Abstract. The ability to recognize human actions using a single viewpoint is affected by phenomena such as self-occlusions or occlusions by other objects. Incorporating multiple cameras can help overcome these issues. However, the question remains how to efficiently use information from all viewpoints to increase performance. Researchers have reconstructed a 3D model from multiple views to reduce dependency on viewpoint, but this 3D approach is often computationally expensive. Moreover, the quality of each view influences the overall model and the reconstruction is limited to volumes where the views overlap. In this paper, we propose a novel method to efficiently combine 2D data from different viewpoints. Spatio-temporal features are extracted from each viewpoint and then used in a bag-of-words framework to form histograms. Two different sizes of codebook are exploited. The similarity between the obtained histograms is represented via the Histogram Intersection kernel as well as the RBF kernel with $\chi^2$ distance. Lastly, we combine all the basic kernels generated by selection of different viewpoints, feature types, codebook sizes and kernel types. The final kernel is a linear combination of basic kernels that are properly weighted based on an optimization process. For higher accuracy, the sets of kernel weights are computed separately for each binary SVM classifier. Our method not only combines the information from multiple viewpoints efficiently, but also improves the performance by mapping features into various kernel spaces. The efficiency of the proposed method is demonstrated by testing on two commonly used multi-view human action datasets. Moreover, several experiments indicate the efficacy of each part of the method on the overall performance.

Keywords. histogram · combination · multi-view · action recognition
1 Introduction

Recognizing human actions from video is an important problem in computer vision and has a wide variety of applications, including video content analysis, video surveillance and human-computer interaction. Actions occur in three-dimensional world space. Projecting a truly three-dimensional phenomenon onto two-dimensional space as observed by a camera results in the loss of important information. One obvious situation when action recognition from a single camera fails is when the subject is occluded by an object. In addition, unavoidable self-occlusions may hide some parts of the action. This issue has led to the use of more than one camera to get a complete picture of action and motivated the research area of multi-view human action recognition. The literature on multi-view human action recognition can be divided into two categories: 3D approach and 2D approach [17]. In the former approach, multiple views are used to reconstruct a 3D model of the human body from which motion representations can be formed for action recognition. The 3D model of the human body can be based on surface mesh, ellipsoids, cylinders or visual hulls generated from silhouettes. Examples of motion representation include motion history volume [49], 3D optical flow [16] and shape histogram [18]. Model-based human pose tracking also aids action recognition, although pose tracking itself is a challenging problem. Such methods could exploit the kinematic constraints of the human body [9] or use specific regions of the human body model, like the upper body [36]. In contrast, the 2D approach uses shape and silhouette features from individual views (e.g. [15]) or motion features, such as 2D optical flow [28], motion history image [45] and local spatio-temporal interest points [48]. These features are then combined using early or late fusion; the former implies fusing the features and classifying them, while the latter implies classifying each feature independently and combining the results of each classifier.

The 3D approach often results in better accuracy than the 2D approach, albeit at higher computational costs [17]. The method by Turaga et al. [41], being the state of the art in 3D methods, has a recognition accuracy of 98.8% on the multiview IXMAS action dataset [49]. The state of the art among 2D approaches is the method by Vitaladevuni et al. [45] with 87.0% accuracy on IXMAS. Comparing the computational costs at a rounded accuracy of 81% on IXMAS, the 3D method by Weinland et al. [47] has an average testing run time of 2.5 fps, while Lv and Nevatias [26] 2D approach runs at 5.1 fps, which is almost double the speed. Thus, with respect to the application at hand and real-time requirements, the 2D approach may be more desirable. Moreover, the quality of segmentation or extraction of features at each viewpoint influences the quality of the overall 3D model. Furthermore, reconstruction of the model is limited to volumes where the views overlap. To build a reliable 3D model, a sufficient number of views should be available. In this paper, we describe a 2D approach for human action recognition using information from multiple cameras. The video is segmented into multiple small sequences and a decision can be made about each segment while the video is running so as to provide
continuous recognition; thus, this approach is more suitable for real-time applications. The proposed approach achieves state of the art performance among the methods that fall under the 2D approach.

Clearly, a 2D method implies that features are extracted from individual views and are then combined to arrive at a decision on action recognition. Feature combination has been studied extensively by researchers in computer vision as well as in machine learning [14,2,6]. It is now well-established that combining a set of diverse and complementary features results in better classification than using a single feature. Sun [39] discussed two methods for learning with multiple distinct feature sets. These methods are based on canonical correlation analysis (CCA) as an early fusion technique and co-training as a late fusion method. Both of these methods have been applied to dimensionality reduction, clustering and other types of learning, such as supervised, semi-supervised, active learning, ensemble learning and transfer learning. Originally proposed for two views, CCA finds linear transformations for each view such that the correlation between the mapped views is maximized. Co-training, on the other hand, trains classifiers for each view on a limited labeled dataset. These classifiers can confidently predict a pool of unlabeled data to enlarge the training labeled dataset. The process continues until a condition is met. The classifiers are used separately or jointly on a new testing sample. Cai et al. [7] proposed a method for feature combination based on a generative mixture model of probabilistic canonical correlation analyzers. Besides early and late fusion, a new fusion technique called mid-level fusion, which is essentially based on kernels, has recently emerged [13]. There are two main advantages of using kernels: i) they can model similarities between samples over different feature spaces and ii) a simple linear classifier is sufficient for classification [13]. Multiple kernel learning approaches have been shown to be one of the most effective methods of kernel-based feature combination for object/scene classification, e.g., [42,14,19]. Most of these methods use a large number of heterogeneous features, e.g., [14] uses shape, color, texture, HOG, HSV and SIFT features for flower classification. In this paper, we use simple local features that can be easily extracted without the prior need for segmentation, as required when working with silhouettes. Instead of extracting many different features, we focus on computing different models using different codebooks and functions from the same set of features and combining them efficiently.

