Joint Recognition and Segmentation of Actions via Probabilistic Integration of Spatio-Temporal Fisher Vectors

Johanna Carvajal, Chris McCool, Brian Lovell, Conrad Sanderson

University of Queensland, Brisbane, Australia
Queensland University of Technology, Brisbane, Australia
NICTA, Australia
Data61, CSIRO, Australia

Abstract. We propose a hierarchical approach to multi-action recognition that performs joint classification and segmentation. A given video (containing several consecutive actions) is processed via a sequence of overlapping temporal windows. Each frame in a temporal window is represented through selective low-level spatio-temporal features which efficiently capture relevant local dynamics. Features from each window are represented as a Fisher vector, which captures first and second order statistics. Instead of directly classifying each Fisher vector, it is converted into a vector of class probabilities. The final classification decision for each frame is then obtained by integrating the class probabilities at the frame level, which exploits the overlapping of the temporal windows. Experiments were performed on two datasets: s-KTH (a stitched version of the KTH dataset to simulate multi-actions), and the challenging CMU-MMAC dataset. On s-KTH, the proposed approach achieves an accuracy of 85.0%, significantly outperforming two recent approaches based on GMMs and HMMs which obtained 78.3% and 71.2%, respectively. On CMU-MMAC, the proposed approach achieves an accuracy of 40.9%, outperforming the GMM and HMM approaches which obtained 33.7% and 38.4%, respectively. Furthermore, the proposed system is on average 40 times faster than the GMM based approach.

1 Introduction

Research on human action recognition can be divided into two areas: **(i)** single-action recognition, and **(ii)** multi-action recognition. In most computer vision literature, action recognition approaches have concentrated on single actions, where each video to be classified contains only one action. However, when observing realistic human behaviour in natural settings, the fundamental problem is segmenting and recognising actions from a *multi-action* sequence [5]. It is challenging due to the high variability of appearances, shapes, possible occlusions, large variability in the temporal scale and periodicity of human actions, the complexity of articulated motion, the exponential nature of all possible movement combinations, as well as the prevalence of irrelevant background [19,34].

Hoai et al. [19] address joint segmentation and classification by classifying temporal regions using a multi-class Support Vector Machine (SVM) and performing segmentation using dynamic programming. A similar approach is presented in [9], where the temporal relationship between actions is considered. Borzeshi et al. [4] proposed the use of hidden Markov models (HMMs) with irregular observations (termed HMM-MIO) to perform multi-action recognition. More recently, Carvajal et al. [8] proposed to model each action using a Gaussian mixture model (GMM) approach where classification decisions are made using overlapping temporal windows. A drawback of [4,19,9] is that they have a large number of parameters to optimise. Furthermore, [4] requires an extra stage to reduce dimensionality due to use of very high dimensional feature vectors, while [19,9] require fully labelled annotations for training. One downside of [8] is that for each action and scenario, a model with a large number of Gaussians is required, making the system computationally expensive.

Typically, the aforementioned approaches used for the multi-action recognition task can be classified as either generative or discriminative models. The approaches presented in [4,8] are generative models, while those presented in [9,19] are discriminative models. Generative and discriminative models have complementary strengths. Generative models can easily deal with variable length sequences and missing data, while also being easier to design and implement [20,22]. In contrast, discriminative models often achieve superior classification and generalisation performance [20,22]. An ideal recognition system would hence combine these two separate but complementary approaches.

The Fisher vector (FV) approach [20,11,29] allows for the fusion of both generative and discriminative models. In contrast to the popular Bag of Words (BoW) approach [39] which describes images by histograms of visual words, the FV approach describes images by deviations from a probabilistic visual vocabulary model. The resulting vectors can then be used by an SVM for final classification. Recently, FV has been successfully applied to the single-action recognition problem [26,38].

A reliable low-level feature descriptor is a crucial stage for the success of an action recognition system. One popular descriptor for action recognition is Spatio-Temporal Interest Points (STIPs) [21]. However, STIP based descriptors have several drawbacks [6,16]: they are computationally expensive, unstable, imprecise and can result in unnecessarily sparse detections. See Fig. 1 for a demonstration of STIP based detection. Other feature extraction techniques used for action recognition include gradients [8] and optical flow [1,16]. Each pixel in the gradient image helps extract relevant information, eg. edges (see Fig. 1). Since the task of action recognition is based on a sequence of frames, optical flow provides an efficient way of capturing the local dynamics [16].
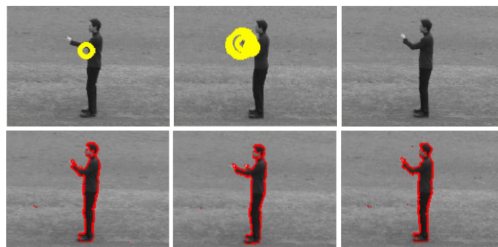

Fig. 1. Top row: feature extraction based on Spatio-Temporal Interest Points (STIPs) is often unstable, imprecise and overly sparse. Bottom row: interest pixels (marked in red) obtained using magnitude of gradient.

Contributions. In this paper we propose a novel hierarchical system to perform multi-action segmentation and recognition. A given video is processed as a sequence of overlapping temporal windows. Each frame in a temporal window is represented through selective low-level spatio-temporal features, based on a combination of gradients with optical flow. Interesting features from each temporal window are then pooled and processed to form a Fisher vector. Instead of directly classifying each Fisher vector, a multi-class SVM is employed to generate a vector of probabilities, with one probability per class. The final classification decision (action label) for each frame is then obtained by integrating the class probabilities at the frame level, which exploits the overlapping of the temporal windows. The proposed system hence combines the benefits of generative and discriminative models.

To the best of our knowledge, the combination of probabilistic integration with Fisher vectors is novel for the *multi-action* segmentation and recognition problem. In contrast to [4,19,9], the proposed system requires fewer parameters to be optimised. We also avoid the need for a custom dynamic programming definition as in [19,9]. Lastly, unlike the GMM approach proposed in [8] the proposed method requires only one GMM for all actions, making it considerably more efficient.

