{T}}$ denotes all 2D+t HOG's extracted from all frames of video i, and y_i is the associated label of activity class. Note that different videos may have different total numbers of features.

Our goal is to identify the most discriminative features in the entire set $\mathbb{D}$, called codewords. In this paper, we consider learning two types of dictionaries. If the codewords are learned so a given class is discriminated well against the other classes, we obtain the dictionary of that class. If the codewords are learned to discriminate well all classes, they form the all-class dictionary.

We formulate dictionary learning within the large-margin framework. Margins play a crucial role in the modern machine learning [17]. They measure the confidence of a classifier when making a decision. There are two types of margins. The more common type, called sample-margin, used for example in SVMs, measures how far positive and negative training examples are separated by the decision surface. In this paper, we consider the other type called hypothesis-margin. It is defined per data instance, and measures a distance between the hypothesis and the closest hypothesis that assigns alternative label to that instance. In particular, for each $\boldsymbol{x}_i$, we seek to maximize its distance to all out-of-class videos, called misses. At the same time, we wish to minimize its distance to all videos belonging to the same class, called hits. These two objectives can be achieved by maximizing the hypothesis-margin of the one-nearest-neighbor classifier (1NN). Maximizing the sample-margin of the SVM has been used for dictionary learning in [16]. However, this formulation, is not suitable for videos, since it would lead to a large scale optimization problem of prohibitive complexity.

To specify the hypothesis-margin of 1NN, we define an asymmetric distance between two videos, $d_{ij} = d(\boldsymbol{x}_i, \boldsymbol{x}_j)$, as a weighted sum of distances between their best matching features, $d_{ij} = \boldsymbol{\delta}_{ij}^{\mathrm{T}} \boldsymbol{w}_i$. The vector $\boldsymbol{\delta}_{ij} = [\delta_{ij1}, \ldots, \delta_{ijk}, \ldots]^{\mathrm{T}}$ consists of χ^2 distances between the histograms of each 2D+t HOG descriptor, x_{ik}, and its best matching descriptor in $\boldsymbol{x}_j$, $\delta_{ijk} = \min_l \chi^2(\boldsymbol{x}_{ik}, \boldsymbol{x}_{jl})$. The non-negative weights, $\boldsymbol{w}_i \geq \mathbf{0}$, and distances $\boldsymbol{\delta}_{ij}$ are associated with features of the first video in the pair, $\boldsymbol{x}_i$, and thus $\boldsymbol{x}_i$, $\boldsymbol{w}_i$, and $\boldsymbol{\delta}_{ij}$ have the same length. Note that the weights $\boldsymbol{w}_i$ serve to indicate the relevance of the corresponding features in $\boldsymbol{x}_i$ for discriminating between activity classes y_i and y_j. Our goal is to learn $\boldsymbol{w}_i$ for all videos $\boldsymbol{x}_i$, so as to maximize the hypothesis-margin of 1NN, and then extract video features with the highest weights to the dictionary. We specify the hypothesis-margin of specific $\boldsymbol{x}_i$ as

$$\rho_i = d_{im} - d_{ih} = (\boldsymbol{\delta}_{im} - \boldsymbol{\delta}_{ih})^{\mathrm{T}} \boldsymbol{w}_i \ , \tag{1}$$

where index m denotes that $\boldsymbol{\delta}_{im}$ is computed with the nearest miss of $\boldsymbol{x}_i$, and index h denotes that $\boldsymbol{\delta}_{ih}$ is computed with the nearest hit of $\boldsymbol{x}_i$. From (1), it follows that maximizing the hypothesis-margin of 1NN will amount to maximizing the distances of all videos from their respective out-of-class videos, and simultaneously minimizing the distances of all videos to their respective in-class videos. This can be formulated using the following notation. Let $\boldsymbol{w}$ be a column vector of concatenated weights $\boldsymbol{w}_i$ for all $\boldsymbol{x}_i \in \mathbb{D}$; $\boldsymbol{z}_m$ be a column vector of concatenated feature distances $\boldsymbol{\delta}_{im}$ for all $\boldsymbol{x}_i \in \mathbb{D}$ to their respective nearest misses; and $\boldsymbol{z}_h$ be a column vector of concatenated feature distances $\boldsymbol{\delta}_{ih}$ for all $\boldsymbol{x}_i \in \mathbb{D}$ to their respective nearest hits. Finally, let $\boldsymbol{z} = \max(\mathbf{0}, \boldsymbol{z}_m - \boldsymbol{z}_h)$. Then, dictionary learning can be specified as the following linear program (LP):

$$\underset{\boldsymbol{w}}{\operatorname{argmax}} \ \boldsymbol{z}^{\mathrm{T}} \boldsymbol{w}, \quad \text{s. t. } \boldsymbol{w} \geq \mathbf{0}, \ \text{ and } \ \|\boldsymbol{w}\| \leq \gamma \ , \tag{2}$$

where γ is a positive constant, and $\|\cdot\|$ is either ℓ_1 or ℓ_2 norm. After solving (2), features with non-zero weights will be selected as codewords in the dictionary.

