This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2017.2691768, IEEE Transactions on Pattern Analysis and Machine Intelligence

Learning a Deep Model for Human Action Recognition from Novel Viewpoints

Hossein Rahmani, Ajmal Mian and Mubarak Shah

Abstract—Recognizing human actions from unknown and unseen (novel) views is a challenging problem. We propose a Robust Non-Linear Knowledge Transfer Model (R-NKTM) for human action recognition from novel views. The proposed R-NKTM is a deep fully-connected neural network that transfers knowledge of human actions from any unknown view to a shared high-level virtual view by finding a set of non-linear transformations that connects the views. The R-NKTM is learned from 2D projections of dense trajectories of synthetic 3D human models fitted to real motion capture data and generalizes to real videos of human actions. The strength of our technique is that we learn a single R-NKTM for all actions and all viewpoints for knowledge transfer of any real human action video without the need for re-training or fine-tuning the model. Thus, R-NKTM can efficiently scale to incorporate new action classes. R-NKTM is learned with dummy labels and does not require knowledge of the camera viewpoint at any stage. Experiments on three benchmark cross-view human action datasets show that our method outperforms existing state-of-the-art.

Index Terms—Cross-view, dense trajectories, view knowledge transfer.

1 Introduction

Video based human action recognition has many applications in human-computer interaction, surveillance, video indexing and retrieval. Actions or movements generate varying patterns of spatio-temporal appearances in videos that can be used as feature descriptors for action recognition. Based on this observation, several visual representations have been proposed for discriminative human action recognition such as space-time pattern templates [1], shape matching [2]–[4], spatio-temporal interest points [5]–[10], and motion trajectories based representation [11]–[14]. Especially, dense trajectory based methods [12]–[14] have shown impressive results for action recognition by tracking densely sampled points through optical flow fields. While these methods are effective for action recognition from a common viewpoint, their performance degrades significantly under viewpoint changes. This is because the same action appears different and results in different trajectories when observed from different viewpoints.

A practical system must recognize human actions from unknown and, more importantly, unseen viewpoints. One approach for recognizing actions across different viewpoints is to collect data from all possible views and train a separate classifier for each case. This approach does not scale well as it requires a large number of labelled samples for each view. To overcome this problem, some techniques infer 3D scene structure and use geometric transformations to achieve view invariance [3], [24]–[27]. These methods often require robust joint estimation which is still an open problem in real-world settings. Other methods focus on view-invariant spatio-temporal features [28]–[32]. However, the discriminative power of these methods is limited by their inherent structure of view-invariant features [33].

Knowledge transfer-based methods [17]–[23], [34] have

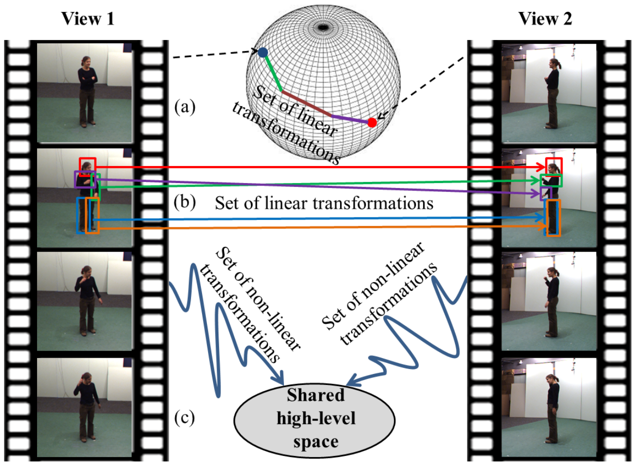
View 1
View 2
(a)
Set of linear transformations
(b)
Set of linear transformations
Set of non-linear transformations
Set of non-linear transformations
(c)
Shared high-level space

Fig. 1: Existing cross-view action recognition techniques [15]–[23] connect two different views with a set of linear transformations that are unable to capture the non-linear manifolds on which real actions lie. (a) Li and Zickler [23] construct cross-view action descriptors by applying a set of linear transformations on view-dependent descriptors. The transformations are obtained by uniformly sampling a few points along the path connecting source and target views. (b) Wang et al. [21] learn a separate linear transformation for each body part using samples from training views to interpolate unseen views. (c) Our proposed R-NKTM learns a shared high-level space among all possible views. The view-dependent action descriptors from both source and target views are independently transferred to the shared space using a sequence of non-linear transformations.

- *H. Rahmani and A. Mian are with the School of Computer Science and Software Engineering, The University of Western Australia, 35 Stirling Highway,Crawley, Western Australia, 6009. E-mail: hossein.rahmani@uwa.edu.au, ajmal.mian@uwa.edu.au*
- *M. Shah is with with the School of Electric Engineering and Computer Science, University of Central Florida, Orlando, FL 32816-2365 USA. E-mail: shah@eecs.ucf.edu*

0162-8828 (c) 2016 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2017.2691768, IEEE Transactions on Pattern Analysis and Machine Intelligence

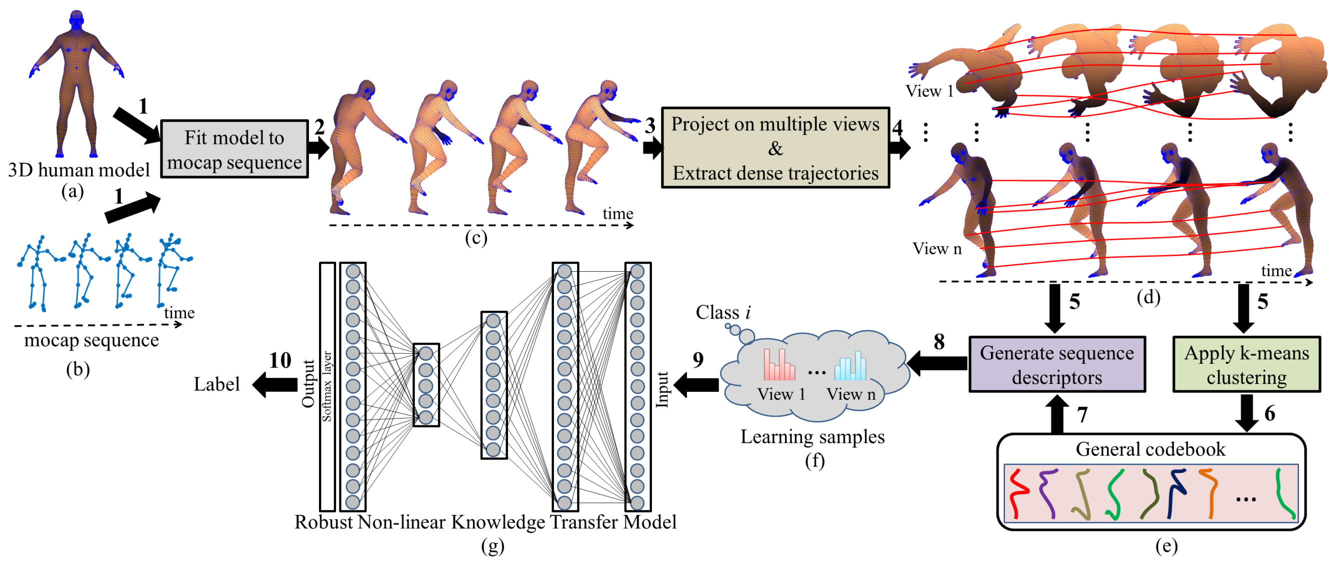

Fig. 2: Framework of the proposed R-NKTM learning algorithm. A realistic 3D human model (a) is fitted to a real mocap sequence (b) to generate 3D action video (c) which is projected to plains viewed from $n = 108$ angles. Projection from only two viewpoints are shown in (d). This results in n sequences of 2D pointclouds that are connected sequentially to construct synthetic trajectories (red curves in (d)) which are used to learn a general codebook (e). A bag-of-features approach is used to build the dense trajectory descriptors (f) from which a single R-NKTM (g) is learned. Note that instead of action labels, we use dummy labels where each 3D video gets a different label. The R-NKTM is learned once only and generalizes to real videos for cross-view feature extraction.

recently become popular for cross-view action recognition. These methods find a view independent latent space in which features extracted from different views are directly comparable. For instance, Li and Zickler [23] proposed to construct virtual views between action descriptors from source and target views. They assume that an action descriptor transforms continuously between two viewpoints and the virtual path connecting two views lies on a hypersphere (see Fig. 1-(a)). Thus, [23] computes virtual views as a sequence of linearly transformed descriptors obtained by making a finite number of stops along the virtual path. This method requires samples from both source and target views during training to construct virtual views.

To relax the above constraint on training data, Wang et al. [21] used a set of discrete views during training to interpolate arbitrary unseen views at test time. They learned a separate linear transformation between different views for each human body part using a linear SVM solver as shown in Fig. 1-(b), thereby limiting the scalability and increasing the complexity of their approach.

Existing view knowledge transfer approaches are unable to capture the non-linear manifolds where realistic action videos generally lie, especially when actions are captured from different views. This is because they only seek a set of linear transformations to construct virtual views between the descriptors of action videos captured from different viewpoints. Furthermore, such methods are either not applicable or perform poorly when recognition is performed on videos acquired from unknown and, more importantly, unseen viewpoints.