The paper is continued as follows. An overview of prior work on feature descriptors as well as single- and multi-action recognition is in Section 2. We then describe the proposed method in Section 3. A comparative evaluation against previous multi-action recognition methods is presented in Section 4. The main findings and potential areas for future work are given in Section 5.

2 Related Work

In this section, we present an overview of notable methods for action recognition. We first describe several popular descriptors used for action recognition. This is followed by an overview of techniques for single-action recognition. We then present a summary of approaches for multi-action segmentation and recognition.

2.1 Low-Level Descriptors

Several descriptors have been used to represent human actions. Among them, we can find descriptors based on Spatio-Temporal Interest Points (STIPs) [21], gradients [8], and optical flow [1]. The STIP based detector finds interest points and is a space-time extension of the Harris and Förstner interest point detector [15,18]. STIP descriptors have led to relatively good action recognition performance [42]. However, STIP based descriptors have several drawbacks: **(i)** they focus on local spatio-temporal information instead of global motion, **(ii)** they can be unstable and imprecise (varying number of STIP detections) leading to low repeatability, **(iii)** they are computationally expensive, **(iv)** produce sparse detections, and **(v)** are high dimensional with default descriptors of 162 dimensions.

The Scale Invariant Feature Transform (SIFT) was first described in [24]. Key locations are chosen at maxima and minima of a difference of Gaussian function applied in scale space. Orientation is assigned to each key point to achieve invariance to image rotation. Finally, the key-point descriptor is created taking a neighbourhood around

the key point. SIFT features holds positive attributes such as local invariance to transformation, rotation, and scale. However, its applicability to large datasets or real-time applications is limited due to it its relatively high dimensionality and high computational demands [14].

A video feature descriptor using dense trajectories and motion boundary descriptors is proposed in [38]. The dense representation allows for good coverage of foreground motion as well as of the surrounding context. This descriptor has become popular for the single-action recognition task despite of being relatively computationally expensive and high dimensional, with the final number of dimensions being 426.

Gradients have been used as a relatively simple yet effective video representation [8]. Each pixel in the gradient image helps extract relevant information, eg. edges of a subject. Gradients can be computed at every spatio-temporal location (x, y, t) in any direction in a video. Lastly, since the task of action recognition is based on an ordered sequence of frames, optical flow can be used to provide an efficient way of capturing local dynamics and motion patterns in a scene [16].

2.2 Single-Action Recognition

Hidden Markov models (HMMs) have been used in conjunction with shape-context features to recognise single actions [25]. The shape-context features are extracted using the image contours and dividing the region where the person is into uniform tiles. For each tile a feature vector is generated. The discrete cosine transform is used in order to minimise redundancy and to compress the data. Continuous HMM with mixed Gaussian output probability is employed with a simple left to right topology.

Gaussian Mixture Models (GMMs) have also been explored for the single-action detection and classification. For the approach presented in [23], each action is represented by a combination of GMMs. Each action is modelled by two sets of feature attributes. The first set represents the change of body size, while the second represents the speed of the action. Features with high correlations for describing actions are grouped into the same Category Feature Vector (CFV). All CFVs related to the same category are then modelled using a GMM. A Confident-Frame-based Recognising algorithm is used for recognition, where the video frames which have high confidence are used as a specialised model for classifying the rest of the video frames.

A video sequence can be also represented as a Bag of Words (BoW) [39]. This approach and its variants are among the most popular approaches for action recognition [38,39]. The standard approach consists of four main steps: low-level feature extraction, offline codebook generation, feature encoding and pooling, and normalisation. The offline codebook, also known as visual dictionary, is constructed using k-means clustering of a large training dataset. Visual words are then defined as the centers of the clusters, with the size of visual dictionary equal to the number of the clusters. To encode low-level features, each feature vector is assigned to its closest dictionary word using Euclidean distance. A given video is then represented as a frequency histogram over the visual words. Normalised histograms can then be used by linear classifiers.

The Fisher vector (FV) representation can be thought as an evolution of the BoW representation, encoding additional information [11,38]. Rather than encoding the frequency of the descriptors, FV encodes the deviations from a probabilistic version of

the visual dictionary (which is typically a GMM). This is done by computing the gradient of the sample log-likelihood with respect the parameters of the dictionary model. Since more information is extracted, a smaller visual dictionary size can be used than for BoW, in order to achieve the same or better performance. Finally, L_2 and power normalisation is often used to improve performance in combination with linear classifiers [29].

FV has been used for a wide range of applications in the computer vision community, such as image retrieval [13], as well as image representation and classification [11,29]. Recently, FV has been also successfully applied to the single-action recognition problem [26,38]. Descriptors based on motion boundary histograms and Scale-Invariant Feature Transform (SIFT) are used in conjunction with FV in [26]. The dimensionality of each descriptor is reduced to 64 dimensions via Principal Component Analysis (PCA). A similar approach is presented in [38], where the SIFT descriptors are replaced by histograms of gradients, optical flow, and trajectories. To combine various descriptor types, FVs derived from each descriptor type are concatenated.

Note that both the FV and BoW approaches build on top of low-level features, which typically describe small image areas. The low-level features may or may not have robustness to scale & rotations (see Section 2.1). However, even if the low-level features are not directly robust, it has been previously shown that the BoW approach (and by extension the FV approach) can still exhibit robustness to moderate variations [7,30,32]. The reasoning is summarised as follows. When an object undergoes moderate scale changes or rotations, the dominant effect on many small parts is translation. Both FV and BoW do not have rigid constraints on the location of object parts, and also do not have rigid constraints on spatial relations between parts. This lack of constraints allows for movement (translation) and dispersion of small image areas, which in turn leads to a degree of overall robustness of FV and BoW.

The action recognition problem can be also solved by decomposing actions into sub-actions or atoms [40,41]. In the former, motion atoms are obtained using a discriminative clustering method. These atoms are basic units used to construct motion phrase with a longer scale. To this end, a bottom-up phrase construction algorithm and a greedy selection method are used. Motion phrase is composed of multiple atomic motion units. The latter work presents a latent hierarchical model. This hierarchical model has a tree structure, where each node represents a sub-action. Two latent variables are used to represent the starting and ending time points of each sub-action.