Figure 1 shows examples of images from two different viewpoints. The two actions check watch and cross arms appear to be very similar in the side view, but different in the top view. Our method helps to learn the discriminative views by differentiating between each pair of actions. CCA maps different views to the same space for use with the same classifier. Co-training trains different classifiers suitable for each viewpoint. Unlike CCA and co-training, our method using multiple kernel learning is able to efficiently weight the transformed features of each view based on their importance for classifying actions. By generating various histograms for representing the action, comparing them in different spaces using different distances and then efficiently combining them
Fig. 1 Example images from actions *cross arms* and *check watch* from side view as well as top view.

Fig. 2 Flowchart of the proposed method.

together, our method achieves the best accuracy among 2D approaches on the IXMAS dataset and the highest accuracy on the WVU dataset [34].

1.1 Overview of the Proposed Method

Figure 2 shows the flowchart of the proposed method. The spatio-temporal features are extracted from the training videos captured from different viewpoints. These features are clustered into a fixed-size visual vocabulary using k-means. The extracted features from the training clips are quantized to form the histograms. Basic kernels are computed from these histograms considering the proper distances. Next, all the basic kernels are linearly combined using a non-sparse multiple kernel learning algorithm that gives the optimum kernel weights. The final classifier uses the linear combination of the kernels as the input for SVM. Similar to the training videos, the features are extracted from all the viewpoints of the query action and quantized using the learned
Fig. 3 The method for generating basic kernels $K_{c,f,v,k}$ from all the possible combinations of constitutive factors, namely, camera viewpoint ($c \in \{\text{View 1, ..., View C}\}$), feature type ($f \in \{\text{Separable Linear Filters, Space-time Corner Detector}\}$), codebook size ($v \in \{V, 2V\}$) and kernel type ($k \in \{\text{HIK distance in a linear function, Chi-Square distance in an exponential function}\}$). The basic kernel $K_{c,f,v,k}$ is weighted with $w_{c,f,v,k}$. $K$, the final kernel used, is the linear combination of basic kernels.

visual codebook to form histograms. Basic kernels are computed in the same method as for the training step. These kernels are then linearly combined with the learned optimum weights to make the final kernel used by the classifier. Figure 3 shows the pseudo code for generating all the basic kernels from the possible combinations of constitutive factors. For each view, two different kinds of local features are extracted. One of the features is obtained by applying separable linear filters to the space and time domains in order to ob-
taining keypoints throughout the video. The cuboids containing the keypoints are described by applying brightness gradients followed by PCA [11]. The other local feature is extracted by applying a space-time corner detector at multiple spatial and temporal scales and is described by an HOG feature concatenated by HOF (Histogram of Optical Flow) [23]. The raw features are clustered by k-means into $V$ and $2V$ number of clusters to generate two codebooks per feature. We use the codebooks to quantize the features extracted from each video in order to represent them as histograms. Two distance measures between the histograms - the histogram intersection kernel (HIK) distance and the $\chi^2$ distance - are computed and inserted into the linear and exponential functions, respectively, to form the fundamental kernels. These kernels are linearly combined with the proper weights to form the final kernel used with the SVM binary classifier. The optimum weights for the fundamental kernels are obtained by a 2-norm non-sparse multiple kernel learning algorithm [21].

1.2 Organization

This paper is organized as follows. Section 2 reviews related works on multiview human action recognition. The representation of video with histograms of words is described in Section 3. The method for combining the histograms is detailed in Section 4. Section 5 presents and discusses the experimental results. Finally, we conclude the paper in Section 6.

2 Related works

In this section, we discuss methods that combine information from different views for action or gesture recognition. These methods generally fall into two categories: the 3D approach, which combines information from multiple views into a 3D model to remove the view-dependent information and subsequently uses the model to compare different actions [49,41,43,32,30,16,9,36,50] and the 2D approach, which extracts features from each 2D viewpoint and subsequently combines these features using an early or late fusion model [38,35,46,48,25,47,26,28,15,6,12,24,20].