In this paper, we propose a different approach to view knowledge transfer that relaxes the assumptions on the virtual path and the requirements on the training data. We approach cross-view action recognition as a non-linear knowledge transfer learning problem where knowledge from multiple views is transferred to a shared compact high-level space. Our approach consists of three phases. Figure 2 shows an overview of the first phase where a Robust Non-linear Knowledge Transfer Model (R-NKTM) is learned. The proposed R-NKTM is a deep fully-connected network with weight decay and sparsity constraints, which learns to transfer action video descriptors captured from different viewpoints to a shared high-level representation. The strongest point of our technique is that we learn a *single* R-NKTM for mapping all action descriptors from all camera viewpoints to a shared compact space. Note that the labels used in Fig. 2 are dummy labels where every sequence is given a unique label that does not correspond to any specific action. For example, for a mocap dataset $S = \{S_1, S_2, \cdots, S_y\}$ consisting of y sequences in any arbitrary order, every sequence S_i is assigned a unique label i which is a dummy label since some of the sequences could belong to the same action performed differently. The motivation for using dummy labels is to force R-NKTM to learn view invariant features rather than features specific to the mocap actions. Thus, action labels are not required while R-NKTM learning or while transferring training and test action descriptors to the shared high-level space using the R-NKTM. The second phase is training, where action descriptors from unknown views are passed through the learned R-NKTM to construct their cross-view action descriptors. Action labels of training data are now required to train the subsequent classifier. In the test phase, view-invariant descriptors of actions observed from unknown and previously unseen views are constructed by forward propagating their view dependent action descriptors through the learned R-NKTM. Any classifier can be trained on the cross-view action descriptors for classification in a view-

0162-8828 (c) 2016 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2017.2691768, IEEE Transactions on Pattern Analysis and Machine Intelligence

invariant manner. We use a simple linear SVM classifier to show the strength of the proposed R-NKTM.

Our R-NKTM learning scheme is based on the observation that similar actions, when observed from different viewpoints, still have a common structure that puts them apart from other actions. Thus, it should be possible to separate action related features from viewpoint related features. The main challenge is that these features cannot be linearly separated. The second challenge comes from learning a non-linear model itself, which requires a large amount of training data. Our solution is to learn the R-NKTM from 2D projections of action trajectories of synthetic 3D human models fitted to real motion capture (mocap) data. By projecting these 3D human models to different views, we can generate a large corpus of synthetic trajectories to learn the R-NKTM. We use k-means to generate a general codebook for encoding the action trajectories. The same codebook is used to encode dense trajectories extracted from real action videos in the training and test phases.

In summary, the major contribution of our approach is that we learn a *single* Robust Non-linear Knowledge Transfer Model (R-NKTM), which can bring any action observed from an unknown viewpoint to its compact high-level representation. Moreover, our method encodes action trajectories using a general codebook learned from synthetic data and then uses the same codebook to encode action trajectories of real videos. Thus, new action classes from real videos can easily be added using the same learned NTKM and codebook. Comparison with eight existing cross-view action recognition methods on five benchmark datasets, including the IXMAS [31], UWA3D Multiview Activity II [35], Northwestern-UCLA Multiview Action3D [21], Hollywood2 [38] and UCF Sports [36] datasets, shows that our method is faster and achieves higher accuracy especially when there are large viewpoint variations.

This paper is an extension of our prior work [37], where we transfer a given action acquired from any viewpoint to its canonical view. Knowledge of the canonical view is required for NKTM learning in [37]. This is a problem because the canonical view is not only action dependent, it is ill-defined. For example, what would be the canonical view of a person walking in a circle? Another limitation of [37] is that cylinders were fitted to the mocap data to approximate human limbs, head and torso. The trajectories generated from such models do not accurately represent human actions. In this paper, we extend our work by removing both limitations. Firstly, we no longer require identification of the canonical view for learning the new R-NKTM and use dummy labels instead. Secondly, we fit realistic 3D human models to the mocap data and hence generate more accurate trajectories. Using 3D human models also enables us to vary and model the human body shape and size. Besides these extensions, we also perform additional experiments on three more datasets namely, the UWA3D Multiview Activity II [35], UCF Sports [36] and Hollywood2 [38]. We denote our prior model [37] by NKTM and the one proposed in this paper by R-NKTM.

2 Related work

The majority of existing literature [1]–[14], [39]–[42] deals with action recognition from a common viewpoint. While these approaches are quite successful in recognizing actions captured from similar viewpoints, their performance drops sharply as the viewpoint changes due to the inherent view dependence of the features used by these methods. To tackle this problem, geometry based methods have been proposed for cross-view action recognition. Rao et al. [30] introduced an action representation to capture the dramatic changes of actions using view-invariant spatio-temporal curvature of 2D trajectories. This method uses a single point (e.g. hand centroid) trajectory. Yilmaz and Shah [24] extended this approach by tracking the 2D points on human contours. Given the human contours for each frame of a video, they generate an action volume by computing point correspondences between consecutive contours. Maximum and minimum curvatures on the spatio-temporal action volume are used as view-invariant action descriptors. However, these methods require robust interest points detection and tracking, which are still challenging problems.

Instead of using geometry constraints, Junejo et al. [32] proposed Self-Similarity Matrix that is constructed by computing the pairwise similarity between any pair of frames. Hankelet [28] represents actions with the dynamics of short tracklets, and achieves cross-view action recognition by finding the Hankelets that are invariant to viewpoint changes. These methods perform poorly on videos acquired from viewpoints that are significantly different from those of the training videos [20], [37].

Recently, transfer learning approaches have been employed to address cross-view action recognition by exploring some form of statistical connections between view-dependent features extracted from different viewpoints. A notable example of this category is the work of Farhadi et al. [18], who employed Maximum Margin Clustering to generate split-based features in the source view, then trained a classifier to predict split-based features in the target view. Liu et. al. [20] learned a cross-view bag of bilingual words using the simultaneous multiview observations of the same action. They represented the action videos by bilingual words in both views. Zheng and Jiang [34] proposed to build a transferable dictionary pair by forcing the videos of the same action to have the same sparse coefficients across different views. However, these methods require feature-to-feature correspondence at the frame-level or video-level during training, thereby limiting their applications.

Li and Zickler [23] assume that there is a smooth virtual path connecting the source and target views. They uniformly sample a finite number of points along this virtual path and consider each point as a virtual view i.e. a linear transformation function. Action descriptors from both views are augmented into cross-view feature vectors, by applying a finite sequence of linear transformations to each descriptor. Recently, Zhang et al. [22] extended this approach by applying an infinite sequence of linear transformations. Although, these methods can operate in

0162-8828 (c) 2016 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2017.2691768, IEEE Transactions on Pattern Analysis and Machine Intelligence

the absence of feature-to-feature correspondence between source and target views, they still require the samples from target view during training.

More recently, Wang et al. [21] proposed cross-view action recognition by discovering discriminative 3D Poselets and learning the geometric relations among different views. However, they learn a separate transformation between different views using a linear SVM solver. Thus, many linear transformations are learned for mapping between different views. For action recognition from unseen views, all learned transformations are used for exhaustive matching and the results are combined with an AND-OR Graph (AOG). This method also requires 3D skeleton data for training, which is not always available. Gupta et al. [15] proposed to find the best match for each training video in large mocap sequences using a Non-linear Circular Temporary Encoding method. The best matched mocap sequence and its projections on different angles are then used to generate more synthetic training data making the process computationally expensive. Moreover, the success of this approach depends on the availability of a large mocap dataset which covers a wide range of human actions [15], [16].

Deep Learning Models: Deep learning models [43]–[45] can learn a hierarchy of features by constructing high-level representations from low-level ones. Due to the impressive results of such deep learning on handwritten digit recognition [44], image classification [46] and object detection [47], several methods have been recently proposed to learn deep models for video based action recognition. Ji et al. [48] extended the deep 2D convolutional neural network (CNN) to 3D where convolutions are performed on 3D feature maps in spatial and temporal dimensions. Simonyan and Zisserman [49] trained two CNNs, one for RGB images and one for optical flow, to learn spatio-temporal features. Gkioxari and Malik [50] extended this approach for action localization. Donahue et al. [51] proposed an end-to-end trainable recurrent convolutional network which processes video frames with a CNN, whose outputs are passed through a recurrent neural network. None of these methods is designed for action recognition in videos acquired from unseen views. Moreover, learning deep models for the task of cross-view action recognition requires a large corpus of training data acquired from multiple views which is typically unavailable and very expensive to acquire and label. These limitations motivate us to propose a pipeline for generating realistic synthetic training data and subsequently learn a Robust Non-linear Knowledge Transfer Model (R-NKTM) which can transfer action videos from any view to a high level space, where actions can be matched in a view-invariant way. Although learned from synthetic data, the proposed R-NKTM is able to generalize to real action videos and achieve state-of-the-art results.