When $\boldsymbol{w}$ and $\boldsymbol{z}$ represent the concatenation of feature weights and distances across all videos, the resulting dictionary will be all-class. Similarly, the dictionary of a specific class can be derived by concatenating into $\boldsymbol{w}$ and $\boldsymbol{z}$ the appropriate values for only those videos that belong to that class.

Note that (2) represents an extremely large optimization problem. Any naive use of general LP solvers, such as simplex or interior point methods, would be computationally too expensive. In the sequel, we propose four alternative algorithms to solve (2), which are very efficient, with linear complexity in the number of input video features.

3.1 Logistic-Regression Formulation

In this subsection, we employ the logistic regression formulation to solve our large LP problem, given by (2). Specifically, to eliminate the constraint $\|\boldsymbol{w}\| \leq \gamma$ from (2), we add a penalty term, $\lambda \|\boldsymbol{w}\|$, directly to the objective function, where λ is a non-negative input parameter. Note, however, that the objective function of (2) represents maximization, whereas the constraint $\|\boldsymbol{w}\| \leq \gamma$ requires minimization. This can be resolved by reformulating (2) as

$$\underset{\boldsymbol{w}}{\operatorname{argmin}} \ \log[1+\exp(-\boldsymbol{z}^{\mathrm{T}}\boldsymbol{w})] + \lambda \|\boldsymbol{w}\|, \ \text{ s.t. } \boldsymbol{w} \geq 0 \,. \tag{3}$$

Note that λ controls the sparseness of the solution, and thus the number of selected codewords in the dictionary.

Eq. (3) is a constrained convex optimization problem. Due to the non-negative constraint on $\boldsymbol{w}$, it cannot be solved directly by gradient descent. To overcome this difficulty, we use the following substitution $\boldsymbol{w} = [v_1^2, \ldots, v_k^2, \ldots]^{\mathrm{T}}$, where v_k are auxiliary variables, and k is the index over all 2D+t HOG's. This gives

$$\underset{\boldsymbol{w}}{\operatorname{argmin}} \ \log[1+\exp(-\textstyle\sum_k z_k v_k^2] + \lambda R(\boldsymbol{v}), \tag{4}$$

where $R(\boldsymbol{v}) = \|\boldsymbol{v}\|_2^2$ for ℓ_1 regularization, or $R(\boldsymbol{v}) = \sqrt{\sum_k v_k^4}$ for ℓ_2 regularization. Consequently, we obtain an unconstrained optimization problem. It is straightforward to derive the following gradient-descent solution of (4):

$$LR\text{-}\ell_1: \ v_k \leftarrow v_k - \eta\left(\lambda - \frac{\exp(-\sum_k z_k v_k^2)}{1+\exp(-\sum_k z_k v_k^2)}\right)\cdot v_k, \qquad \text{for } \ell_1, \tag{5}$$

$$LR\text{-}\ell_2: \ v_k \leftarrow v_k - \eta\left(\lambda\frac{v_k^2}{\sqrt{\sum_k v_k^4}} - \frac{\exp(-\sum_k z_k v_k^2)}{1+\exp(-\sum_k z_k v_k^2)}\right)\cdot v_k, \quad \text{for } \ell_2, \tag{6}$$

where η is the learning rate determined by the standard line search. Once v_k are estimated, we then compute the feature relevances as $w_k = v_k^2$, $k = 1, 2, \ldots$. The convergence of this logistic-regression based algorithm is explained at the end of this section, after we specify the other two alternative algorithms.

3.2 Alternative LP Formulation

In practice, the update rules given by (5) and (6) have a serious limitation. In particular, if the term $\sum_k z_k v_k^2$ is large, then $\exp(-\sum_k z_k v_k^2)$ drops exponentially to zero, and the update depends only on the penalty term. To overcome this problem, we modify the LP given by (2), as follows.