2.1 3D approach

Among the methods using the 3D model, Weinland et al. [49] introduced Motion History Volumes based on the idea of Motion History Images [5]. They used extracted silhouettes from different views to generate a 3D visual hull using the carving method. In this method, the initial 3D volume is discretized into voxel points. Using the calibration matrices, each voxel point is projected onto the image planes of the cameras; only the voxel points projected onto the foreground of all cameras are considered part of the visual hull. The integration of all visual hulls over time in a sequence is represented in the Motion
History Volume. Subsequently, Fourier transforms in cylindrical coordinates are extracted from the volumes as the motion descriptors. The descriptors are projected by PCA and classified based on the Mahalanobis distance from each class mean. Turaga et al. [41] exploited the same motion descriptors, but the actions were modeled and learned using a sophisticated statistical model based on Stiefel and Grassmann manifolds. Moreover, Veeraraghavan et al. [43] used a time series of circular FFT features from [49] in a rate invariant model that is robust to changes in execution rate. Weinland et al. [47] learned 3D exemplars projected along different viewpoints and compared them with 2D observations. Recognition is performed using Hidden Markov Models.

Visual hulls are also used in the work of Peng and Qian [32]. Voxel data are projected onto a low dimensional pose coefficient space using multilinear analysis. Eventually, the sequences of pose descriptors are represented and classified by Hidden Markov Models. Pehlivian and Duygulu [30] divided visual hulls into horizontal layers where each layer is coded by the enclosing circle. The properties of enclosing circles are used to describe each pose. Consequently, the pose descriptors of all the frames in the sequence are combined into a motion descriptor that is used in a nearest neighbor framework.

3D optical flow was also used in a 3D model by Holte et al. [16]. They used motion information by embedding Histograms of 3D Optical Flow in a spherical histogram. Subsequently, 3D Motion Context (3D-MC) can be extracted, which is a 3D motion-based descriptor based on the idea of shape context [4]. Another 3D descriptor proposed by [16] is Harmonic Motion Context (HMC), which extends the 3D-MC descriptor by decomposing it into a set of spherical harmonic basis functions.

A probabilistic method has been proposed by Cheng and Trivedi [9] in which the spatial distribution of voxels is modeled by Gaussian Mixture Models. Each body segment is described by a single Gaussian component. All of the components are kinematically constrained based on a skeleton model. A Bayesian inference framework with a particle filter was exploited by Song et al. [36], who used depth information from two cameras to track upper body poses.

Yan et al. [50] proposed a 4D action feature model in which spatio-temporal features are extracted from each view and mapped to a sequence of reconstructed visual hulls. Their method can be used for recognizing actions from arbitrary viewpoints. Vemulapalli et al. [44] represented 3D skeletons as points in a lie group using rotations and translations in 3D space.

2.2 2D approach

Although 3D methods usually result in better accuracy, they are computationally expensive. In addition, the quality of features or silhouettes extracted from each view affects the quality of the overall reconstructed model. This makes the model vulnerable to the deficiencies in each viewpoint. Generally, reconstruction of a 3D model is confined by the volumes where the views over-
lap. Thus, a reliable 3D model cannot be obtained unless sufficient viewpoints are available. These drawbacks have motivated researchers to employ the 2D approach based on their needs.

Methods that fall under the 2D approach extract features from 2D images associated with different viewpoints. Based on availability, some methods use all of the viewpoints to classify a query action [28, 15], while others use a single viewpoint [38, 35, 46, 48, 25, 47, 26]. In both methods, training is done using all available viewpoints. Therefore, the query viewpoint is among the trained viewpoints. If the query viewpoint is different from the viewpoints learned, the method performs cross-view recognition, which is a quite challenging task [12, 24, 20].

Matikainen et al. [28] used motion information to detect pointing gestures and estimate the direction using two camera views. The circular shift invariance property of the discrete Fourier transform (DFT) magnitudes was used by Gkalelis et al. [15] to solve the view correspondence between training and test sets. Actions can be described and classified using Fuzzy Vector Quantization (FVQ) and Linear Discriminant Analysis (LDA). Pehlivan and Forsyth [31] fused the action labels over frames and cameras to get the sequence label. Wu et al. [6] presented three fusion methods for multi-view action recognition: (a) Best-view fusion: essentially a late fusion method in which a classifier is trained for each camera independently. For each episode of the clip, the fusion process chooses the best view camera. Different criteria can be used to determine the best-view camera. The authors used the number of detected spatio-temporal features as their criterion. (b) Combined-view fusion: an early fusion technique in which all the features from different cameras are simply concatenated into a single feature to be classified. (c) Mixed-view fusion: neither early nor late fusion, this method uses the same classifier regardless of viewpoint. Best-view fusion has the highest accuracy while mixed-view fusion has the lowest. All of these methods -except mixed-view fusion- use every available viewpoint to classify an action query.