2.3 Multi-Action Recognition

In contrast to single action recognition, relatively little work exists on multi-action recognition. Multi-action recognition, in our context, consists of segmenting and recognising separate actions from an image sequence, where one person performs a sequence of such actions [34]. The process for segmenting and recognising multiple actions in a video can be solved either as two independent problems or as a joint problem.

Two methods [4,19] have been applied to realistic multi-action datasets. Hoai et al. [19] deal with the dual problem of human action segmentation and classification. The recognition model is trained using labelled data with a multi-class SVM. Once the model for all actions has been obtained, the video segmentation and recognition is done

using dynamic programming, maximising the SVM score of the winning class while suppressing those of the non-maximum classes. The feature mapping depends on the dataset employed, and includes trajectories, features extracted from binary masks and STIPs.

Although the method proposed in [19] is promising, it has several drawbacks. One drawback is the requirement of fully labelled annotations for training. Furthermore, it suffers from the limitations of dynamic programming where writing the code that evaluates sub-problems in the most efficient order is often nontrivial [37]. Also, the binary masks are not always available and the STIP descriptors have deficiencies as mentioned in Section 2.1. This method also requires an extensive search for optimal parameters.

An approach termed Hidden Markov Model for Multiple, Irregular Observations (HMM-MIO) [4] has also been proposed for the multi-action recognition task. HMM-MIO jointly segments and classifies observations which are irregular in time and space, and are characterised by high dimensionality. The high dimensionality is reduced by probabilistic PCA. Moreover, HMM-MIO deals with heavy tails and outliers exhibited by empirical distributions by modelling the observation densities with the long-tailed Student's t distribution. HMM-MIO requires the search of the following four optimal parameters: **(i)** the resulting reduced dimension, **(ii)** the number of components in each observation mixture, **(iii)** the degree of the t-distribution, and **(iv)** the number of cells (or regions) used to deal with space irregularity. As feature descriptors, HMM-MIO extracts STIPs, with the default 162-dimensional descriptor. HMM-MIO hence suffers from the drawback of a large search of optimal parameters and the use of STIP descriptors.

A recent approach to multi-action recognition via stochastic modelling of optical flow and gradients is presented in [8]. Each action is modelled using a GMM. Action segmentation is achieved by integrating classification decisions on overlapping temporal windows. In each window, the average log-likelihood is obtained for each model using a large number of components (1024). This approach outperforms HMM-MIO, but is limited by being a purely generative model. Moreover, due to the large number of components per action model, the calculation of the likelihoods using all action models is computationally very expensive.

3 Proposed Method

The proposed system has a hierarchical nature, stemming from progressive reduction and transformation of information, starting at the pixel level. The system is comprised of four main components:

(i) Division of a given video into overlapping multi-frame temporal windows, followed by extracting interesting low-level spatio-temporal features from each frame in each window.

(ii) Pooling of the interesting features from each temporal window to generate a sequence of Fisher vectors.

(iii) Conversion of each Fisher vector into a vector of class probabilities with the aid of a multi-class SVM.

(iv) Integration of the class probabilities at the frame level, leading to the final classification decision (action label) for each frame.

Each of the components is explained in more detail in the following subsections.

3.1 Overlapping and Selective Feature Extraction

A video $\mathcal{V} = (\boldsymbol{I}_t)_{t=1}^T$ is an ordered set of T frames. We divide $\mathcal{V}$ into a set of overlapping temporal windows $(\mathcal{W}_s)_{s=1}^S$, with each window having a length of L frames. To achieve overlapping, the start of each window is one frame after the start of the preceding window. Each temporal window is hence defined as a set of frame identifiers: $\mathcal{W}_s = (t_{\text{start}}, \ldots, t_{\text{start}-1+L})$.

Each frame $\boldsymbol{I}_t \in \mathbb{R}^{r \times c}$ can be represented by a set of feature vectors $F_t = \{\boldsymbol{f}_p\}_{p=1}^{N_t}$ (with $N_t < r \cdot c$) corresponding to interesting pixels. Following [8], we first extract the following $D = 14$ dimensional feature vector for each pixel in a given frame t:

$$\boldsymbol{f} = [\, x,\ y,\ \boldsymbol{g},\ \boldsymbol{o} \,] \tag{1}$$

where x and y are the pixel coordinates, while $\boldsymbol{g}$ and $\boldsymbol{o}$ are defined as:

$$\boldsymbol{g} = \left[\, |J_x|,\ |J_y|,\ |J_{yy}|,\ |J_{xx}|,\ \sqrt{J_x^2 + J_y^2},\ \operatorname{atan}\frac{|J_y|}{|J_x|} \,\right] \tag{2}$$

$$\boldsymbol{o} = \left[\, u,\ v,\ \frac{\partial u}{\partial t},\ \frac{\partial v}{\partial t},\ \left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y}\right),\ \left(\frac{\partial v}{\partial x} - \frac{\partial u}{\partial y}\right) \,\right] \tag{3}$$

The first four gradient-based features in Eq. (2) represent the first and second order intensity gradients at pixel location (x, y). The last two gradient features represent gradient magnitude and gradient orientation. The optical flow based features in Eq. (3) represent in order: the horizontal and vertical components of the flow vector, the first order derivatives with respect to time, the divergence and vorticity of optical flow [1].

Typically only a subset of the pixels in a frame correspond to the object of interest. As such, we are only interested in pixels with a gradient magnitude greater than a threshold τ [17]. We discard feature vectors from locations with a small magnitude. In other words, only feature vectors corresponding to interesting pixels are kept. This typically results in a variable number of feature vectors per frame. See the bottom part in Fig. 1 for an example of the retained pixels.

3.2 Representing Windows as Fisher Vectors

Given a set of feature vectors, the Fisher Vector approach encodes the deviations from a probabilistic visual dictionary, which is typically a diagonal GMM. The parameters of a GMM with K components can be expressed as $\{w_k, \boldsymbol{\mu}_k, \boldsymbol{\sigma}_k\}_{k=1}^{K}$, where, w_k is the weight, $\boldsymbol{\mu}_k$ is the mean vector, and $\boldsymbol{\sigma}_k$ is the diagonal covariance matrix for the k-th Gaussian. The parameters are learned using the Expectation Maximisation algorithm [3] on training data.