Without a loss of generality, we replace the constraint $\|\boldsymbol{w}\| \leq \gamma$ by $\|\boldsymbol{w}\| = \gamma$, leading to the following new LP formulation

$$\underset{\boldsymbol{w}}{\operatorname{argmax}}\ \boldsymbol{z}^{\mathrm{T}} \frac{\boldsymbol{w}}{\|\boldsymbol{w}\|}, \text{ s. t. } \boldsymbol{w} \geq 0. \tag{7}$$

As in Sec. 3.1, the non-negative constraint in (7) can be reformulated by using the following substitution $\boldsymbol{w} = [v_1^2, \ldots, v_k^2, \ldots]^{\mathrm{T}}$, where v_k are auxiliary variables, and k is the index over all video features. This gives

$$\underset{\boldsymbol{w}}{\operatorname{argmax}}\ \frac{1}{R(\boldsymbol{v})} \sum_k z_k v_k^2, \qquad \boldsymbol{w} = [v_1^2, \ldots, v_k^2, \ldots]^{\mathrm{T}}, \tag{8}$$

where, as in (4), $R(\boldsymbol{v}) = \|\boldsymbol{v}\|_2^2$ for ℓ_1, or $R(\boldsymbol{v}) = \sqrt{\sum_k v_k^4}$ for ℓ_2 regularization. It is straightforward to derive the following gradient-ascent solution of (8):

$$LP\text{-}\ell_1 : \ v_k \leftarrow v_k + \eta \frac{\left(z_k \sqrt{R(\boldsymbol{v})} - \sum_k z_k v_k^2\right)}{R(\boldsymbol{v})} \cdot v_k, \qquad \text{for } \ell_1, \tag{9}$$

$$LP\text{-}\ell_2 : \ v_k \leftarrow v_k + \eta \frac{\left(z_k \sqrt{R(\boldsymbol{v})} - \frac{v_k^2}{R(\boldsymbol{v})} \sum_k z_k v_k^2\right)}{R(\boldsymbol{v})} \cdot v_k, \qquad \text{for } \ell_2, \tag{10}$$

where η is the learning rate determined by the standard line search. Once v_k are estimated, we then compute the feature relevances as $w_k = v_k^2$, $k = 1, 2, \ldots$.

Convergence: In both LP formulations, presented in Sec. 3.1 and 3.2, we reformulate the non-negative constraints in (3) and (7). The resulting objective functions, given by (4) and (8), are convex and concave, respectively. Consequently, the gradient descent in (5)–(6), and the gradient ascent in (9)–(10) converge to their respective globally optimal solutions. The full proof that (4) is convex, and (8) is concave is given in the supplemental material. The proof first shows that the substitution $w_k = v_k^2$, $k = 1, 2, \ldots$, does not change the concavity of the original LP formulation, given by (2). Then, we use the classical theoretical results in convex optimization about the convexity and concavity of a composition of two functions ($f \circ g$) to prove that the logistic regression formulation is convex, and the alternative normed objective is concave.

Complexity of both formulations presented in Sec. 3.1 and 3.2 is linear in the number of input video features. Since our features are descriptors of meanshift segments, the total number of our features is significantly smaller than interest-point features, typically used in existing approaches to activity recognition.

After convergence, all 2D+t HOG's from all videos in $\mathbb{D}$ whose weights are nonzero are declared as codewords. Finally, the 2D+t HOG descriptor of each codeword is augmented with the time stamp of a frame from which the codeword has been extracted, normalized relative to the length of the originating video. This is used to enforce the temporal consistency of codewords along time series representing videos.

4 Representing Videos as Time Series of Activity Codewords

This section describes how to compute the time-series representation of a video. We first extract multiscale meanshift regions in each video frame, and then match their 2D+t HOGs with the codewords. The standard Viterbi algorithm is applied to track the best matches (Fig. 1), where for each frame only one best matching codeword-region pair is selected. Tracking seeks to maximize the joint likelihood of all matches along the Viterbi path, under the constraint that the tracked codewords along the path are locally smooth in the 2D space, and temporally consistent. In the following, we specify the Viterbi algorithm.

Let $\Omega = \{\boldsymbol{\omega}_l\}_{l=1,2,\dots}$ denote the dictionary of activity codewords, and let $\boldsymbol{x}^{(t)} = \{\boldsymbol{x}_k^{(t)}\}_{k=1,2,\dots}$ denote 2D+t HOG's extracted from frame t of video $\boldsymbol{x}$. In each frame t, the goal of the Viterbi algorithm is to select a single, best matching pair $(\boldsymbol{x}_k^{(t)}, \boldsymbol{\omega}_l)$ out of the entire product space $\boldsymbol{x}^{(t)} \times \Omega$. The selected, unique pair $(\boldsymbol{x}_k^{(t)}, \boldsymbol{\omega}_l)$ is referred to as instantiation of codeword $\boldsymbol{\omega}_l$ in frame t, and denoted as $\hat{\boldsymbol{\omega}}_l^{(t)} = (\boldsymbol{x}_k^{(t)}, \boldsymbol{\omega}_l)$. Across all frames, the goal of the Viterbi algorithm is to satisfy the temporal constraints between the instantiated codewords $\{\hat{\boldsymbol{\omega}}_l^{(t)}\}_{t=1,2,\dots}$, and produce a locally smooth trajectory in the 2D space. Temporal consistency is enforced via a Markov chain which is informed by the time stamps associated with codewords, as mentioned in the previous section. To formalize the above two goals of the Viterbi algorithm, we below first specify the likelihood that measures the quality of matches between video features and codewords, and then define the transition probability of the Markov chain which favors spatially smooth and temporally consistent codeword instantiations from one frame to another.