In some applications we may train a recognition system using all the available viewpoints, but classification is done using only one of the learned views. In work by Souvenir et al. [38] R-transforms of silhouettes were computed, which led to R-transformed surfaces for each sequence. Manifold learning is used to model actions relative to viewpoint. Reddy et al. [35] proposed using feature-trees to index motion features regardless of viewpoint. A voting scheme is used to recognize the actions. A Bayesian framework was introduced by Vitaladevuni et al. [45] to classify actions based on Ballistic Dynamics. Weinland et al. [48] proposed a method that is robust to viewpoint change and occlusion by local partitioning and hierarchical classification of a 3D Histogram of Oriented Gradients (HOG). The results from local classifiers are then fused to get the final decision. Shape context was used by Lv et al. [26] to represent key postures and classify actions using the Viterbi Path Searching algorithm. In this method, learning is computationally expensive since they used synthetic data rendered from a wide range of views for training. The bag-of-video-words framework was used by Liu et al. [25] in which video-words are further clus-
tered using maximization of mutual information. This method does not involve any smart fusion of viewpoints. A single classifier is learned regardless of viewpoint. Since features from different viewpoints vary in appearance, this method may not be very efficient for classifying different views. Ashraf et al. [1] decomposed a posture into a set of projective depths that are invariant to camera internal parameters and orientations.

Cross-view action recognition techniques can classify views not previously learned. The correlation of actions from different views is addressed in the work of Farhadi and Tabrizi [12]. A split-based representation was used to describe the cluster of video-words in each viewpoint. The transfer of these splits between viewpoints was learned from sample sequences. Liu et al. [24] proposed fusing two types of features to handle image variation: a vocabulary of local spatio-temporal features and a vocabulary of spin-images. Lastly, Juneju et al. [20] proposed a descriptor based on observing the self-similarities of video sequences over time, which shows stability under view changes.

Our proposed method fully exploits the available sets of viewpoints by using all of them for learning as well as recognition. Moreover, by incorporating an efficient kernel fusion technique between each pair of classes, the discriminative features extracted from each view are boosted, giving rise to our models state of the art performance among 2D approaches.

3 Histogram Representation of Video

The representation of videos as histograms enables us to model videos with different criteria and different details, which helps us analyze them in greater depth. The different histograms are generated based on two local features - Separable Linear Filters [11] and Space-Time Corner Detector [23]- as well as two sizes of codebooks. The codebooks are generated by clustering the training pool of features by k-means using Euclidean distance as a metric.

3.1 Separable Linear Filters

Assuming a stationary camera, separable linear filters apply a Gaussian filter to the spatial domain and a quadratic pair of Gabor filters to the temporal dimension to obtain a response function as follows [11]:

$$ R = (I * g * h_{ev})^2 + (I * g * h_{od})^2, $$

where \( g(x, y; \sigma) \) is the 2D Gaussian smoothing filter, and \( h_{ev} \) and \( h_{od} \) are the quadratic Gabor filters defined as \( h_{ev}(t; \tau, \omega) = -\cos(2\pi t \omega) \times e^{-t^2/\tau^2} \) and \( h_{ev}(t; \tau, \omega) = -\sin(2\pi t \omega) \times e^{-t^2/\tau^2} \). The parameters \( \sigma \) and \( \tau \) represent the spatial and temporal scale of the detector, respectively. \( \omega \) is always chosen as \( 4/\tau \). In order to handle multiple scales, the detector can be run at multiple spatial and temporal scales. For computational reasons, we use only one scale and rely on the vocabulary to obtain robustness with respect to scale changes.
In all of the experiments we use $\sigma = 2, \tau = 3$. The space-time interest points are detected at the local maxima of the response function. A cuboid is extracted around each keypoint with a size roughly six times the scale parameter along that dimension.

The extracted cuboids are described using their brightness gradients along the directions $x, y$ and $t$. Before computing the gradients, the cuboids are smoothed at different scales. The computed gradients at different directions and different smoothing scales are concatenated to form a vector. This vector is projected onto a lower dimensional space by Principal Component Analysis (PCA) to obtain the final descriptor. The size of the final descriptor in our experiments is 100.

3.2 Space-Time Corner Detector

The Space-Time Corner Detector is an extension of the Harris corner detector that was proposed by Laptev and Lindeberg [22]. A spatio-temporal second-moment matrix is computed by

$$
\mu(0; \sigma', \tau') = g(0; s\sigma', s\tau') * (\nabla L(0; \sigma', \tau')(\nabla L(0; \sigma', \tau'))^T),
$$

where $g$ is the Gaussian smoothing kernel with independent spatial and temporal scale values $\sigma', \tau'$, $\nabla L$ is the space-time gradient and $\ast$ denotes convolution. The final location of keypoints are determined by local maxima of the function $Z = \text{det}(\mu) - \rho \text{trace}^3(\mu), Z > 0$. Instead of spatio-temporal scale selection, as proposed by [22], we follow [23] and use multiple levels of spatio-temporal scales. This has been shown to be effective for action recognition [23]. We choose spatio-temporal scales as $\sigma'^2 = 4, 8, 16, 32, 64, 128, \tau'^2 = 2, 4$. We also use $\rho = 5 \times 10^{-7}$.

Histogarms of spatial gradients and optical flow are computed for each cuboid around the keypoints in order to capture the information regarding appearance and motion [23]. The size of the cuboids in the neighborhood of keypoints is determined by $\Delta_x, \Delta_y = 2qs', \Delta_z = 2qr'$. Each cuboid is subdivided into a $n_x \times n_y \times n_t$ grid of cells. for each cell, we compute 4-bin HOG and 5-bin HOF. The computed histograms are normalized and concatenated to form the final descriptor, which is similar to the famous SIFT descriptor. Parameter values of $q = 5, n_x, n_y = 3$ and $n_t = 2$ are used in the experiments, which result in a descriptor with the dimensions of 162.