For each temporal window $\mathcal{W}_s$, the feature vectors are pooled into set X containing $N = \sum_{t \in \mathcal{W}_s} N_t$ vectors. The deviations from the GMM are then accumulated using [29]:

$$\mathcal{G}_{\boldsymbol{\mu}_k}^{X} = \frac{1}{N\sqrt{w_k}} \sum\nolimits_{n=1}^{N} \gamma_n(k) \left(\frac{\boldsymbol{f}_n - \boldsymbol{\mu}_k}{\boldsymbol{\sigma}_k} \right) \tag{4}$$

$$\mathcal{G}_{\boldsymbol{\sigma}_k}^{X} = \frac{1}{N\sqrt{2w_k}} \sum\nolimits_{n=1}^{N} \gamma_n(k) \left[\frac{(\boldsymbol{f}_n - \boldsymbol{\mu}_k)^2}{\boldsymbol{\sigma}_k^2} - 1 \right] \tag{5}$$

where vector division indicates element-wise division and $\gamma_n(k)$ is the posterior probability of $\boldsymbol{f}_n$ for the k-th component:

$$\gamma_n(k) = \frac{w_k \mathcal{N}(\boldsymbol{f}_n | \boldsymbol{\mu}_k, \boldsymbol{\sigma}_k)}{\sum_{i=1}^{K} w_i \mathcal{N}(\boldsymbol{f}_n | \boldsymbol{\mu}_i, \boldsymbol{\sigma}_i)} \tag{6}$$

The Fisher vector for window $\mathcal{W}_s$ is represented as the concatenation of $\mathcal{G}_{\boldsymbol{\mu}_k}^{X}$ and $\mathcal{G}_{\boldsymbol{\sigma}_k}^{X}$ (for $k = 1, \ldots, K$) into vector $\boldsymbol{\Phi}_s$. As $\mathcal{G}_{\boldsymbol{\mu}_k}^{X}$ and $\mathcal{G}_{\boldsymbol{\sigma}_k}^{X}$ are D-dimensional, $\boldsymbol{\Phi}_s$ has the dimensionality of $2DK$. Note that we have omitted the deviations for the weights as they add little information [29].

3.3 Generation of Probability Vectors

For each Fisher vector we generate a vector of probabilities, with one probability per action class. First, a multi-class SVM [10] is used to predict class labels, outputting a set of raw scores. The scores are then transformed into a probability distribution over classes by applying Platt scaling [28]. The final probability vector derived from Fisher vector $\boldsymbol{\Phi}_s$ is expressed as:

$$\boldsymbol{q}_s = [\, P(l = 1 | \boldsymbol{\Phi}_s), \; \cdots, \; P(l = A | \boldsymbol{\Phi}_s) \,] \tag{7}$$

where l indicates an action class label and A is the number of action classes. The parameters for the multi-class SVM are learned using Fisher vectors obtained from pre-segmented actions in training data.

3.4 Integrating Probability Vectors to Label Frames

As the temporal windows are overlapping, each frame is present in several temporal windows. We exploit the overlapping to integrate the class probabilities at the frame level. The total contribution of the probability vectors to each frame t is calculated by:

$$\boldsymbol{Q}_t = \sum\nolimits_{s=1}^{S} \mathbf{1}_{\mathcal{W}_s}(t) \cdot \boldsymbol{q}_s \tag{8}$$

where $\mathbf{1}_{\mathcal{W}_s}(t)$ is an indicator function, resulting in 1 if $t \in \mathcal{W}_s$, and 0 otherwise. The estimated action label for frame t is then calculated as:

$$\widehat{l}_t = \arg\max_{l=1,...,A} \boldsymbol{Q}_t^{[l]} \qquad (9)$$

where $\boldsymbol{Q}_t^{[l]}$ indicates the l-th element of $\boldsymbol{Q}_t$.

4 Experiments

We evaluated our proposed method for joint action segmentation and recognition on two datasets: **(i)** a stitched version of the KTH dataset [33], and **(ii)** the challenging Carnegie Mellon University Multi-Modal Activity Dataset (CMU-MMAC) [12]. The results are reported in terms of frame-level accuracy as the ratio between the number of matched frames over the total number of frames.

4.1 Datasets

The **s-KTH** (stitched KTH) dataset is obtained by simply concatenating existing single-action instances into sequences [4]. The KTH dataset contains 25 subjects performing 6 types of human actions and 4 scenarios. The scenarios vary from indoor, outdoor, scale variations, and different clothes. Each original video of the KTH dataset [33] contains an individual performing the same action. This action is performed four times and each subdivision or action-instance (in terms of start-frame and end-frame) is provided as part of the dataset. This dataset contains 2391 action-instances, with a length between 1 and 14 seconds [2]. The image size is 160×120 pixels, and temporal resolution is 25 frames per second.

The action-instances (each video contains four instances of the action) were picked randomly, alternating between the two groups of {boxing, hand-waving, hand-clapping} and {walking, jogging, running} to accentuate action boundaries. See Fig. 2 for an example. The dataset was divided into two sets as in [4,8]: one for training and one for testing. In total, 64 and 36 multi-action videos were used for training and testing, respectively. We used 3-fold cross-validation.

The **CMU-MMAC** dataset is considerably more challenging as it contains realistic multi-action videos [12]. A kitchen was built to record subjects preparing and cooking food according to five recipes. This dataset has occlusions, a cluttered background, and many distractors such as objects being deliberately moved. For our experiments we have used the same subset as per [4], which contains 12 subjects making brownies. The subjects were asked to make brownies in a natural way (no instructions were given). Each subject making the brownie is partially seen, as shown in Fig. 3.