Video feature $\boldsymbol{x}_k^{(t)}$ matches codeword $\boldsymbol{\omega}_l$ with likelihood $P(\boldsymbol{x}_k^{(t)}|\boldsymbol{\omega}_l) \propto e^{-\alpha\chi^2(\boldsymbol{x}_k^{(t)}, \boldsymbol{\omega}_l)}$, where $\alpha = 0.01$ (empirically found at equal error rate) weights the χ^2 histogram distance (for equal error rate, we get). The Markov-chain transition probability is defined as $P(\hat{\boldsymbol{\omega}}_l^{(t)}|\hat{\boldsymbol{\omega}}_j^{(t-1)}) \propto e^{-\boldsymbol{\beta}^{\mathrm{T}}|\hat{\boldsymbol{\omega}}_l^{(t)} - \hat{\boldsymbol{\omega}}_j^{(t-1)}|}$, where $\boldsymbol{\beta} = [0.1, 0.1]^{\mathrm{T}}$ (empirically found at equal error rate), and $|\cdot|$ denotes the absolute difference of the corresponding spatial and time coordinates of the instantiated codewords $\hat{\boldsymbol{\omega}}_l^{(t)}$ and $\hat{\boldsymbol{\omega}}_j^{(t-1)}$. Specifically, for their spatial coordinates, we take the centroids of meanshift regions that got matched to $\boldsymbol{\omega}_l$ and $\boldsymbol{\omega}_j$ in frames t and $(t-1)$. For their time coordinates, we take the time stamps that $\boldsymbol{\omega}_l$ and $\boldsymbol{\omega}_j$ carry from their source exemplar videos. With these definitions, we specify the Viterbi algorithm as finding the optimal sequence of codewords so the following Markov chain is maximized:

$$P(\hat{\boldsymbol{\omega}}_l^{(t)}) = \max_{\hat{\boldsymbol{\omega}}_j^{(t-1)},\ \boldsymbol{x}_k^{(t)}} P(\hat{\boldsymbol{\omega}}_j^{(t-1)}) P(\hat{\boldsymbol{\omega}}_l^{(t)}|\hat{\boldsymbol{\omega}}_j^{(t-1)}) P(\boldsymbol{x}_k^{(t)}|\hat{\boldsymbol{\omega}}_l^{(t)}), \tag{11}$$

where $P(\hat{\boldsymbol{\omega}}_j^{(t-1)})$ is recursively defined. The Viterbi algorithm retrieves the best path across the frames (Fig. 1) with linear complexity in the number of video features.

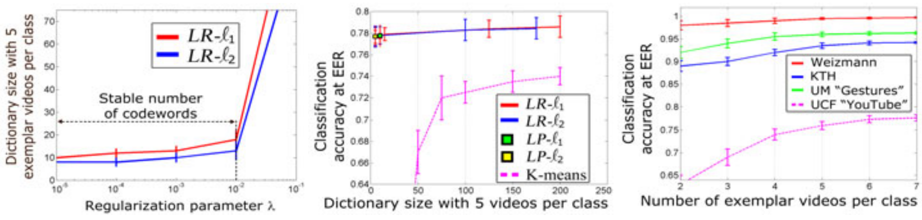
Extracting the Compact Representation: The obtained Viterbi path is characterized by a sequence of likelihoods $P(\boldsymbol{x}_k^{(t)}|\hat{\boldsymbol{\omega}}_l^{(t)})$, $t = 1, 2, \dots$. This sequence has modes and valleys, as illustrated in Fig. 1. The valleys indicate low confidence in the corresponding codeword instantiations. We identify and eliminate the valleys in this likelihood sequence by the popular quick-shift mode-seeking algorithm [18]. As a result, we obtain the compact time-series representation.