3.3 Codebook Size

After extracting the features, we use them to obtain a codebook for each view. We apply k-means clustering to the training features for each view point using the Euclidean distance. We use two codebooks, one with a size of $V$ and the other with a size of $2V$. According to Gehler and Nowozin [14], adding any feature (kernel) - even an uninformative or non-discriminative one - to the kernel
weight optimization methods will not reduce the classification performance. In particular, when the added feature (kernel) is informative or discriminative, the classification performance will increase. Using two different sizes of vocabulary will enable us to model the actions with two different scales of detail. Here, we empirically chose V to be 75 for all features. The features are subsequently quantized based on the vocabularies to form different histograms.

4 Efficient Combination of View-dependent Histograms

After computing histograms for each view and for each feature and codebook size, we are left with a number of different histograms representing an action video. In order to efficiently capture the similarities between different histograms, we must use the right measures of distance. Then the distances can be integrated into the appropriate functions to form similarities in a kernel matrix. To model actions in different spaces, in this paper we use the HIK distance within a linear function as well as the $\chi^2$ distance within an exponential function.

4.1 Computing Histogram Intersection Kernel

The Histogram Intersection Kernel (HIK), which was first introduced by Swain and Ballard [40], is a measure of similarity between histograms. HIK between histograms $h$ and $h'$ with $d$ number of bins is defined as:

$$k_{HIK}(h_a, h_b) = \sum_{i=1}^{d} \min(h_a(i), h_b(i)).$$

(3)

Barla et al. [3] showed that HIK is positive definite and, hence, can be used as a kernel with SVM. Moreover, they verified that image classification using SVM with HIK is effective. On the other hand, Maji et al. [27] proposed a method for accelerating the computation of the kernel that greatly reduces the computational cost. These facts motivated us to use HIK for our histograms.

4.2 Computing Radial Basis Function Kernel with $\chi^2$ Distance

Radial Basis Function (RBF) kernel maps the samples nonlinearly onto a higher dimensional space. It is defined as [8]:

$$k_{RBF}(h_a, h_b) = \exp(-\gamma D(h_a, h_b)), \gamma > 0,$$

(4)

where $\gamma$ is the bandwidth parameter and $D$ is an arbitrary distance metric. In this work, we use $\chi^2$ as the distance metric, defined as:

$$D_{\chi^2}(h_a, h_b) = \frac{1}{2} \sum_{i=1}^{d} \frac{(h_a(i) - h_b(i))^2}{h_a(i) + h_b(i)}.$$

(5)

$\chi^2$ distance is commonly used to compare histograms.
4.3 Learning an Efficient Combination of Kernels

The HIK and RBF kernels from different histograms must be combined in an efficient way to obtain an optimized final kernel. The final kernel is used with SVM to classify the actions. Thus, the binary SVM classifier will be in the form of [8]:

\[ F(x) = \text{sign}(\sum_{i=1}^{N} \alpha_i y_i K(x, x_i) + \beta_0), \quad (6) \]

where \( \{x_i\}_{i=1}^{N} \) are the training sample videos and \( \{y_i\}_{i=1}^{N} \in \{-1, +1\} \) are the associated binary labels. \( \{\alpha_i\}_{i=1}^{N} \) and \( \beta_0 \) are the parameters of SVM learned from the examples \(^1\). \( K \) is a linear combination of basis kernels \( K_m \) [21]:

\[ K(x, x') = \sum_{m=1}^{M} w_m K_m(x, x'), 0 \leq w_m. \quad (7) \]

\( \{K_m\}_{m=1}^{M} \) are computed from different histograms using either the HIK or RBF kernel. \( w_m \) is the kernel weight that changes (scales) the influence of the kernel space associated with \( K_m \) and, subsequently the corresponding histogram space. The condition \( 0 \leq w_m \) ensures that the obtained matrices are positive-semidefinite. We use a 1-vs-1 classification scheme and choose different weights for each binary classifier. Choosing the optimum weights is important for the performance of the system, since some histograms (feature spaces) may be discriminative in differentiating between a pair of classes but non-informative for another pair. For instance, two actions may seem different from one viewpoint, but similar from another. Also, a particular local feature may be informative for distinguishing some pairs of actions but not others. Moreover, some actions are more efficiently represented using HIK kernels, while others are more easily distinguished using RBF kernels. Due to these uncertainties, we use a learning method that estimates the weights from the data in an optimum way.