3 Proposed approach

The proposed approach comprises three main stages including feature extraction, Robust Non-linear Knowledge Transfer Model (R-NKTM) learning, and cross-view action description. In the feature extraction stage, synthetic dense trajectories are first generated by fitting 3D human models to mocap sequences and projecting the resulting 3D videos on 2D image planes corresponding to different viewpoints. The 2D dense trajectories are then represented by bag-of-features. In the model learning stage, a deep fully-connected network, called R-NKTM, is learned such that it transfers the view-dependent trajectory descriptors of the same action observed from different viewpoints to a shared high-level virtual view. In the third stage, the dense trajectory descriptors of real action videos are passed through the learned R-NKTM to construct cross-view action descriptors. Details of each stage are given below.

3.1 Feature extraction

Dense trajectories have shown to be effective for action recognition [12]–[15]. Our motivation for using dense trajectories is that they can be easily extracted from conventional videos as well as the 2D projections of synthetic 3D videos generated from mocap data.

3.1.1 Dense trajectories from videos

To extract trajectories from videos, Wang et al. [12], [13] proposed to sample dense points from each frame and track them using displacement information from a dense optical flow field. The shape of a trajectory encodes the local motion pattern. Given a trajectory of length L, a sequence S of displacement vectors $\Delta P_t = (P_{t+1} - P_t) = (x_{t+1} - x_t, y_{t+1} - y_t)$ is formed and normalized as:

$$S = \frac{(\Delta P_t, ..., \Delta P_{t+L-1})}{\sum_{i=t}^{t+L-1} \|\Delta P_i\|}. \tag{1}$$

The descriptor S encodes the shape of the trajectory. To embed appearance and motion information, a spatio-temporal volume aligned with the trajectory is subdivided into a spatio-temporal grid and Histogram of Oriented Gradient (HOG), Histogram of Optical Flow (HOF) and Motion Boundary Histogram (MBH) descriptors are computed in each cell of the grid. The bag-of-features approach is then employed to construct a histogram of visual word occurrences for each descriptor (trajectory shape, HOG, HOF, MBH) separately. The final descriptor is a concatenation of these four histograms. However, unlike [12], [13] we only use the trajectory descriptors since their extraction using multiple viewpoints and scales is computationally efficient as shown in Section 3.1.2. The same process, on the other hand, is computationally very expensive for the remaining three descriptors i.e. HOG, HOF, and MBH. Moreover, using trajectories only is also robust to changes in visual appearance due to clothing and lighting conditions.

3.1.2 Dense trajectories from mocap sequences

Figure 2 gives an overview of the steps involved in generating synthetic dense trajectories using different human body shapes performing a large number of actions rendered from numerous viewpoints. Details are below.

0162-8828 (c) 2016 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2017.2691768, IEEE Transactions on Pattern Analysis and Machine Intelligence

3D human body models: There are different ways to generate 3D human models. For example, Bogo et al. [52] developed the FAUST dataset containing full 3D human body scans of 10 individuals in 30 poses. However, the skeleton data is not provided for these scans. Another way to generate a 3D human model is to use the open source MakeHuman software [53], which can synthesize different realistic 3D human shapes in a predefined pose and also provide the joints positions that can be used for generating human models in different poses. We use this technique for generating the 3D human models in our work.

Fitting 3D human models to mocap sequences: Several approaches [54], [55] have been proposed in the literatures to fit a 3D human model to the motion capture skeleton data of a human subject. For instance, the SCAPE method [55] learns pose and body-shape deformation models from the training scans of different human bodies in a few poses. Given a set of markers, SCAPE constructs a full mesh which is consistent with the SCAPE models, best matches with the given markers and maintains realistic muscle deformations. This method takes approximately 3 minutes to generate each frame. Another example is the MoSh method [54], which estimates an accurate 3D body shape directly from the mocap skeleton without the use of 3D human scans. MoSh is also able to estimate soft-tissue motions from mocap data and subsequently use them to produce animations with subtlety and realism. MoSh requires about 7 minutes to estimate a subject's shape. However, these methods are computationally expensive to apply on a large corpus of mocap sequences. Thus, we use the open source Blender package [56] to fit 3D human models to mocap data. Given a 3D human model generated by the MakeHuman software and a mocap sequence, Blender normalizes the mocap skeleton data with respect to the skeleton data of the human model and then fits the model to the normalized mocap data. This process results in a synthetic but realistic full 3D human body video corresponding to a mocap sequence.

Projection from multiple viewpoints: We deploy a total of 108 synthetic cameras (at distinct latitudes and longitudes) on a sphere surrounding the subject performing an action, as shown in Fig. 3. Given a perspective camera and a frame of a synthetic full 3D human body sequence, we deal with self-occlusions by removing points that are not visible from the given camera viewpoint. First, we perform back-face culling by removing 3D points whose normals face away from the camera. Then, the hidden point removal technique [57] is applied on the remaining 3D points. This gives us a set of visible 3D points corresponding to the given viewpoint. The visible 3D points are projected to the $x-y$ plain using perspective projection resulting in a 2D pointcloud. We repeat this process for all 108 cameras and all frames of the synthetic full 3D human body sequence, thereby, 108 sequences of 2D pointclouds are generated for each synthetic full 3D human action sequence corresponding to a mocap sequence.

Dense trajectory extraction: Since we already have dense correspondence between the 3D human models in each pose, it is straight forward to extract trajectory features from their projected sequence of 2D pointclouds by simply connecting them in time over a fixed horizon of L frames. A sequence S of normalized displacement vectors ΔP_t is calculated for each point (1). We use the same $L = 15$ for both synthetic and real videos. We represent each video (synthetic or real) by a set of motion trajectory descriptors.

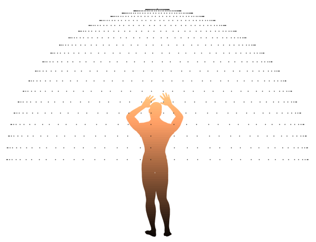
Fig. 3: Virtual cameras are placed on the hemisphere looking towards the center of the sphere to generate 108 virtual views.

We construct a codebook of size $k = 2000$ by clustering the trajectory descriptors with k-means. Note that clustering is performed only over the synthetic trajectory descriptors to learn the codebook. Thus, unlike existing cross-view action recognition techniques [15]–[17], [20], [23] the codebook we learn does not use the trajectory descriptors of real videos from IXMAS [31], UWA3DII [35] or Northwestern-UCLA [21] datasets. We call this the general codebook. We consider each cluster as a codeword that represents a specific motion pattern shared by the trajectory descriptors in that cluster. One codeword is assigned to each trajectory descriptor based on the minimum Euclidean distance. The resulting histograms of codeword occurrences are used as trajectory descriptors. Real action videos are encoded with the same codebook. Recall that unlike dense trajectory-based methods [12], [13] which use HOF, HOG, and MBH descriptors along with trajectories, our method only uses trajectory descriptors.

3.2 Non-linear Knowledge Transfer Model

Besides the limitations of employing linear transformation functions between views, existing cross-view action recognition methods [15], [18], [20]–[23] are either not applicable to unseen views or require augmented training samples which cover a wide range of human actions. Moreover, these methods do not scale well to new data and need to repeat the computationally expensive model learning process when a new action class is to be added. To simultaneously overcome these problems, we propose a Robust Non-linear Knowledge Transfer Model (R-NKTM) that learns to transfer the action trajectory descriptors from all possible views to a shared compact high-level virtual view. Our R-NKTM is learned using synthetic training data and is able to generalize to real data without the need for retraining or fine-tuning, thereby increasing its scalability.

As depicted in Fig. 4, our R-NKTM is a deep network, consisting of Q fully-connected layers (where $Q = 4$) followed by a softmax layer and $p^{(q)}$ units in the q-th fully-connected layer where $q = 1, 2, \cdots, Q$ and $p^{(1)} = 2000, p^{(2)} = 1000, p^{(3)} = 500, p^{(4)} = 2488$. For a given

0162-8828 (c) 2016 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2017.2691768, IEEE Transactions on Pattern Analysis and Machine Intelligence

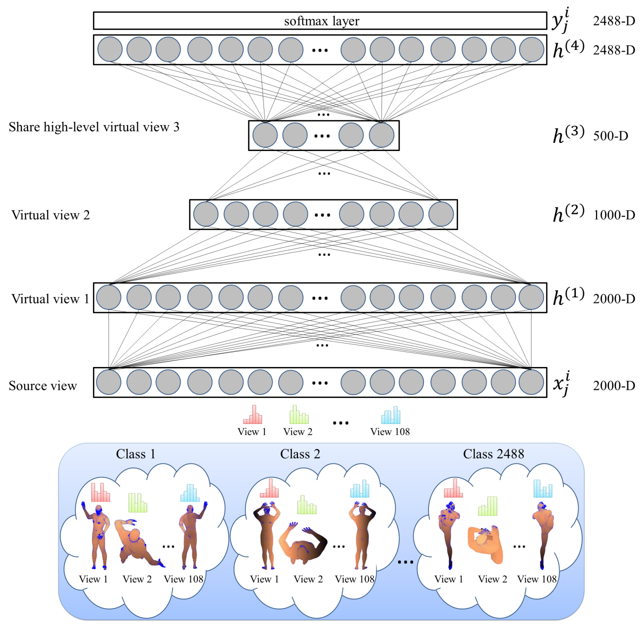

Fig. 4: R-NKTM learns to find a shared high-level view and the corresponding set of non-linear transformations connecting the input views to it.