The videos have a high resolution and are longer than in s-KTH. The image size is 1024×768 pixels, and temporal resolution is 30 frames per second. The average duration of a video is approximately 15,000 frames and the average length of an action instance is approximately 230 frames (7.7s), with a minimum length of 3 frames (0.1s) and a maximum length of 3,269 frames (108s) [4]. The dataset was annotated using 14 labels, including the actions *close*, *crack*, *open*, *pour*, *put*, *read*, *spray*, *stir*, *switch-on*,

Fig. 2. Example of a multi-action sequence in the stitched version of the KTH dataset (s-KTH): boxing, jogging, hand clapping, running, hand waving and walking.

Fig. 3. Example of a challenging multi-action sequence in the CMU-MMAC kitchen dataset: crack, read, stir, and switch-on.

take, *twist-off*, *twist-on*, *walk*, and the remaining actions (eg. frames in between two distinct actions) were grouped under the label *none* [36]. We used 12-fold cross-validation, using one subject for testing on a rotating basis. The experiments were implemented with the aid of the Armadillo C++ library [31].

4.2 Setup

All videos were converted into gray-scale. Additionally, the videos from the CMU-MMAC dataset were re-scaled to 128×96 to reduce computational requirements. Based on [8] and preliminary experiments on both datasets, we used $\tau = 40$, where τ is the threshold used for selection of interesting low-level feature vectors (Section 3.1) . Although the interesting feature vectors are calculated in all frames, we only use the feature vectors extracted from every second frame in order to speed up processing.

Parameters for the visual vocabulary (GMM) were learned using a large set of descriptors obtained from training videos using the iterative Expectation-Maximisation algorithm [3]. Specifically, we randomly sampled 100,000 feature vectors for each action and then pooled all the resultant feature vectors from all actions for training. Experiments were performed with three separate GMMs with varying number of components: $K = \{64, 128, 256\}$. We have not evaluated larger values of K due to increased computational complexity and hence the exorbitant amount of time required to process the large CMU-MMAC dataset.

To learn the parameters of the multi-class SVM, we used video segments containing single actions. For s-KTH this process is straightforward as the videos have been previously segmented. The CMU-MMAC dataset contains continuous multi-actions. For this reason, to train our system we obtain one Fisher vector per action in each video, using the low-level feature vectors belonging to that specific action.

4.3 Effect of Window Length and Dictionary Size

We have evaluated the performance of two variants of the proposed system: **(1)** probabilistic integration with Fisher vectors (**PI-FV**), and **(2)** probabilistic integration with

BoW histograms (**PI-BoW**), where the Fisher vector representation is replaced with BoW representation. We start our experiments by studying the influence of the segment length L, expressed in terms of seconds. The results are reported in Figs. 4 and 5, in terms of average accuracy over the folds.

Using the PI-FV variant (Fig. 4), we found that using $L = 1s$ and $K = 256$ leads to the best performance on the s-KTH dataset. For the CMU-MMAC dataset, the best performance is obtained with $L = 2.5s$ and $K = 64$. Note that using larger values of K (ie., 128 and 256) leads to worse performance. We attribute this to the large variability of appearance in the dataset, where the training data may not be a good representative of the test data. Consequently, using a large value of K may lead to overfitting to the training data.

The optimal segment length for each dataset is different. We attribute this to the s-KTH dataset containing short videos whose duration is between $1s$ and $7s$, while CMU-MMAC has a large range of action durations between $0.1s$ and $108s$. While the optimal values of L and K differ across the datasets, the results also show that relatively good overall performance across both datasets can be obtained with $L = 1s$ and $K = 64$.

The results for the PI-BoW variant are shown in Fig. 5. The best performance for the PI-BoW variant on the s-KTH dataset is obtained using $L = 1s$ and $K = 256$, while on the CMU-MMAC dataset it is obtained with $L = 2.5s$ and $K = 256$. These are the same values of L and K as for the PI-FV variant. However, the performance of the PI-BoW variant is consistently worse than the PI-FV variant on both datasets. This can be attributed to the better representation power of FV, as described in Section 2. Note that the visual dictionary size K for BoW is usually higher in order to achieve performance similar to FV. However, due to the large size of the CMU-MMAC dataset, and for direct comparison purposes, we have used the same range of K values throughout the experiments.

4.4 Confusion Matrices

Figs. 6 and 7 show the confusion matrices for the PI-FV and PI-BoW variants on the CMU-MMAC dataset. The confusion matrices show that in 50% of the cases (actions), the PI-BoW variant is unable to recognise the correct action. Furthermore, the PI-FV variant on average obtains better action segmentation than PI-BoW.

For five actions (*crack*, *open*, *read*, *spray*, *twist-on*), PI-FV has accuracies of 0.5% or lower. Action *crack* implies crack and pour eggs into a bowl, but it's annotated only as *crack*, leading to confusion between *crack* and *pour*. We suspect that actions *read* and *spray* are poorly modelled due to lack of training data; they are performed by a reduced number of subjects. Action *twist-on* is confused with *twist-off* which are essentially the same action.
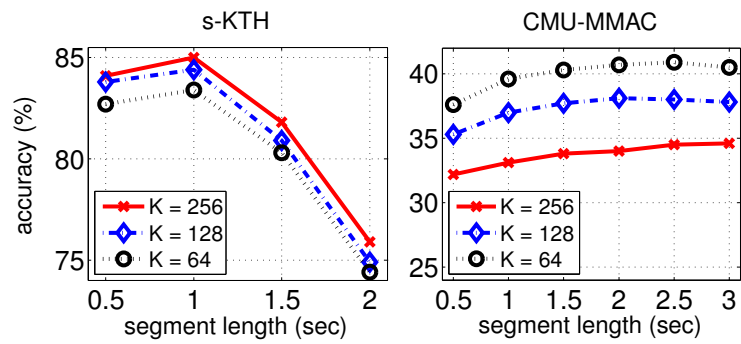
Fig. 4. Performance of the proposed **PI-FV** approach for varying the segment length on the s-KTH and CMU-MMAC datasets, in terms of average frame-level accuracy over the folds.
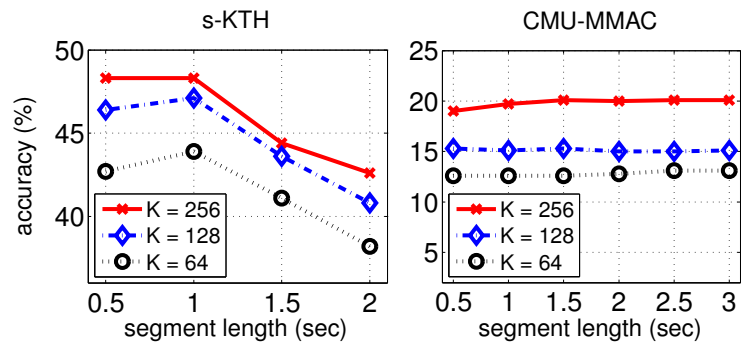