Exemplar-based Recognition: Given a query video, we use the same algorithm to extract its time series of codewords. For recognition, we align the time series of the query and exemplar videos. Note that the sought activity may not start at the beginning, or finish at the end of the query video. Therefore, the query-exemplar alignment is not only aimed at finding the best matching exemplar, but also to localize a subsequence of codewords, in the query time series, that represents the activity. The label of the best aligned exemplar is taken as the activity class of the query. Also, the codewords identified to represent the activity in the time series are back-tracked to the space-time locations of the corresponding meanshift regions in the query video. All this results in the simultaneous detection and localization of the activity in the query video. In this paper, two temporal sequences of codewords are aligned by the cyclic dynamic time warping (CDTW), presented in [15]. CDTW finds correspondences between codewords of the two sequences by identifying the optimal path in a cost matrix of all pairwise codeword matches. This is done by respecting the ordering of each input sequence. The costs are χ^2 distance between the 2D+t HOG histograms of each codeword. We use the cyclic variant of DTW, because it efficiently identifies the optimal start and end of the alignment path in the cost matrix, regardless of the lengths of the input sequences. Complexity of CDTW is linear in the total number of elements in the two sequences.

5 Results

Experiments are conducted on five benchmark datasets: Weizmann activities [14], KTH [19], UM "Gestures" [12], CMU "Crowded" videos [8], and UCF "YouTube" [4]. KTH contains a varied set of challenges, including scale changes, variation in the speed of activity execution, and indoor and outdoor illumination variations. In UM "Gestures", training videos are captured by a static, high-resolution camera, with the person standing in front of a uniform background; whereas test videos are captured by a moving camera, in the presence of a background clutter, and other moving objects. The CMU "Crowded" videos are acquired by a hand-held camera, in unconstrained environments, with moving people or cars in the background. Each CMU video may contain several target actions, where we identify only one. This dataset is challenging due to significant spatial- and temporal-scale differences in how the subjects perform the actions. In the UCF "YouTube" videos the actors interact with objects, such as a horse, bicycle, or dog, which define the corresponding activities. This dataset is challenging due to: a mix of steady and shaky cameras, cluttered background, low resolution, and variations in scale, viewpoint, and illumination.

For activity recognition, we use 5 exemplars per each class from the considered dataset. The activity class of a given query is defined by a majority voting of M, best-aligned exemplars, where M is estimated by the leave-one-out (LOO) strategy. We report the average classification accuracy at equal error rate (EER), where the accuracy is



(a) (b) (c)

Fig. 3. (a) Average dictionary size per activity class in the UCF "YouTube" dataset as a function of the regularization parameter λ in LR-ℓ_1 and LR-ℓ_2. (b) Classification accuracy at EER averaged over the UCF "YouTube" classes vs. the average size of the dictionary generated by LR-ℓ_1, LR-ℓ_2, LP-ℓ_1, LP-ℓ_2, and unsupervised K-means clustering of 2D+t HOGs. (c) Average classification accuracy on all datasets vs. the number of available exemplar videos, when the dictionary is learned by LP-ℓ_1. (best viewed in color).

averaged over all classes in the dataset. On all datasets, we achieve EER for input parameters $\lambda = 10^{-3}$, $\alpha = 0.01$, and $\boldsymbol{\beta} = [0.1, 0.1]$. In the following, we present evaluation of the individual steps of our approach.

Dictionary Learning: In the following two experiments, we use the UCF "YouTube" dataset to extract distinct dictionaries for each class (not the all-class dictionary). First, we evaluate our sensitivity to the specific choice of λ in LR-ℓ_1 and LR-ℓ_2. Fig. 3a shows the average dictionary size as a function of input λ values, where each dictionary is learned on five exemplar videos per class, and the dictionary size is averaged over all "YouTube" classes. As can be seen, for a wide range of λ values, when $\lambda < 10^{-2}$, both LR-ℓ_1 and LR-ℓ_2 produce a "stable" number of codewords. Second, we evaluate our classification accuracy at equal error rate (EER) versus the average size of different dictionary types produced by LR-ℓ_1, LR-ℓ_2, LP-ℓ_1, LP-ℓ_2, as well as by unsupervised K-means clustering. Fig. 3b shows that all our learning methods outperform the unsupervised clustering of video features by K-means. As can be seen in Fig. 3b, when using all four learning methods we achieve similar classification accuracy.

Depending on a particular application, one may prefer to work with the dictionary generated by LP-ℓ_1, because LP-ℓ_1 yields the sparsest solution with the fewest codewords, and it does not require any input parameter (unlike LR-ℓ_1 and LR-ℓ_2). Therefore, in the following, we continue with evaluation of our approach when using only LP-ℓ_1 for dictionary learning.