training sample $\mathbf{x}_j^i \in \mathbb{R}^k$, where $\mathbf{x}_j^i$ is the j-th sample in i-th view, the output of the first layer is $\mathbf{h}^{(1)} = f(\mathbf{W}^{(1)}x_j^i + \mathbf{b}^{(1)}) \in \mathbb{R}^{p^{(1)}}$, where $\mathbf{W}^{(1)} \in \mathbb{R}^{p^{(1)} \times k}$ is a weight matrix to be learned in the first layer, $\mathbf{b}^{(1)} \in \mathbb{R}^{p^{(1)}}$ is a bias vector, and $f(\cdot)$ is a non-linear activation function which is typically a ReLU (Rectified Linear Unit), sigmoid or tangent hyperbolic function. The ReLU function, $f(a) = \max(0, a)$, does not suffer from the gradient vanishing problem like the sigmoid and tangent hyperbolic functions do. Moreover, it has been shown that deep networks can be trained efficiently using the ReLU function even without the need for pre-training [58]. Finally, ReLU generates sparse representations with true zeros that are suitable for exploiting sparsity in the data, which is the case for histogram of codeword occurrences [58]. Therefore, we use ReLU as the activation function in our proposed model. Similarly, the output of the qth layer $\mathbf{h}^{(q)}$ is computed as :

$$\mathbf{h}^{(q)} = f(\mathbf{W}^{(q)}\mathbf{h}^{(q-1)} + \mathbf{b}^{(q)}) \in \mathbb{R}^{p^{(q)}}, \tag{2}$$

where $q = 2, 3, \cdots, Q$. Finally, the output of the last fully-connected layer $\mathbf{h}^{(Q)}$ is passed through a softmax layer to find the appropriate class label.

We use this structure to find a shared high-level space among all possible views. Specifically, in our problem, the inputs to the R-NKTM are synthetic trajectory descriptors corresponding to mocap sequences over different views, while the output is their dummy class labels. Since we use the CMU mocap dataset [59] consisting of 2488 action sequences, the last fully-connected layer has 2488 units whose outputs are given to the softmax layer. The basic idea of this R-NKTM is that regardless of the input view of an unknown action (recall that we do not use the action labels of the mocap sequences), we encourage the output class label of the R-NKTM to be the same for all views of the given action. We explain this idea in the following.

Assume that there is a set of sequential non-linear transformations connecting two views. Thus, there are n sets of transformations connecting n input views to a shared virtual view. We refer to the intermediate transformations as virtual views. Moreover, assume that the videos of the same action observed from different viewpoints share the same high-level feature representation. R-NKTM learns the non-linear transformations to map any action observed from any viewpoint to this shared high level space.

The learning of the proposed R-NKTM is carried out by updating its parameters $\theta_K = \{\theta_\mathbf{W}, \theta_\mathbf{b}\}$, where $\theta_\mathbf{W} = \{\mathbf{W}^{(1)}, \mathbf{W}^{(2)}, \cdots, \mathbf{W}^{(Q)}\}$ and $\theta_\mathbf{b} = \{\mathbf{b}^{(1)}, \mathbf{b}^{(2)}, \cdots, \mathbf{b}^{(Q)}\}$, for minimizing the following objective function over all samples of the input views:

$$E_1(\theta_K; \mathbf{x}_j^i \in \mathbf{X}) = \frac{1}{2nm} \sum_{j=1}^{m} \sum_{i=1}^{n} \ell(z_j, g(\mathbf{x}_j^i)), \tag{3}$$

where n is the number of viewpoints, m is the number of samples in the mocap dataset (for CMU mocap dataset [59]: $m = 2488$), z_j denotes class label of the j-th mocap sequence i.e. $z_j = j$ and ℓ denotes softmax loss function.

Due to the high flexibility of the proposed R-NKTM (e.g. number of units in each layer $p^{(q)}$, θ_K), appropriate settings in the configuration of the R-NKTM are needed to ensure that it learns the underlying data structure. Since the input data $\mathbf{x}_j^i \in \mathbb{R}^{p^{(0)}}$, where $p^{(0)} = 2000$, we discard the redundant information in the high dimensional input data by mapping it to a compact, high-level and low dimensional representation. This operation is performed by 3 fully-connected layers ($\mathbf{h}^{(1)}, \mathbf{h}^{(2)}, \mathbf{h}^{(3)}$) of the R-NKTM.

To avoid over-fitting and improve generalization of the R-NKTM, we add weight decay J_w and sparsity J_s regularization terms to the training criterion i.e. the loss function (3) [60], [61] as given in (4). Large weights cause highly curved non-smooth mappings. Weight decay keeps the weights small and hence the mappings smooth to reduce over-fitting [62]. Weight decay is useful for unsticking hidden units that have developed very large weights early in the training and are either always firmly on or always firmly off. Hidden units that are only rarely active are usually easier to interpret than those that are active about half of the time. However, a better way to allow such units to become useful again is to use a sparsity target [61].

$$E_2(\theta_K; \mathbf{x}_j^i \in \mathbf{X}) = E_1(\theta_K; \mathbf{x}_j^i \in \mathbf{X}) + \lambda_w J_w + \lambda_s J_s, \tag{4}$$

where λ_w and λ_s are the weight decay and sparsity parameters respectively. The J_w penalty tends to decrease the magnitude of the weights $\theta_\mathbf{W} = \{\mathbf{W}_1, \mathbf{W}_2, \mathbf{W}_3\}$:

$$J_w = \sum_{q=1}^{Q} \|\mathbf{W}^{(q)}\|_F^2, \tag{5}$$

where $\|\mathbf{W}^{(q)}\|_F^2$ returns the Frobenius norm of the weight matrix $\mathbf{W}^{(q)}$ of the q-th layer. Let

$$\hat{\rho}_t^{(q)} = \frac{1}{M} \sum_{i=1}^{n} \sum_{j=1}^{m_i} \mathbf{h}_t^{(q)}(\mathbf{x}_j^i) \ , \tag{6}$$

0162-8828 (c) 2016 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2017.2691768, IEEE Transactions on Pattern Analysis and Machine Intelligence

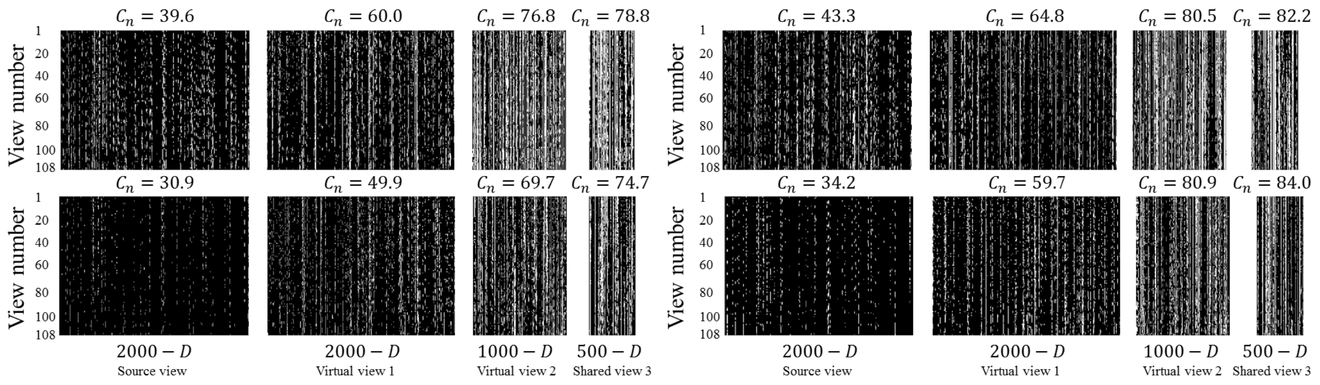

Fig. 5: Visualization of R-NKTM layer outputs for four unseen mocap sequences. Each sequence gives 108 descriptors corresponding to cameras placed at azimuth angles 0° to 340° (20° step) and zenith angles 0°, 10°, 30°, 50°, 70°, 90°. The outputs of the R-NKTM layers (source view $\mathbf{x}_j^i$ and three virtual views $\mathbf{h}^{(1)}$, $\mathbf{h}^{(2)}$, $\mathbf{h}^{(3)}$) are visualized as images. The 108 rows in an image correspond to 108 viewpoints of the same action. The norm of correlation coefficient (C_n) is shown above each image where larger values indicate higher similarity between the rows. Note that as the action descriptors progress through the R-NKTM layers, the similarity of the same action observed from 108 viewpoints increases.

be the mean activation of the t-th unit of the q-th layer (averaged over all the training samples $\mathbf{x}_j^i \in \mathbf{X}$). The J_s penalty forces $\hat{\rho}_t^{(q)}$ to be as close as possible to a sparsity target ρ, and is defined in terms of the Kullback-Leibler (KL) divergence between a Bernoulli random variable with mean $\hat{\rho}_t^{(q)}$ and a Bernoulli random variable with mean ρ as

$$J_s = \sum_{q=1}^{Q} \sum_{t} \rho \log \frac{\rho}{\hat{\rho}_t^{(q)}} + (1-\rho) \log \frac{1-\rho}{1-\hat{\rho}_t^{(q)}} \ . \quad (7)$$

The reasons for using these two regularization terms are twofold. Firstly, not all features are equally important. Secondly, sparsity forces the R-NKTM to find a compact, shared and high-level virtual view, $\mathbf{h}^{(3)}$, by selecting only the most critical features. A dense representation may not learn a good model because almost any change in the input layer modifies most of the entries in the output layer.