Fig. 5. As per Fig. 4, but showing the performance of the **PI-BoW** variant (where the Fisher vector representation is replaced with BoW representation).
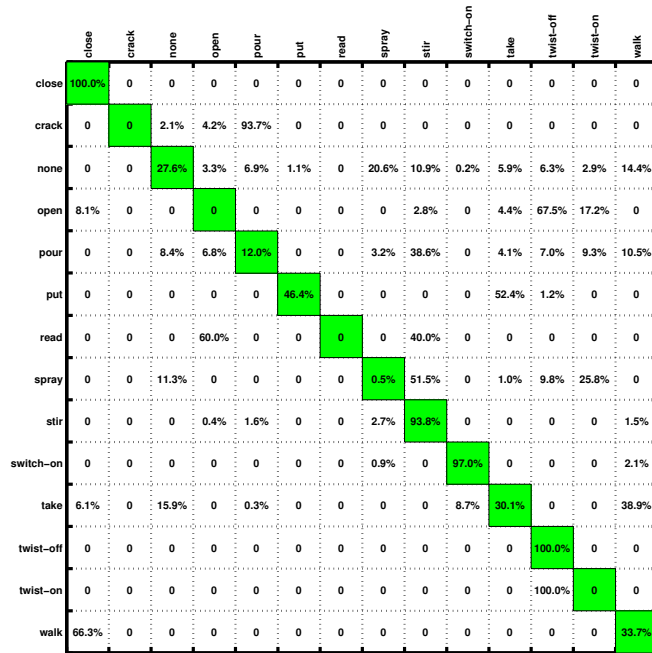
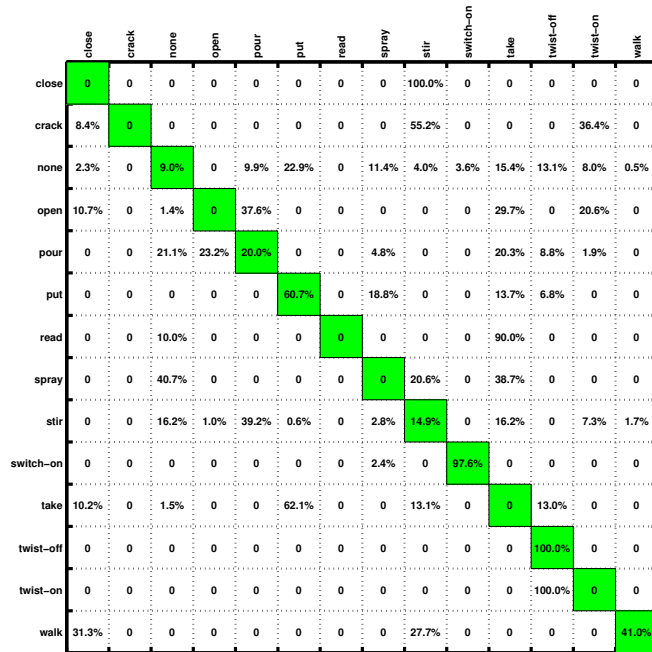**Fig. 6.** Confusion matrix for the PI-FV variant on the CMU-MMAC dataset.

	close	crack	none	open	pour	put	read	spray	stir	switch-on	take	twist-off	twist-on	walk
close	100.0%	0	0	0	0	0	0	0	0	0	0	0	0	0
crack	0	0	2.1%	4.2%	93.7%	0	0	0	0	0	0	0	0	0
none	0	0	27.6%	3.3%	6.9%	1.1%	0	20.6%	10.9%	0.2%	5.9%	6.3%	2.9%	14.4%
open	8.1%	0	0	0	0	0	0	0	2.8%	0	4.4%	67.5%	17.2%	0
pour	0	0	8.4%	6.8%	12.0%	0	0	3.2%	38.6%	0	4.1%	7.0%	9.3%	10.5%
put	0	0	0	0	0	46.4%	0	0	0	0	52.4%	1.2%	0	0
read	0	0	0	60.0%	0	0	0	0	40.0%	0	0	0	0	0
spray	0	0	11.3%	0	0	0	0	0.5%	51.5%	0	1.0%	9.8%	25.8%	0
stir	0	0	0	0.4%	1.6%	0	0	2.7%	93.8%	0	0	0	0	1.5%
switch-on	0	0	0	0	0	0	0	0.9%	0	97.0%	0	0	0	2.1%
take	6.1%	0	15.9%	0	0.3%	0	0	0	8.7%	0	30.1%	0	0	38.9%
twist-off	0	0	0	0	0	0	0	0	0	0	0	100.0%	0	0
twist-on	0	0	0	0	0	0	0	0	0	0	100.0%	0	0	0
walk	66.3%	0	0	0	0	0	0	0	0	0	0	0	0	33.7%

Fig. 7. As per Fig. 6, but using the PI-BoW variant.