Accuracy vs. Number of Exemplars: We test our performance on each dataset versus the number of randomly selected exemplar videos per class. Classification accuracy is averaged over all classes within the specific dataset. Fig. 3c shows that only few exemplars are needed to achieve high accuracy for the challenging datasets.

HOG vs. 2-D+t HOG: We test whether adding motion cues to the standard HOG increases performance. Fig. 4 shows that our performance on the UCF "YouTube" videos is better with 2D+t HOG's than that with HOG's, since the additional motion features help disambiguate similar static appearance features.

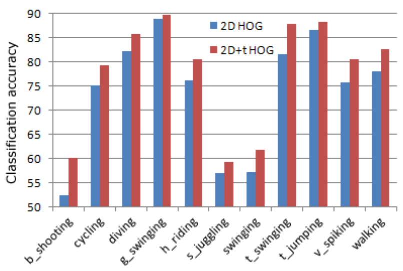


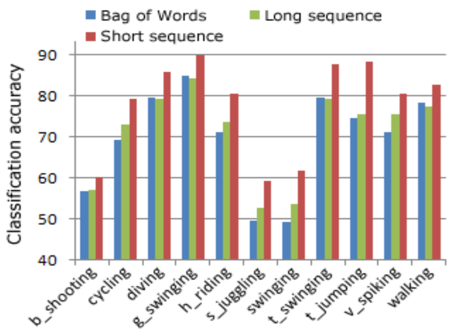
Fig. 4. Classification accuracy at EER when using HOG's and 2D+t HOG's on the UCF "YouTube" dataset, for $LP\text{-}\ell_1$

Table 1. Recall of detecting relevant video parts for activity recognition by our Viterbi algorithm, evaluated with respect to the manually annotated bounding boxes around actors, and averaged over all videos and classes

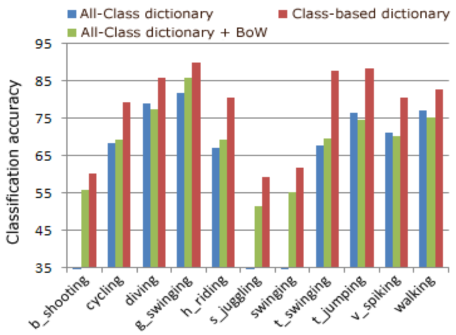
	Recall
Weizmann	0.95
KTH	0.94
UM "Gestures"	0.95

Viterbi-based Codeword Tracking: We evaluate recall of our Viterbi-based detection of relevant video parts for activity recognition. To this end, we use the ground-truth bounding boxes around actors, provided in the Weizmann, KTH and UM "Gestures" datasets. Ideally, the Viterbi algorithm would associate codewords with those meanshift regions that fall within the bounding boxes in every video frame. We estimate recall as a ratio between the number of true positives and the total number of frames, where a true positive is a detected meanshift region with more than 50% of its area falling within the bounding box. Our recall averaged over all videos and classes is shown in Table 1.

Viterbi vs. Bag-of-Words: Tracking codewords by the Viterbi algorithm increases complexity vs. a simpler Bag of Words (BoW) approach, which scans all meanshift regions, and finds the best matching region-codeword pair, in each frame, irrespective of the results in other frames. The increased complexity is justified by significant increase in our classification accuracy vs. BoW, as shown in Fig. 5a.



(a)



(b)

Fig. 5. Classification accuracy at EER of Bag-of-words, and our approach with $LP\text{-}\ell_1$, on the UCF "YouTube" dataset: (a) Our approach uses short time series, and long sequence of codewords as the video representation. The short time series enables faster and more accurate activity recognition. (b) Our approach uses the all-class dictionary and a set of dictionaries learned per class. The class-based dictionary learning gives better performance.

Table 2. AUC for CMU "Crowded" videos

	[3]	[8]	Ours (LP-ℓ_1)
pick-up	0.58	0.47	0.60
one-hand wave	0.59	0.38	0.64
jumping jack	0.43	0.22	0.45
two-hands wave	0.43	0.64	0.65

Table 3. Average classification accuracy at EER

	[14]	[12]	[4]	[3]	Ours (LP-ℓ_1)
Weizmann	97.5	X	X	X	99.7
KTH	X	95.7	91.8	87.8	94.2
UM "Gestures"	X	95.2	X	X	96.3
UCF "YouTube"	X	X	71.2	X	77.8

Long vs. Short Time Series: After the Viterbi algorithm has identified the optimal path of codewords in a video, we eliminate a number of codeword detections with low confidence, and thus extract the short time series representation. Fig. 5a shows significant performance gains, on the the "YouTube" dataset, when using the short time series vs. the long sequence of codewords instantiated in every video frame, as the video representation. In addition, the short time series enable nearly two-orders-of-magnitude speed ups of recognition. On the "YouTube" videos, recognition by aligning long sequences (whose size is the same as the number of video frames) takes on average 302.2ms, whereas short time series are aligned in only 4.6ms. Our implementation is in C on 2.8GHz 8GB RAM PC.