For every combination of local feature and codebook size, we incorporate only one instance of the HIK kernel and four instances of the RBF kernel. This is because HIK kernels are mixed linearly \((w_1^{HIK} K_h + w_2^{HIK} K_h)\), and any linear combination can be replaced with one kernel with the appropriate weight. On the other hand, RBF kernels are combined with different bandwidths \((w_1^{RBF} \exp(-\gamma_1 D_{\chi^2}) + w_2^{RBF} \exp(-\gamma_2 D_{\chi^2}))\) and, hence, they form different kernels. We use four different RBF kernels of bandwidth: \( \gamma = 0.005, 0.05, 0.5, \) and \( 1. \)

In this paper, we use a non-sparse multiple kernel learning algorithm [21]. The common sparse kernel combinations - such as Semi-Infinite Linear Program (SILP) [37] or SimpleMKL [33] - are interpretable by extracting the relevant pieces of information. In addition, they reduce complexity by removing unrelated features. However, trivial baseline methods using average

\(^1\) note that \( \alpha_i \neq 0 \) only for support vectors
weights usually outperform sparse mixtures [10]. The reason for this is that, in real-world applications, the kernels are highly sophisticated and the whole set participates in capturing the relevant information. As a result, removing some kernels in sparse models may lead to poor performance. Thus, we use a general $l_p$-norm multiple kernel learning model where no feature is removed; rather, all features participate and with different contributions. We empirically select the $l_2$-norm. Newton descent is used for optimization due to its fast performance compared to cutting planes [21].

5 Experimental Results

In order to show the efficiency of the proposed method, we tested it on two multi-view datasets: the multi-view IXMAS dataset and the WVU multi-view dataset [34]. Moreover, experiments evaluating the influence of each component were carried out on the challenging IXMAS dataset. Experiments were designed in such a way as to analyze each component of the algorithm, viz., the number of views, the effect of features, the number of codebooks and the distance measures between histograms. Lastly, we compared our method with other fusion techniques as well as recent results on the IXMAS dataset. In all of the experiments, we fixed the SVM cost parameter ($C$) to 1. We used the leave-one-person-out cross validation scheme. Thus, each time we tested on the sequences of one person and trained on the sequences of the remaining persons. We iterated on all persons. Accuracy was computed as the average recognition rate over all repetitions. The multiple kernel learning algorithm gives the optimum SVM parameters in equation (6) as well as the optimum kernel weights in equation (7).

INRIA IXMAS dataset [49] is a challenging multi-view dataset for action recognition that is publicly available. It comprises 14 daily life actions, each performed three times by 12 actors. In order to test view-invariance, the actors freely change orientations in each performance without any information provided other than the labels. To be comparable, in our experiments we used the same 11 actions and 10 subjects used in [49] and [32]. Example images of 11 actions are shown in figure 4. The motions in the IXMAS dataset were captured using five fire-wire cameras. Figure 5 shows example views from the five cameras for the kick action. Similar to [49] and [32], we used leave-one-person-out cross validation: in each cycle, we trained with the data of nine persons and tested with the data of the remaining person. This procedure was repeated for all 10 persons. The accuracy reported is the average of all 10 runs.

5.1 Results and Analysis

In this section we aim for the best performance using all possible kernels. Thus, for the IXMAS dataset, we consider every combination of constitutive factors,
Fig. 4 Example images of the actions in IXMAS dataset.
Efficient 2D Viewpoint Combination for Human Action Recognition

Fig. 5 Example views from five cameras in IXMAS dataset for kick action.

Fig. 6 Confusion matrix for the best result achieved on IXMAS.

namely five camera viewpoints ($C_1, C_2, C_3, C_4, C_5$), two feature types (Separable Linear Filters and Space-time corner detectors), two codebook sizes ($V$ and $2V$) and two kernel types (HIK and RBF). In addition, we apply four RBF kernels instead of one. This gives a total number of 100 basic kernels to be linearly combined, according to equation (7). The kernel weights are computed for each binary SVM separating a pair of classes. Figure 6 shows the confusion matrix for the IXMAS dataset. As shown, most of the confusion occurs between check watch and cross arms or wave, which have somewhat similar hand movements. Most other actions are perfectly discriminated. The overall recognition accuracy is 95.8%, which is the state of the art among 2D methods for the IXMAS dataset. The best previously reported accuracies are 87.0% for a 2D model [46] and 98.8% accuracy for a 3D model [41]. In addition, we tested our method on the WVU multi-view action recognition dataset [34]. This dataset comprises 10 actions recorded using eight cameras within the same angular distance of each other covering 360 degrees. Sample images of the actions for this dataset are shown in figure 7. For each viewpoint, there are 47 clips for training and 18 clips for testing. Different people appear in the videos. In the training samples, the subject is always at the center of the room; in the testing videos, the subject can be in any location. The dataset

3 The actions are standing still, clapping, waving one arm, waving two arms, punching, jogging, jumping jack, kicking, bending and bowling.
is provided as a set of images, which are recorded at a rate of 20 fps with a resolution of 640x480. The authors in [34] used bounding boxes of humans and extracted silhouettes from the subjects. They achieved an accuracy of almost 98% using all the cameras. We reduced the resolution to 160x120 and used gray levels. Also, we regenerate the original clips using the same frame rate. Unlike [34], which used bounding boxes, we used the entire image including the background. Dollar features [11] were extracted from the clips. We used a codebook size of 300 to quantize the visual words. Using an HIK kernel, our framework achieved recognition accuracy of 100%. In this dataset, the angle between the subject and camera varied between the training and testing samples. Since our method assumes a fixed subject-camera angle, we rearranged the viewpoint labels in the test samples so they matched the angle for the training samples.
5.2 Viewpoint Analysis

In order to compare different camera viewpoints, we experimented with all the kernels extracted from each individual view of IXMAS. The result is shown in figure 8. As expected, the best accuracy was obtained with the front (camera 2) and side (camera 4) views. The top view (camera 5) had the worst overall recognition rate, as most of the movements are partially covered when observed from the top.