Our goal is to solve the optimization problem $E_2(\theta_K; \mathbf{x}_j^i \in \mathbf{X})$ in (4) as a function of $\theta_{\mathbf{W}}$ and $\theta_{\mathbf{b}}$. Therefore, we use stochastic gradient descent through back-propagation to minimize this function over all training samples in the mocap data $\mathbf{x}_j^i \in \mathbf{X}$.

Figure 5 visualizes the output features of the learned R-NKTM layers for four mocap actions that were not used during learning. In each case, a 3D human model was fitted to the mocap sequence and projected from 108 viewpoints. Dense trajectories of each view were computed to get 108 descriptors which were then individually passed through the learned R-NKTM. Figure 5 shows the outputs of each layer as an image. As expected, the outputs of the shared virtual view $\mathbf{h}^{(3)}$ are very similar for all 108 views. Note that we drop the outputs of the last fully-connected $\mathbf{h}^{(4)}$ and softmax layers because they are the 2048 class scores which correspond to dummy labels.

3.3 Cross-View Action Description

So far we have learned an R-NKTM whose input is a synthetic trajectory descriptor corresponding to a mocap sequence fitted with a 3D human model observed from any arbitrary view. The output of the model is the class label which is the same for all views of the sequence. While the appearance of synthetic data is different from real data, the dense trajectories extracted from synthetic and real videos are similar. Therefore, our model is learned from dense trajectories only and is able to extract view-invariant features from the dense trajectories of real data.

Figure 6 shows an overview of the proposed method for extracting cross-view action descriptors from real videos. Given a real human action video, the view-dependent descriptor $\mathbf{x}$ is constructed by extracting dense trajectories from multiple spatial scales of the given video and then building the histogram of codeword occurrences using the learned general codebook as discussed in Section 3.1. Recall that the R-NKTM learns to find a shared high-level virtual view, $\mathbf{h}^{(3)}$, and the intermediate virtual views, $\mathbf{h}^{(1)}, \mathbf{h}^{(2)}$, connecting the input view and the shared virtual view. This means that we have a set of non-linear transformation functions which transfer the view-dependent action trajectory descriptor $\mathbf{x}$ from an unknown view to the shared high-level virtual view. Recall that we remove the last fully-connected $\mathbf{h}^{(4)}$ and softmax layers because these layers correspond to dummy labels which do not provide any useful information for representing real videos.

The output of the shared virtual view $\mathbf{h}^{(3)}$ alone is not sufficient for view invariant representation. Figure 5 shows that the first and the intermediate layers also encode important features shared between the same actions and this shared information is especially large between nearby viewpoints. Thus, the cross-view action descriptor is constructed by concatenating the transformed features into a long feature vector $\left[\mathbf{x}, \mathbf{h}^{(1)}, \mathbf{h}^{(2)}, \mathbf{h}^{(3)}\right]$. This new descriptor implicitly incorporates the non-linear changes from the unknown input view to the shared high-level virtual view.

To perform cross-view action recognition on any real video, we extract its dense trajectories, code them with the general codebook and then pass it through the learned R-NKTM. The view-invariant features from R-NKTM are then used to train a linear SVM classifier. The use of dense trajectories, which are common between the synthetic and real videos, and training the classifier with real videos

0162-8828 (c) 2016 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2017.2691768, IEEE Transactions on Pattern Analysis and Machine Intelligence

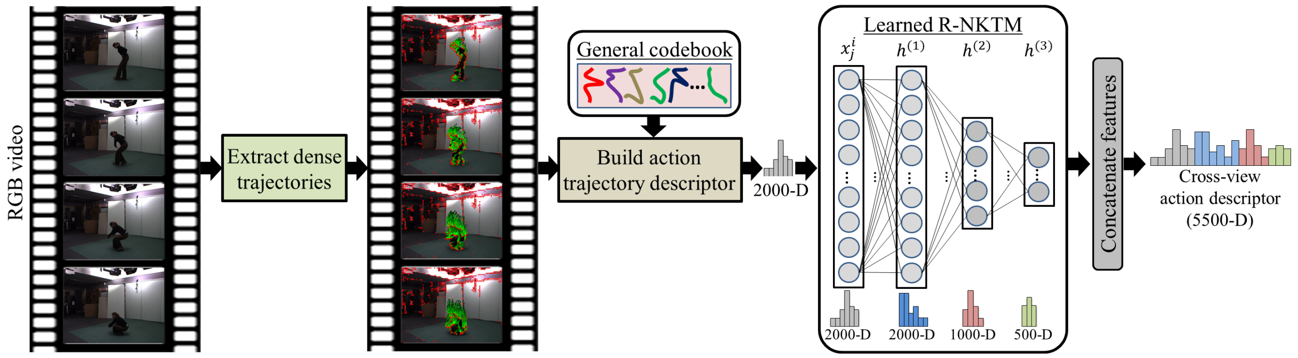

Fig. 6: Extracting cross-view action descriptors from real videos. The view-dependent dense trajectory descriptor $\mathbf{x}$ is extracted from a training or test video and forward propagated through the learned R-NKTM for transfer to the shared high-level virtual view by performing a set of non-linear transformations. The outputs of these transformation functions $\{\mathbf{x}, \mathbf{h}^{(1)}, \mathbf{h}^{(2)}, \mathbf{h}^{(3)}\}$ are concatenated to form a cross-view action descriptor.

bridge the gap between synthetic data used for R-NKTM learning and real data used for testing. For a given sample at test time (i.e. samples from target view), we extract its cross-view features in exactly the same way and feed it to the trained SVM classifier to find its label.

4 Experiments

We evaluate our proposed method on five benchmark datasets including the INRIA Xmas Motion Acquisition Sequences (IXMAS) [31], UWA3D Multiview Activity3DII (UWA3DII) [35], Northwestern-UCLA Multiview Action3D (N-UCLA) [21], Hollywood2 [38] and UCF Sports [36]. We compare our performance to the state-of-the-art action recognition methods including Dense Trajectories (DT) [13], Hankelets [28], Discriminative Virtual Views (DVV) [23], Continuous Virtual Path (CVP) [22], Non-linear Circulant Temporal Encoding (nCTE) [15], AND-OR Graph (AOG) [21], Long-term Recurrent Convolutional Network (LRCN) [51], Action Tubes [50], 3D CNN C3D [48], and Two-stream CNN [49].

We report action recognition results of our method for unseen and unknown views i.e. unlike DVV [23] and CVP [22] we assume that no videos, labels or correspondences from the target view are available at training time. More importantly, unlike existing techniques [15], [21]–[23], [50], [51] we learn our R-NKTM and build the codebook using only synthetic motion trajectories generated from mocap sequences. Therefore, the R-NKTM and the codebook are general and can be used for cross-view action recognition on any action video without the need for retraining or fine-tuning. More precisely, we use the same R-NKTM learned from synthetic data on all datasets without fine-tuning the R-NKTM. In contrast, nCTE [15], DVV [23], CVP [22] and AOG [21] need to learn different models to transfer knowledge across views for different datasets. C3D [48], Two-stream CNN [49], Action Tubes [50] and LRCN [51] require to fine-tune a pre-trained model for each action video dataset.

In addition to the accuracy of our method, we report the recognition accuracy of the NKTM proposed in our prior work [37] which learned to transfer actions observed from unknown viewpoints to their canonical view.

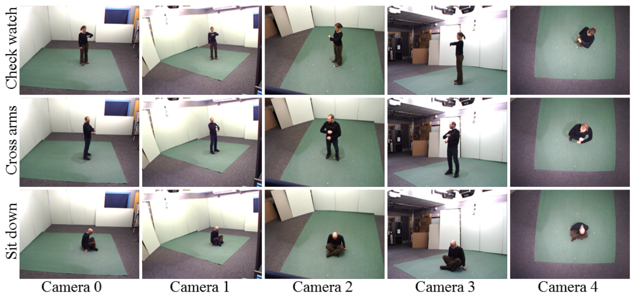

Fig. 7: Sample frames from the IXMAS [31] dataset.

4.1 Implementation Details

For a fair comparison, we pass the dense trajectory descriptors, instead of spatio-temporal interest point descriptors, to DVV [23] and CVP [22]. Moreover, we use 10 virtual views, each with a 30-dimensional features. The baseline results are obtained using publicly available implementations of [13], [15], [23], [28], [48]–[51] or from the original papers. We fine-tuned the C3D [48] and Two-stream CNNs [49] models each time on the training partition of the respective datasets. We used the MatConvNet toolbox [63] for implementing the proposed R-NKTM.