	close	crack	none	open	pour	put	read	spray	stir	switch-on	take	twist-off	twist-on	walk
close	0	0	0	0	0	0	0	0	100.0%	0	0	0	0	0
crack	8.4%	0	0	0	0	0	0	0	55.2%	0	0	0	36.4%	0
none	2.3%	0	9.0%	0	9.9%	22.9%	0	11.4%	4.0%	3.6%	15.4%	13.1%	8.0%	0.5%
open	10.7%	0	1.4%	0	37.6%	0	0	0	0	0	29.7%	0	20.6%	0
pour	0	0	21.1%	23.2%	20.0%	0	0	4.8%	0	0	20.3%	8.8%	1.9%	0
put	0	0	0	0	0	60.7%	0	18.8%	0	0	13.7%	6.8%	0	0
read	0	0	10.0%	0	0	0	0	0	0	0	90.0%	0	0	0
spray	0	0	40.7%	0	0	0	0	0	20.6%	0	38.7%	0	0	0
stir	0	0	16.2%	1.0%	39.2%	0.6%	0	2.8%	14.9%	0	16.2%	0	7.3%	1.7%
switch-on	0	0	0	0	0	0	0	2.4%	0	97.6%	0	0	0	0
take	10.2%	0	1.5%	0	62.1%	0	0	0	13.1%	0	0	13.0%	0	0
twist-off	0	0	0	0	0	0	0	0	0	0	0	100.0%	0	0
twist-on	0	0	0	0	0	0	0	0	0	0	100.0%	0	0	0
walk	31.3%	0	0	0	0	0	0	0	27.7%	0	0	0	0	41.0%

Table 1. Comparison of the proposed methods (PI-FV and PI-BoW) against several recent approaches on the stitched version of the KTH dataset (s-KTH) and the challenging CMU-MMAC dataset.

Method	s-KTH	CMU-MMAC
HMM-MIO [4]	71.2%	38.4%
Stochastic Modelling [8]	78.3%	33.7%
PI-FV	**85.0**%	**40.9**%
PI-BoW	48.0%	20.1%

4.5 Comparison with GMM and HMM-MIO

We have compared the performance of the PI-FV and PI-BoW variants against the HMM-MIO [4] and stochastic modelling [8] approaches previously used for multi-action recognition. The comparative results are shown in Table 1.

The proposed PI-FV method obtains the highest accuracy of 85.0% and 40.9% for the s-KTH and CMU-MMAC datasets, respectively. In addition to higher accuracy, the proposed method has other advantages over previous techniques. There is just one global GMM (representing the visual vocabulary). This is in contrast to [8] which uses one GMM (with a large number of components) for each action, leading to high computational complexity. The HMM-MIO method in [4] requires the search for many optimal parameters (as described in Section 2.3), whereas the proposed method has just two parameters (L and K).

4.6 Wall-Clock Time

Lastly, we provide an analysis of the computational cost (in terms of wall-clock time) of our system and the stochastic modelling approach. The wall-clock time is measured under optimal configuration for each system, using a Linux machine with an Intel Core processor running at 2.83 GHz. On the s-KTH dataset, the stochastic modelling system takes on average 228.4 minutes to segment and recognise a multi-action video. In comparison, the proposed system takes 5.6 minutes, which is approximately 40 times faster.

5 Conclusions and Future Work

In this paper we have proposed a hierarchical approach to multi-action recognition that performs joint segmentation and classification in videos. Videos are processed through overlapping temporal windows. Each frame in a temporal window is represented using selective low-level spatio-temporal features which efficiently capture relevant local dynamics and do not suffer from the instability and imprecision exhibited by STIP descriptors [21]. Features from each window are represented as a Fisher vector, which captures the first and second order statistics. Rather than directly classifying each Fisher vector, it is converted into a vector of class probabilities. The final classification decision for each frame (action label) is then obtained by integrating the class probabilities at the frame level, which exploits the overlapping of the temporal windows. The proposed approach has a lower number of free parameters than previous methods which

use dynamic programming or HMMs [4]. It is also considerably less computationally demanding compared to modelling each action directly with a GMM [8].

Experiments were done on two datasets: s-KTH (a stitched version of the KTH dataset to simulate multi-actions), and the more challenging CMU-MMAC dataset (containing realistic multi-action videos of food preparation). On s-KTH, the proposed approach achieves an accuracy of 85.0%, considerably outperforming two recent approaches based on GMMs and HMMs which obtained 78.3% and 71.2%, respectively. On CMU-MMAC, the proposed approach achieves an accuracy of 40.9%, outperforming the GMM and HMM approaches which obtained 33.7% and 38.4%, respectively. Furthermore, the proposed system is on average 40 times faster than the GMM based approach.

Possible future areas of exploration include the use of the fast Fisher vector variant proposed in [35], where for each sample the deviations for only one Gaussian are calculated. This can deliver a large speed up in computation, at the cost of a small drop in accuracy [27].

References

1. S. Ali and M. Shah. Human action recognition in videos using kinematic features and multiple instance learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(2):288–303, 2010.
2. M. Baccouche, F. Mamalet, C. Wolf, C. Garcia, and A. Baskurt. Sequential deep learning for human action recognition. In *Human Behavior Understanding*, volume 7065 of *Lecture Notes in Computer Science*, pages 29–39. 2011.
3. C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.
4. E. Borzeshi, O. Perez Concha, R. Xu, and M. Piccardi. Joint action segmentation and classification by an extended hidden Markov model. *IEEE Signal Processing Letters*, 20(12):1207–1210, 2013.
5. D. Buchsbaum, K. R. Canini, and T. Griffiths. Segmenting and recognizing human action using low-level video features. *Annual Conference of the Cognitive Science Society*, 2011.
6. L. Cao, Y. Tian, Z. Liu, B. Yao, Z. Zhang, and T. Huang. Action detection using multiple spatial-temporal interest point features. In *International Conference on Multimedia and Expo (ICME)*, pages 340–345, 2010.
7. F. Cardinaux, C. Sanderson, and S. Bengio. User authentication via adapted statistical models of face images. *IEEE Transactions on Signal Processing*, 54(1):361–373, 2006.
8. J. Carvajal, C. Sanderson, C. McCool, and B. C. Lovell. Multi-action recognition via stochastic modelling of optical flow and gradients. In *Workshop on Machine Learning for Sensory Data Analysis (MLSDA)*, pages 19–24, 2014.
9. Y. Cheng, Q. Fan, S. Pankanti, and A. Choudhary. Temporal sequence modeling for video event detection. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2235–2242, 2014.
10. K. Crammer and Y. Singer. On the algorithmic implementation of multiclass kernel-based vector machines. *Journal of Machine Learning Research*, 2:265–292, 2001.