We also analyzed the effect of combining different views. Specifically, we considered the recognition accuracy when the number of views varied from 1 to 5. Since there are a number of ways by which the views can be combined, we show only those results that give the best accuracy in figure 9. As expected, more views result in higher recognition accuracy. The specific viewpoints mentioned in parenthesis are the combinations which give the best accuracy among all the possible combinations. For instance, for 3 views there are 10 possible combinations. The combination C1-C2-C4 gave the best result in this case. According to figure 9, the combinations composed of the best viewpoints in figure 8 gave the best performance.

5.3 Effect of Feature Type

Here, we studied the performance of each local feature type individually. We combined all the basic kernels obtained with the Separable Linear Filter and compared them to the combination of kernels using the Space-time Corner Detector. The results for each feature-type are illustrated separately and together in figure 10. As shown Separable Linear Filter significantly outperforms
Fig. 9 Best accuracy for combination of viewpoints in IXMAS. The camera viewpoints which have resulted in the best accuracy are: 1 viewpoint (C2), 2 viewpoints (C2-C4), 3 viewpoints (C1-C2-C4), 4 viewpoints (C1-C2-C3-C4).

Fig. 10 Performance of each feature type.

Space-time Corner Detector. This is likely due to the much denser keypoints generated from the separable filter method, which was also mentioned by Dollar et al. [11]. Moreover, the Gabor filter applied in the time domain helps capture periodic motions more efficiently. The accuracy is more than 85% when only Dollars feature is used.

In order to compare our method with mixed-view fusion [25,6], as explained in Section 2, we explored using the histograms obtained from Dollars feature
trained regardless of viewpoint using a single SVM classifier (with only the HIK kernel). This resulted in 58.3% accuracy, which is around 30% lower than using our proposed combination method with only HIK kernels. The accuracy reported in [25] is 82.8%, which was obtained using all views without the top view (four views) and based on a voting method. Also, they used 13 actions and 12 actors from IXMAS. The low classification rate of 58.3% was due to classifying different views with different appearances using the same model.

5.4 Effect of using more Codebooks

We formed the histograms based on two codebooks. These codebooks were generated from the same set, but had different sizes (V and 2V). Here, we studied how performance was affected by adding one more codebook. Figure 11 shows the performance of one codebook versus two codebooks using all the basic kernels. As shown, simply increasing the number of codebooks to two with different sizes improved the performance. This is much easier and more computationally efficient than adding another feature. Having two different sizes of codebook will enable us to model action in different scales of details. Adding more codebooks (more than two) may result in better accuracy, but comes with a higher computational cost.

5.5 Effect of Kernel Type

Here we explored the impact of choosing different kernel types (HIK kernel and RBF kernel with $\chi^2$ distance) and combinations on performance. We experimented using only HIK kernels, only RBF kernels and a combination of...
both. Note that we used four RBF kernels. In Section 4.3, we explained that we can use more than one RBF kernel with different bandwidths. The results are illustrated in Figure 12. Given a sufficient amount of training data, one RBF has almost the same performance as HIK; however, as the size of the training set decreases, HIK outperforms RBF. Although using more RBF kernels improves the results, HIK still outperforms four RBF in a small training set. Combining all the kernels does result in higher accuracy.

Note that by using different types of kernels and combining them, we increased performance without going to the trouble of extracting more information from the video. In fact, we represented the same data in different spaces with different metrics (histogram intersection and $\chi^2$). Each of these models has its advantages and disadvantages and none was absolutely better than another. Thus, we combined them with adaptive weights that depend on the pairs of classes to optimize the performance.

5.6 Comparison of Different Fusion Methods

Our proposed method weights each basic kernel differently. In order to show that our weighting is efficient, we compared our results with equal weighting of basic kernels. Thus, to combine the basic kernels, we simply averaged them to form the final kernel and then classified it with SVM. Since HIK and RBF kernels are from different spaces with different scales, we cannot simply add them together. Thus, we studied them separately. The results are shown in figure 13 According to the figure, our combination method outperforms the simple averaging of kernels. In the case of the RBF kernel, the difference is larger.
Fig. 13 Comparison of different fusion methods.

Table 1 Comparison of Recognition Accuracy on IXMAS dataset.