Dense Trajectories Extraction: To generate synthetic dense trajectory descriptors from multiple viewpoints, we use the CMU Motion Capture dataset [59], which contains over 2600 mocap sequences of different subjects performing a variety of daily-life actions. We remove the short sequences containing less than 15 frames since dense trajectories require $L = 15$ minimum frames. The remaining 2488 mocap sequences are used for generating synthetic training data to learn the R-NKTM. Each sequence is treated as a different action and given a unique dummy label. We can generate as many different views from the 3D videos as we desire. Using azimuthal angle $\phi \in \Phi = \{0° : 20° : 340°\}$, and zenith angle $\theta \in \Theta = \{0°, 10°, 30°, 50°, 70°, 90°\}$, we generate ($n = 108$) camera viewpoints and project the 3D videos. Dense trajectories are then extracted from the 2D projections and clustered into $k = 2000$ clusters using k-means to make the general codebook. From real videos,

0162-8828 (c) 2016 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2017.2691768, IEEE Transactions on Pattern Analysis and Machine Intelligence

TABLE 1: Accuracy (%) comparison with state-of-the-art methods under 20 combinations of source (training) and target (test) views on the IXMAS [31] dataset. Each column corresponds to one source|target view pair. The last column shows the average accuracy. The best result of each pair is shown in bold. AOG [21] cannot be applied to this dataset because the 3D joint positions are not provided. Note that DVV and CVP require samples from the target view which are not required by our method.

Source|Target	0|1	0|2	0|3	0|4	1|0	1|2	1|3	1|4	2|0	2|1	2|3	2|4	3|0	3|1	3|2	3|4	4|0	4|1	4|2	4|3	Mean
DT [13]	93.9	64.2	81.8	27.6	87.6	66.4	75.2	22.4	70.0	83.0	73.9	53.3	75.5	77.0	67.0	34.8	42.1	25.8	63.3	48.8	61.7
Hankelets [28]	83.7	59.2	57.4	33.6	84.3	61.6	62.8	26.9	62.5	65.2	72.0	60.1	57.1	61.5	71.0	31.2	39.6	32.8	68.1	37.4	56.4
DVV [23]	72.4	13.3	53.0	28.8	64.9	27.9	53.6	21.8	36.4	40.6	41.8	37.3	58.2	58.5	24.2	22.4	30.6	24.9	27.9	24.6	38.2
CVP [22]	78.5	19.5	60.4	33.4	67.9	29.8	55.5	27.0	41.0	44.9	47.0	41.0	64.3	62.2	24.3	26.1	34.9	28.2	29.8	27.6	42.2
nCTE [15]	**94.8**	69.1	83.9	39.1	90.6	79.7	79.1	30.6	72.1	**86.1**	77.3	62.7	82.4	79.7	70.9	37.9	48.8	40.9	70.3	49.4	67.4
LRCN [51]	66.7	63.6	39.4	16.7	60.6	51.5	36.4	16.7	63.3	27.3	50.0	30.3	45.5	47.9	42.1	15.2	14.8	13.6	18.2	13.9	36.7
Action Tubes [50]	68.5	65.2	24.2	17.0	65.8	57.6	45.5	13.3	63.6	32.7	57.0	26.1	44.2	35.5	63.9	14.5	17.0	14.8	22.1	12.7	38.1
C3D [48]	93.9	65.2	83.9	33.6	90.6	72.0	77.3	22.1	71.0	83.7	78.5	49.4	83.7	77.3	68.5	33.6	39.1	26.1	68.1	39.1	62.8
Two-stream [49]	72.1	71.5	41.0	24.6	63.9	58.5	40.9	22.1	65.8	33.6	55.5	30.6	48.8	53.0	49.4	26.1	16.7	24.6	24.6	27.6	42.5
NKTM	92.7	**84.2**	**83.9**	44.2	**95.5**	77.6	86.1	40.9	82.4	79.4	**85.8**	71.5	82.4	80.9	82.7	**44.2**	**57.1**	48.5	78.8	51.2	72.5
R-NKTM	92.7	80.3	**83.9**	**55.2**	**95.5**	**80.6**	**86.4**	**47.0**	**82.7**	83.6	83.6	**75.5**	**85.8**	**85.2**	**84.9**	**44.2**	56.0	**53.0**	**79.0**	**52.4**	**74.1**

we extract dense trajectories using the method by Wang et al. [13]. We take the length of each trajectory $L = 15$ for both mocap and video sequences. As recommended by [13], we use 8 spatial scales spaced by a factor of $1/\sqrt{2}$ and the dense sampling step size 5 for video samples.

R-NKTM Architecture: There are different recommendations for exploring hyper-parameters (e.g. weight decay, sparsity target, number of units in each layer and number of layers) of deep networks such as manual search, automatic search and combinations of both [60]. In our learning process, we use multi-resolution search. The idea is to test some values from a larger parameter range, select a few best configurations and then test again with smaller steps around these values. To optimize the number of R-NKTM layers, we tested networks with increasing number of layers [64] and stopped where the performance peaked on our validation data. The hidden layer sizes varied in the intervals $[500, 4000]$, the weight decay parameter varied between 0.00001 and 0.01, the learning rates varied between 0.0001 and 0.1. We used a momentum of 0.9, weight decay $\lambda_w = 0.0005$, sparsity parameter $\lambda_s = 0.5$, and sparsity target $\rho = 0.05$.

4.2 IXMAS Dataset

This dataset [31] consists of synchronized videos observed from 5 different views including four side views and a top view. It contains 11 daily-life actions including *check watch*, *cross arms*, *scratch head*, *sit down*, *get up*, *turn around*, *walk*, *wave*, *punch*, *kick*, and *pick up*. Each action was performed three times by 10 subjects. Figure 7 shows examples from this dataset.

We follow the same evaluation protocol as in [15], [23], [28] and verify our algorithm on all possible pairwise view combinations. In each experiment, we use all videos from one camera as training samples and then evaluate the recognition accuracy on the video samples from the 4 remaining cameras. Comparison of the recognition accuracy for 20 possible combinations of training and test cameras is shown in Table 1.

R-NKTM achieves better recognition accuracy than the NKTM [37], which requires to define a same canonical view for all actions. Moreover, the proposed R-NKTM outperforms the state-of-the-art methods on most view pairs and achieves 74.1% average recognition accuracy, that is about 7% higher than the nearest competitor nCTE [15]. Note that our R-NKTM can perform much better (about 10% on average) than the nearest competitor nCTE [15], when camera 4 is considered as either source or target view (see Table 2). As shown in Fig. 7, camera 4 captured videos from the top view, so the appearance of these videos is completely different from the videos captured from the side views (i.e. camera 0 to 3). Hence, we believe that the recognition results on camera 4 are the most important for evaluating cross-view action recognition. Moreover, some actions such as *check watch, cross arms*, and *scratch head* are not available in the mocap dataset. However, our R-NKTM achieves 66.7% average accuracy on these three actions which is about 11% higher than nCTE [15]. This demonstrates that the proposed R-NKTM is able to transfer knowledge across views without requiring all action classes in the learning phase.

Among the knowledge transfer based methods, DVV [23] and CVP [22] did not perform well. The deep learning based methods such as LRCN [51] and Action Tubes [50] achieve low accuracy because they were originally proposed for action recognition from a common viewpoint. DT [13] achieves a high overall recognition accuracy because the motion trajectories of action videos captured from the side views are similar. However, its average accuracy when camera 4 is considered as either source or target view, is over 18% lower than our proposed method.

Figure 8 compares the class specific action recognition accuracies of our proposed R-NKTM with NKTM [37]. R-NKTM achieves higher accuracies for all action classes excluding *check watch*. This demonstrates the effectiveness of our new architecture for cross-view action recognition.

4.3 UWA3D Multiview Activity II Dataset

This dataset [35] consists of a variety of daily-life human actions performed by 10 subjects with different scales. It includes 30 action classes: *one hand waving*, *one hand Punching*, *two hand waving*, *two hand punching*, *sitting down*, *standing up*, *vibrating*, *falling down*, *holding chest*, *holding head*, *holding back*, *walking*, *irregular walking*,

0162-8828 (c) 2016 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2017.2691768, IEEE Transactions on Pattern Analysis and Machine Intelligence

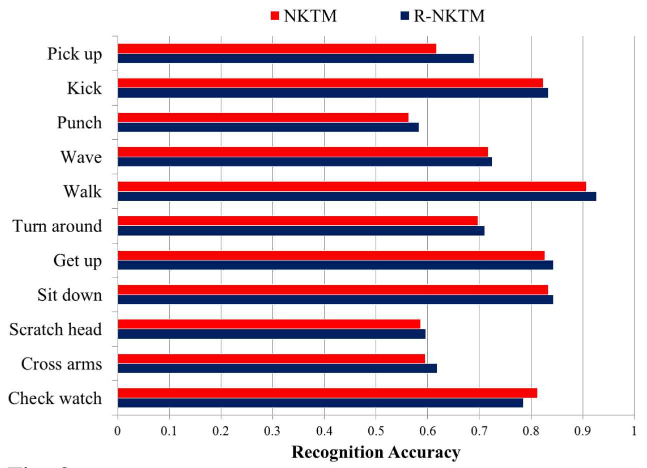

Fig. 8: Per class recognition accuracy of our proposed R-NKTM and NKTM [37] on the IXMAS [31] dataset.

TABLE 2: Average accuracies (%) on the IXMAS [31] dataset e.g. C_0 is the average accuracy when camera 0 is used for training or testing. R-NKTM gives the maximum improvement for the most challenging case, Camera 4 (top view).