All-class dictionary vs. class-based dictionaries: Fig. 5b compares our performance, when using a set of dictionaries learned per class vs. the all-class dictionary. As can be seen, the all-class dictionary yields inferior performance. This is because the all-class dictionary is typically very sparse, so that an activity class may not be even represented by any codeword (see 'b_shooting', 's_juggling' and 'swinging'). Interestingly, for a few classes, BoW with the all-class dictionary outperforms our approach with the all-class dictionary.

Training Transfer: We evaluate whether our approach can be trained on a simple, sanitized setting of the Weizmann videos, and then used for activity recognition on the challenging CMU "Crowded" videos. Specifically, we use 5 exemplar videos per class

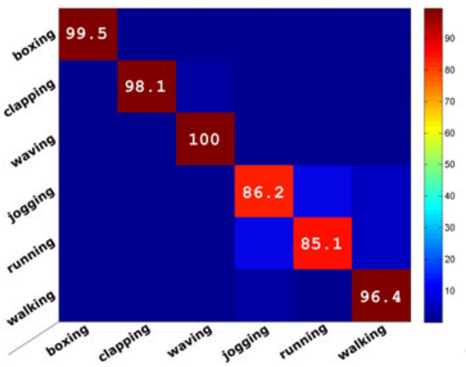


Fig. 6. Our confusion matrix for KTH

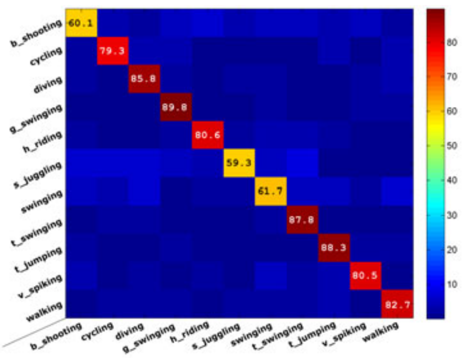


Fig. 7. Our confusion matrix for UCF videos

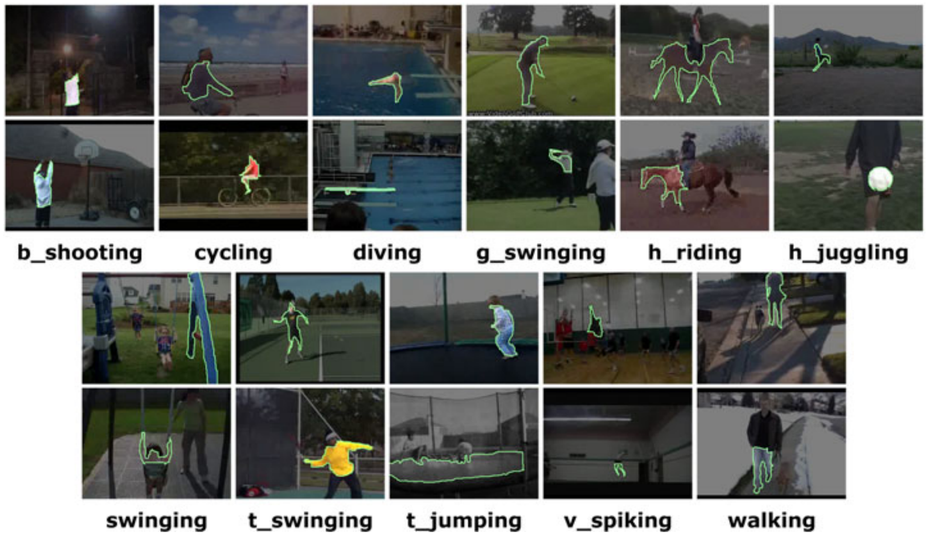


Fig. 8. Examples of the learned codewords from the UCF "YouTube" dataset. The codewords are highlighted in the frames of exemplar videos from which the codewords have been extracted.

from the Weizmann dataset, and take queries from the CMU "Crowded" videos. Table 2 shows the area under the ROC curve (AUC) that we have obtained for LP-ℓ_1 by varying the values of input parameters α and $\boldsymbol{\beta}$. As can be seen, even when our training occurs on the sanitized dataset, our AUC values, for four different activity classes, are better than that of the competing approaches [3, 8].