<table>
<thead>
<tr>
<th>Method</th>
<th>Dim</th>
<th>Accuracy(%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Turaga et al. [41]</td>
<td>3D</td>
<td>98.8</td>
</tr>
<tr>
<td>Veeraraghavan et al. [43]</td>
<td>3D</td>
<td>98.2</td>
</tr>
<tr>
<td>Ours</td>
<td>2D</td>
<td>95.8</td>
</tr>
<tr>
<td>Pehlivan and Forsyth [31]</td>
<td>2D</td>
<td>95.1</td>
</tr>
<tr>
<td>Peng and Qian [32]</td>
<td>3D</td>
<td>94.6</td>
</tr>
<tr>
<td>Weinland et al. [49]</td>
<td>3D</td>
<td>93.3</td>
</tr>
<tr>
<td>Pehlivan and Duygulu [30]</td>
<td>3D</td>
<td>90.9</td>
</tr>
<tr>
<td>Ashraf et al. [1]</td>
<td>2D</td>
<td>90.5</td>
</tr>
<tr>
<td>Vitaladevuni et al. [45]</td>
<td>2D</td>
<td>87.0</td>
</tr>
<tr>
<td>Weinland et al. [48]</td>
<td>2D</td>
<td>83.4</td>
</tr>
<tr>
<td>Liu and Shah [25]</td>
<td>2D</td>
<td>82.8</td>
</tr>
<tr>
<td>Weinland et al. [47]</td>
<td>3D</td>
<td>81.3</td>
</tr>
<tr>
<td>Lv and Nevatia [26]</td>
<td>2D</td>
<td>80.6</td>
</tr>
<tr>
<td>Yan et al. [50]</td>
<td>3D</td>
<td>78.0</td>
</tr>
<tr>
<td>Reddy et al. [35]</td>
<td>2D</td>
<td>72.6</td>
</tr>
</tbody>
</table>

5.7 Comparison with Other Methods on IXMAS Dataset

To better study the efficiency of the proposed method, we compared our result with the recent results of action recognition on the IXMAS dataset in table 1. As shown, the 3D methods generally perform better than 2D methods. This is because the 3D methods exploit all the views to evaluate a query action, unlike the reported 2D methods which are based on single-view testing. However, 3D methods are usually computationally expensive and not suitable for real-time applications. Moreover, the quality of the reconstructed 3D model usually depends on the number of viewpoints used. For methods that use silhouettes or tracking, the quality of the segmentation or tracking algorithm influences...
Table 2 Comparison of the average testing run time on IXMAS dataset.

<table>
<thead>
<tr>
<th>Method</th>
<th>Dim</th>
<th>Average fps</th>
<th>Accuracy(%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Naiel et al. [29]</td>
<td>2D</td>
<td>702.7</td>
<td>84.6</td>
</tr>
<tr>
<td>Ours</td>
<td>2D</td>
<td>9.5</td>
<td>95.8</td>
</tr>
<tr>
<td>Lv and Nevatia [26]</td>
<td>2D</td>
<td>5.1</td>
<td>80.6</td>
</tr>
<tr>
<td>Weinland et al. [47]</td>
<td>3D</td>
<td>2.5</td>
<td>81.3</td>
</tr>
</tbody>
</table>

the 3D model. Self-occlusion is an important problem when generating a 3D model from multiple views and leads to merging different body segments. Occlusion by other objects is similar to self-occlusion. Since the method uses multiple views to capture the motion, the parts being occluded in one view can usually be seen in other views. Since we did not construct a 3D model, the different viewpoints do not interfere with each other and the features are captured separately. Furthermore, reconstructing the 3D data is limited to spaces where the views overlap. As a 2D approach, our method can efficiently incorporate information from all views without the need for expensive 3D models. By using an efficient combination, we achieved results comparable to expensive 3D approaches with the state of the art results. The method by Turaga et al. [41] used a sophisticated statistical model based on Stiefel and Grassmann manifolds. The method by Veeraraghavan et al. [43] used a time series of 3D descriptors in a rate-invariant model. Our method is the state of the art among 2D approaches with results comparable to 3D approaches.

To show the computational efficiency, we compared our run time to several notable methods. Table 2 shows a comparison of the average testing run time based on fps on the IXMAS dataset using an Intel Core 2 CPU at 3 GHz. Our 2D method runs at 9.5 fps, while the method by Weinland et al. [47] - which is the fastest reported 3D algorithm [29] - runs at 2.5 fps. This shows that our method is computationally more efficient than 3D methods. The method by Naiel et al. [29] is the fastest method reported on the IXMAS dataset. Their method uses a parallel structure that applies 2DPCA to motion energy images in both spatial and transformed forms. The drawback of this method is the need for clear silhouettes, which are hard to extract in complex situations.

6 Conclusion

In this paper, we proposed a novel method for combining information from multiple viewpoints. Using the bag-of-words framework, different histograms were constructed from each viewpoint by changing the codebook size. The similarity between histograms is represented by the Histogram Intersection as well as the $\chi^2$ distance in an exponential function. All the generated basic kernels were combined linearly with the proper weights obtained by optimization. The final kernel was used with SVM. We performed extensive experiments to study the influence of each constitutive factor and verified the efficiency of our method. In our experiments, we showed that adding more number of views improved recognition of actions. We also verified that using features with denser
keypoints resulted in better performance. Using more number of kernels improved the accuracy. Our kernel combination method outperformed baseline averaging. Our method achieved state of the art results on the challenging multi-view IXMAS dataset compared to the other 2D methods reported to date. We believe that our proposed method offer significant advantages over 3D methods as they need sufficient number of views and the cameras require to be calibrated.

References