Method	C_0	C_1	C_2	C_3	C_4
DT [13]	67.8	66.4	67.6	66.8	39.8
Hankelets [28]	59.7	59.9	65.0	56.3	41.2
DVV [23]	44.7	45.6	31.2	42.0	27.3
CVP [22]	50.0	49.3	34.7	45.9	31.0
nCTE [15]	72.6	72.7	73.5	70.1	47.5
LRCN [51]	46.3	40.1	43.3	36.3	17.4
Action Tubes [50]	45.7	41.7	48.5	37.2	17.2
C3D [48]	70.1	67.9	69.6	67.7	38.9
Two-stream [49]	50.6	46.1	48.7	42.8	24.6
NKTM	77.8	75.2	80.3	74.7	54.6
R-NKTM	**78.4**	**78.0**	**80.7**	**75.8**	**57.8**

lying down, *turning around*, *drinking*, *phone answering*, *bending*, *jumping jack*, *running*, *picking up*, *putting down*, *kicking*, *jumping*, *dancing*, *moping floor*, *sneezing*, *sitting down (chair)*, *squatting*, and *coughing*. Each subject performed 30 actions 4 times. Each time the action was captured from a different viewpoint (front, top, left and right side views). Video acquisition from multiple views was not synchronous thus there are variations in the actions besides viewpoints. This dataset is challenging because of varying viewpoints, self-occlusion and high similarity among actions. For instance, action *drinking* and *phone answering* have very similar motion, but the location of hand in these two actions is slightly different. Also, actions like *holding head* and *holding back* have self-occlusion. Moreover, in the top view, the lower part of the body was not properly captured because of occlusion. Figure 9 shows four sample actions observed from 4 viewpoints.

We follow [35] and use the samples from two views as training data, and the samples from the remaining views as test data. Table 3 summarizes our results. The proposed R-NKTM significantly outperforms NKTM [37] and the state-of-the-art methods on all view pairs. The overall accuracy of the view knowledge transfer based methods such as DVV [23] and CVP [22] is low because motion

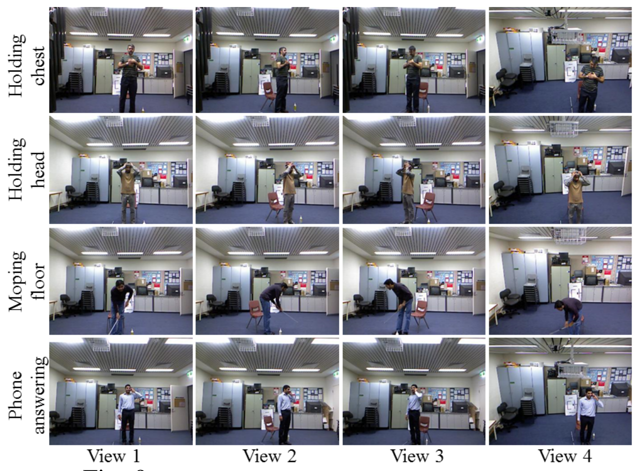

Fig. 9: Sample frames from the UWA3DII [35].

and appearance of many actions look very similar across view changes.

Interestingly, our method achieves 67.5% average recognition accuracy which is about 8% higher than than the nearest competitor nCTE [15], when view 4 is considered as the test view. As shown in Fig. 9, view 4 is the top view which is challenging because the lower part of the subject's body was not fully captured by the camera.

Figure 10 compares the class specific action recognition accuracies of R-NKTM and NKTM [37]. The proposed R-NKTM achieves better recognition accuracy on most action classes. The easiest action to identify is *jumping jack*, with an average accuracy of 95.4% and the hardest is *phone answering* with an average accuracy of 33.3%. These results are not surprising, since *jumping jack* is one of the activities with the most discriminative trajectories while *phone answering* is confused with *drinking* because the motion of these actions is very similar.

Notice that for many actions in the UWA3D Multiview ActivityII dataset such as *holding chest, holding head, holding back, sneezing* and *coughing*, there are no similar actions in the CMU mocap dataset. However, our method still achieves high recognition accuracies for these actions. This demonstrates the effectiveness and generalization ability of our proposed model for representing human actions from unseen and unknown views in a view-invariant space.

4.4 N-UCLA Multiview Action3D Dataset

This dataset [21] contains RGB, depth and skeleton data captured simultaneously by 3 Kinect cameras. The dataset consists of 10 action categories including *pick up with one hand*, *pick up with two hands*, *drop trash*, *walk around*, *sit down*, *stand up*, *donning*, *doffing*, *throw*, and *carry*. Each action was performed by 10 subjects from 1 to 6 times. Fig. 11 shows some examples. This dataset is very challenging because the subjects performed some *walking* within most actions and the motion of some actions such as *carry* and *walk around* are very similar. Moreover, most activities involve human-object interactions.

We follow [21] and use the samples from the first two cameras for training and samples from the remaining camera for testing. The comparison of the recognition accuracy is shown in Table 4. The proposed R-NKTM

0162-8828 (c) 2016 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2017.2691768, IEEE Transactions on Pattern Analysis and Machine Intelligence

TABLE 3: Comparison of action recognition accuracy (%) on the UWA3D Multiview ActivityII dataset. Each time two views are used for training and the remaining ones are individually used for testing. Our method achieves the best performance in all cases.

Sources|Target	$\{1,2\}\|3$	$\{1,2\}\|4$	$\{1,3\}\|2$	$\{1,3\}\|4$	$\{1,4\}\|2$	$\{1,4\}\|3$	$\{2,3\}\|1$	$\{2,3\}\|4$	$\{2,4\}\|1$	$\{2,4\}\|3$	$\{3,4\}\|1$	$\{3,4\}\|2$	Mean
DT [13]	57.1	59.9	54.1	60.6	61.2	60.8	71	59.5	68.4	51.1	69.5	51.5	60.4
Hankelets [28]	46.0	51.5	50.2	59.8	41.9	48.1	66.6	51.3	61.3	38.4	57.8	48.9	51.8
DVV [23]	35.4	33.1	30.3	40.0	31.7	30.9	30.0	36.2	31.1	32.5	40.6	32.0	33.7
CVP [22]	36.0	34.7	35.0	43.5	33.9	35.2	40.4	36.3	36.3	38.0	40.6	37.7	37.3
nCTE [15]	55.6	60.6	56.7	62.5	61.9	60.4	69.9	56.1	70.3	54.9	71.7	54.1	61.2
LRCN [51]	53.9	20.6	43.6	18.6	37.2	43.6	56.0	20.0	50.5	44.8	53.3	41.6	40.3
Action Tubes [50]	49.1	18.2	39.6	17.8	35.1	39.0	52.0	15.2	47.2	44.6	49.1	36.9	37.0
C3D [48]	59.5	59.6	56.6	64.0	59.5	60.8	71.7	60.0	69.5	53.5	67.1	50.4	61.0
Two-stream [49]	63.0	47.1	55.8	60.6	53.4	54.2	66.0	50.9	65.3	55.5	68.0	51.9	57.6
NKTM	60.1	61.3	57.1	65.1	61.6	66.8	70.6	59.5	73.2	59.3	72.5	54.5	63.5
R-NKTM	**64.9**	**67.7**	**61.2**	**68.4**	**64.9**	**70.1**	**73.6**	**66.5**	**73.6**	**60.8**	**75.5**	**61.2**	**67.4**

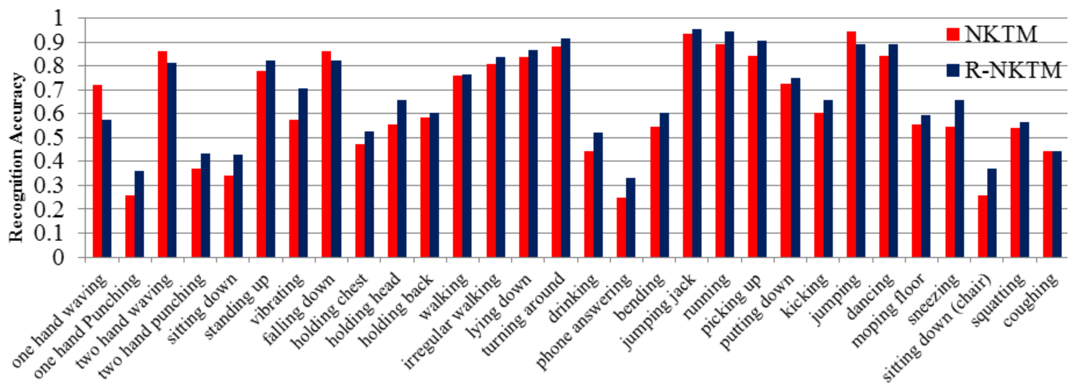

Fig. 10: Per class recognition accuracy of the proposed R-NKTM and NKTM [37] on the UWA3D Multiview ActivityII [35] dataset.