Other Evaluation: Table 5 shows that we compare favorably with the state-of-the-art. We also provide confusion matrices for KTH and UCF "YouTube" datasets in Fig. 5 and Fig. 5. Fig. 8 shows two examples of the learned codewords for each class of the "YouTube" dataset. As can be seen, the codewords may represent only a body part, or objects defining the activity (the trampoline for 't_jumping' or the swinging gear for 'swinging').

6 Conclusion

We have shown that certain human actions can be efficiently represented by short time series of activity codewords. The codewords represent still snapshots of human body parts in their discriminative postures, relative to other activity classes. In addition, the codewords may represent discriminative objects that people interact with while performing the activity. Typically, our time series representation compresses the original hundreds of video frames to only about 10 key human postures. This carries many advantages for developing a robust, efficient, and scalable activity recognition system. Our main focus has been on specifying four alternative methods for learning the dictionary of codewords from a large set of static and local-motion video features, under only weak supervision. We have formulated this learning as maximization of the hypothesis margin of the 1-NN classifier with ℓ_1 and ℓ_2 regularization. For the four learning methods, we have presented strong theoretical guarantees of their convergence to the globally optimum solution. The methods have linear complexity in the number of video features, and small generalization error. We have evaluated the proposed time-series representation on the challenging problem of activity detection and localization in realistic videos (YouTube) with dynamic, cluttered backgrounds. Our activity recognition yields better performance when using a set of dictionaries learned per each activity class than the all-class dictionary. Interestingly, significant classification-accuracy gains are achieved when using the short time series of codewords vs. a long sequence of codewords (one per each video frame) as the video representation. Our results show that, with small computation times, we outperform the state of the art on the benchmark datasets.

References

1. Laptev, I., Marszalek, M., Schmid, C., Rozenfeld, B.: Learning realistic human actions from movies. In: CVPR (2008)
2. Niebles, J.C., Han, B., Ferencz, A., Fei-Fei, L.: Extracting moving people from internet videos. In: Forsyth, D., Torr, P., Zisserman, A. (eds.) ECCV 2008, Part IV. LNCS, vol. 5305, pp. 527–540. Springer, Heidelberg (2008)
3. Yao, B., Zhu, S.C.: Learning deformable action templates from cluttered videos. In: ICCV (2009)
4. Liu, J., Luo, J., Shah, M.: Recognizing realistic actions from videos "in the wild". In: CVPR (2009)
5. Fanti, C., Zelnik-Manor, L., Perona, P.: Hybrid models for human motion recognition. In: CVPR (2005)
6. Bissacco, A., Yang, M.H., Soatto, S.: Detecting humans via their pose. In: NIPS (2007)
7. Ning, H., Xu, W., Gong, Y., Huang, T.: Discriminative learning of visual words for 3d human pose estimation. In: CVPR (2008)
8. Ke, Y., Sukthankar, R., Hebert, M.: Event detection in crowded videos. In: ICCV (2007)
9. Dalal, N., Triggs, B.: Histograms of oriented gradients for human detection. In. CVPR, pp. 886–893 (2005)
10. Lin, Z., Davis, L.S.: A pose-invariant descriptor for human detection and segmentation. In: Forsyth, D., Torr, P., Zisserman, A. (eds.) ECCV 2008, Part IV. LNCS, vol. 5305, pp. 423–436. Springer, Heidelberg (2008)
11. Marszalek, M., Laptev, I., Schmid, C.: Actions in context. In: CVPR (2009)
12. Lin, Z., Jiang, Z., Davis, L.S.: Recognizing actions by shape-motion prototype trees. In: ICCV (2009)
13. Liu, J., Shah, M.: Learning human actions via information maximization. In: CVPR (2008)
14. Gorelick, L., Blank, M., Shechtman, E., Irani, M., Basri, R.: Actions as space-time shapes. IEEE TPAMI 29, 2247–2253 (2007)
15. Brendel, W., Todorovic, S.: Video object segmentation by tracking regions. In: ICCV (2009)
16. Frome, A., Singer, Y., Sha, F., Malik, J.: Learning globally-consistent local distance functions for shape-based image retrieval and classification. In: ICCV (2007)
17. Gilad-Bachrach, R., Navot, A., Tishby, N.: Margin based feature selection – theory and algorithms. In: ICML, vol. 43 (2004)
18. Vedaldi, A., Soatto, S.: Quick shift and kernel methods for mode seeking. In: Forsyth, D., Torr, P., Zisserman, A. (eds.) ECCV 2008, Part IV. LNCS, vol. 5305, pp. 705–718. Springer, Heidelberg (2008)
19. Schueldt, C., Laptev, I., Caputo, B.: Recognizing human actions: A local SVM approach. In: ICPR (2004)