11. G. Csurka and F. Perronnin. Fisher vectors: Beyond bag-of-visual-words image representations. In *Computer Vision, Imaging and Computer Graphics. Theory and Applications*, volume 229, pages 28–42. 2011.

12. F. De la Torre, J. K. Hodgins, J. Montano, and S. Valcarcel. Detailed human data acquisition of kitchen activities: the CMU-multimodal activity database (CMU-MMAC). In *CHI Workshop on Developing Shared Home Behavior Datasets to Advance HCI and Ubiquitous Computing Research*, 2009.

13. M. Douze, A. Ramisa, and C. Schmid. Combining attributes and Fisher vectors for efficient image retrieval. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 745–752, 2011.

14. M. El-Gayar, H. Soliman, and N. Meky. A comparative study of image low level feature extraction algorithms. *Egyptian Informatics Journal*, 14(2), 2013.

15. M. A. Föstner and E. Gülch. A fast operator for detection and precise location of distinct points, corners and centers of circular features. In *ISPRS Intercommission Workshop*, 1987.

16. O. K. Gross, Y. Gurovich, T. Hassner, and L. Wolf. Motion interchange patterns for action recognition in unconstrained videos. In *European Conference on Computer Vision (ECCV)*, 2012.

17. K. Guo, P. Ishwar, and J. Konrad. Action recognition from video using feature covariance matrices. *IEEE Transactions on Image Processing*, 22(6):2479–2494, 2013.

18. C. Harris and M. Stephens. A combined corner and edge detector. In *Alvey Vision Conference*, pages 147–151, 1988.

19. M. Hoai, Z.-Z. Lan, and F. De la Torre. Joint segmentation and classification of human actions in video. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3265–3272, 2011.

20. T. Jaakkola and D. Haussler. Exploiting generative models in discriminative classifiers. In *Advances in Neural Information Processing Systems 11*, pages 487–493, 1998.

21. I. Laptev. On space-time interest points. *International Journal of Computer Vision*, 64(2-3):107–123, 2005.

22. J. Lasserre and C. M. Bishop. Generative or discriminative? Getting the best of both worlds. In J. Bernardo, M. Bayarri, J. Berger, A. Dawid, D. Heckerman, A. Smith, and M. West, editors, *Bayesian Statistics*, volume 8, pages 3–24. Oxford University Press, 2007.

23. W. Lin, M.-T. Sun, R. Poovandran, and Z. Zhang. Human activity recognition for video surveillance. In *IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 2737–2740, 2008.

24. D. Lowe. Object recognition from local scale-invariant features. In *International Conference on Computer Vision (CVPR)*, volume 2, pages 1150–1157, 1999.

25. M. A. Mendoza and N. Pérez de la Blanca. HMM-based action recognition using contour histograms. In *Pattern Recognition and Image Analysis*, volume 4477 of *Lecture Notes in Computer Science*, pages 394–401. 2007.

26. D. Oneata, J. Verbeek, and C. Schmid. Action and event recognition with Fisher vectors on a compact feature set. In *International Conference on Computer Vision (ICCV)*, pages 1817–1824, 2013.

27. O. M. Parkhi, K. Simonyan, A. Vedaldi, and A. Zisserman. A compact and discriminative face track descriptor. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014.

28. J. C. Platt. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In *Advances in Large Margin Classifiers*, pages 61–74, 1999.

29. J. Sánchez, F. Perronnin, T. Mensink, and J. Verbeek. Image classification with the Fisher vector: Theory and practice. *International Journal of Computer Vision*, 105(3):222–245, 2013.

30. C. Sanderson, S. Bengio, and Y. Gao. On transforming statistical models for non-frontal face verification. *Pattern Recognition*, 39(2):288–302, 2006.
31. C. Sanderson and R. Curtin. Armadillo: a template-based C++ library for linear algebra. *Journal of Open Source Software*, 1:26, 2016.
32. C. Sanderson and B. Lovell. Multi-region probabilistic histograms for robust and scalable identity inference. *Lecture Notes in Computer Science (LNCS)*, 5558:199–208, 2009.
33. C. Schuldt, I. Laptev, and B. Caputo. Recognizing human actions: A local SVM approach. In *International Conference on Pattern Recognition (ICPR)*, volume 3, pages 32–36, 2004.
34. Q. Shi, L. Wang, L. Cheng, and A. Smola. Discriminative human action segmentation and recognition using semi-Markov model. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–8, 2008.
35. K. Simonyan, A. Vedaldi, and A. Zisserman. Deep Fisher networks for large-scale image classification. In C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 163–171. 2013.
36. E. H. Spriggs, F. D. L. Torre, and M. Hebert. Temporal segmentation and activity classification from first-person sensing. In *IEEE Workshop on Egocentric Vision, CVPR*, 2009.
37. D. B. Wagner. Dynamic programming. *The Mathematica Journal*, pages 42–51, 1995.
38. H. Wang and C. Schmid. Action recognition with improved trajectories. In *International Conference on Computer Vision (ICCV)*, 2013.
39. H. Wang, M. M. Ullah, A. Klaser, I. Laptev, and C. Schmid. Evaluation of local spatio-temporal features for action recognition. In *British Machine Vision Conference (BMVC)*, pages 124.1–124.11, 2009.
40. L. Wang, Y. Qiao, and X. Tang. Mining motion atoms and phrases for complex action recognition. In *International Conference on Computer Vision (ICCV)*, pages 2680–2687, 2013.
41. L. Wang, Y. Qiao, and X. Tang. Latent hierarchical model of temporal structure for complex activity classification. *Image Processing*, 23(2):810–822, 2014.
42. E. Zare-Borzeshi, O. Perez-Concha, and M. Piccardi. Human action recognition in video by fusion of structural and spatio-temporal features. In *Structural, Syntactic, and Statistical Pattern Recognition*, volume 7626 of *Lecture Notes in Computer Science*, pages 474–482. 2012.