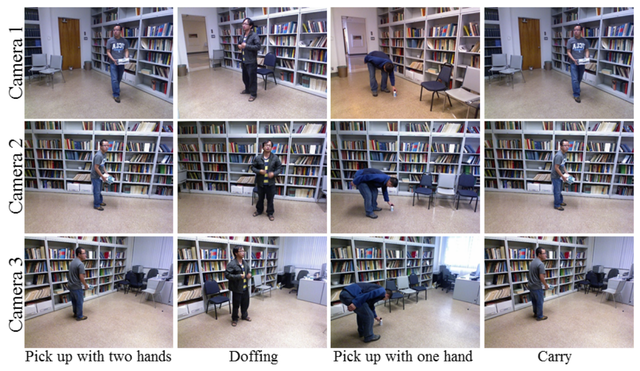

Fig. 11: Sample frames from Northwestern-UCLA Multiview Action3D dataset [21]. Each column shows a different action.

TABLE 4: Accuracy (%) on the N-UCLA Multiview dataset [21]. DVV and CVP use samples from the target view. AOG requires the joint positions of training samples. Our method neither requires target view samples nor joint positions.

Method	Accuracy	Method	Accuracy
DT [13]	72.7	Hankelets [28]	45.2
DVV [23]	58.5	CVP [22]	60.6
nCTE [15]	68.6	AOG [21]	73.3
LRCN [51]	64.7	Action Tubes [50]	61.5
NKTM	75.8	R-NKTM	**78.1**

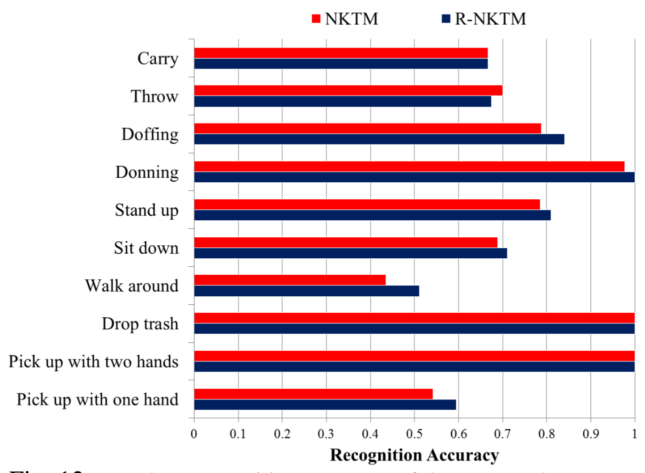

Fig. 12: Per class recognition accuracy of the proposed R-NKTM and NKTM [37] on the N-UCLA Action3D dataset [21].

again outperforms the NKTM [37] and achieves the highest recognition accuracy.

Figure 12 compares the per action class recognition accuracy of our proposed R-NKTM and NKTM [37]. Our method achieves higher accuracy than NKTM [37] for most action classes. Note that a search for some actions such as *donning*, *doffing* and *drop trash* returns no results on the CMU mocap dataset [59] used to learn our R-NKTM. However, our method still achieves 76.8% average recognition accuracy on these three actions which is about 10% higher than nCTE [15]. Moreover, *walk around* and *carry* have maximum confusion with each other because the motion of these actions are very similar.

4.5 Other Datasets

While the focus of the proposed approach is on action recognition from unknown and unseen views, we also

0162-8828 (c) 2016 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2017.2691768, IEEE Transactions on Pattern Analysis and Machine Intelligence

TABLE 5: Comparison of action recognition accuracy on the UCF Sports and Hollywood2 datasets.

Dataset	Method	Only traj.	Combined
UCF Sports	DT [13]	75.2	88.2
	R-NKTM(DT)	**76.7**	**90.0**
	iDT [14]	78.9	92.3
	R-NKTM(iDT)	**82.8**	**93.8**
Hollywood2	DT [13]	47.7	58.3
	R-NKTM	**49.8**	**59.4**
	iDT [14]	51.1	62.2
	R-NKTM(iDT)	**53.3**	**63.0**

evaluate its performance for recognizing actions from previously seen views to have a baseline and to show that our method performs equally good when the viewpoint of the test action is not novel. The evaluation is performed on the UCF Sports dataset [36] containing videos from sports broadcasts in a wide range of scenes. As recommended in [36], we use the Leave-One-Out (LOO) cross-validation scheme. We compare our proposed method to Dense Trajectories (DT) [13] and improved Dense Trajectories (iDT) [14] which are most relevant to our work. Table 5 shows the accuracy of our method in two settings i.e. R-NKTM (DT) where we pass the dense trajectory descriptors through R-NKTM and RNKTM (iDT) where we pass the iDT descriptors through the same R-NKTM. Using only trajectory descriptors, our method achieves 1.5% and 3.9% higher accuracy than DT [13] and iDT [14]. However, combining HOG, HOF, and MBH descriptors with the trajectory descriptors significantly increases the recognition accuracy of of DT [13] and iDT [14] by 13% and 13.4%, respectively. Similarly, adding these features to our cross-view action descriptor significantly improves the accuracy of our method in both settings.

Table 5 also shows the mean average precision of the R-NKTM in both settings on the Hollywood2 dataset [38]. Using only cross-view trajectory descriptors, our method achieves 2.1% and 2.2% higher accuracy than DT [13] and iDT [14] respectively. Combining the appearance descriptors with our cross-view trajectory descriptor further increases the accuracy.

Interestingly, combining the view dependent HOG, HOF and MBH descriptors with our cross-view descriptor also improves the accuracy for the multiview case especially when the difference between the viewpoints is not large. Table 6 shows comparative results of combined descriptors and the cross-view trajectory only descriptors on the IXMAS dataset. The accuracy of most source|target combinations from side views have improved after combining the features. This is because the appearance of these views is quite similar.

4.6 Effects of Concatenating Virtual Views

We evaluate the intermediate performance of our cross-view descriptor by sequentially adding the virtual views. Figure 13(a)-(e) shows the recognition accuracy on IXMAS dataset for all possible source|target view pairs. For most source|target view pairs, the accuracy increases as more virtual views (starting from the first layer of the R-NKTM) are added to the cross-view action descriptor. The maximum incremental gain is obtained when camera 4 (top view) is used as training or test view. The minimum gain is for $0|1$ view pair because the viewpoints of these cameras are very similar. Thus the raw trajectory descriptors already achieve high accuracy. On the other hand as shown in Fig. 13(f)-(i), starting from the last layer of the proposed R-NKTM, the recognition accuracy also increases as more intermediate layer are added to the cross-view action descriptor. Notice that the minimum incremental gain is obtained when camera 4 is used as training or test view which demonstrates that the shared high-level (last fully connected) layer is more robust to viewpoints changes compared to the other layers. Fig. 14 shows that for all source|target view pairs of UWA3DII dataset, the recognition accuracy increases by adding virtual views to the descriptor.

TABLE 7: Computation time (in minutes) including feature extraction on the N-UCLA dataset [21]. Train+1 is the time required to add a new action class after training with 9 classes. Testing time is for classifying 429 action videos.

Method	Train+1	Testing
AOG [21]	780	240
nCTE [15]	19	12
R-NKTM	**0.52**	**12**

4.7 Computation Time

Our technique outperforms the current cross-view action recognition methods on the IXMAS [31], UWA3DII [35] and N-UCLA [21] datasets by transferring knowledge across views using the same R-NKTM learned without supervision (without real action labels). Therefore, compared to existing cross-view action recognition techniques, the proposed R-NKTM is more general and can be used in on-line action recognition systems. More precisely, the cost of adding a new action class using our approach in an on-line system is equal to SVM training. On the other hand, this situation is computationally expensive for most existing techniques especially for our nearest competitors [15], [21] as shown in Table 7. For instance nCTE [15] requires to perform computationally expensive spatio-temporal matching for each video sample of the new action class. Similarly, AOG [21] needs to retrain the AND/OR structure and tune its parameters. Table 7 compares the computational complexity of the proposed method with AOG [21] and nCTE [15]. Compared to AOG [21] and nCTE [15], the training time of the proposed method for adding a new action class is negligible. Thus, it can be used in an on-line system. Moreover, the test time of the proposed method is much faster than AOG [21] and comparable to nCTE [15]. However, nCTE [15] requires 30GB memory to store the augmented samples whereas our model requires 57MB memory to store the learned R-NKTM and the general codebook.

5 Conclusion

We presented an algorithm for unsupervised learning of a Robust Non-linear Knowledge Transfer Model (R-NKTM) for cross-view action recognition. We call it unsupervised

0162-8828 (c) 2016 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2017.2691768, IEEE Transactions on Pattern Analysis and Machine Intelligence

TABLE 6: Effects of combining HOG, HOF, MBH with our proposed cross-view descriptor on the IXMAS [31] dataset

Source|Target	0|1	0|2	0|3	0|4	1|0	1|2	1|3	1|4	2|0	2|1	2|3	2|4	3|0	3|1	3|2	3|4	4|0	4|1	4|2	4|3	Mean
R-NKTM (Traj. only)	92.7	80.3	83.9	**55.2**	95.5	80.6	86.4	**47.0**	**82.7**	83.6	83.6	**75.5**	85.8	85.2	84.9	**44.2**	**56.0**	**53.0**	79.0	**52.4**	**74.1**
R-NKTM (all)	**96.7**	**80.3**	**89.1**	51.5	**96.7**	**80.9**	**88.5**	43.3	79.7	**87.9**	**84.8**	73.9	**86.1**	**87.9**	**87.9**	43.3	54.5	50.3	**84.2**	52.4	**75.0**

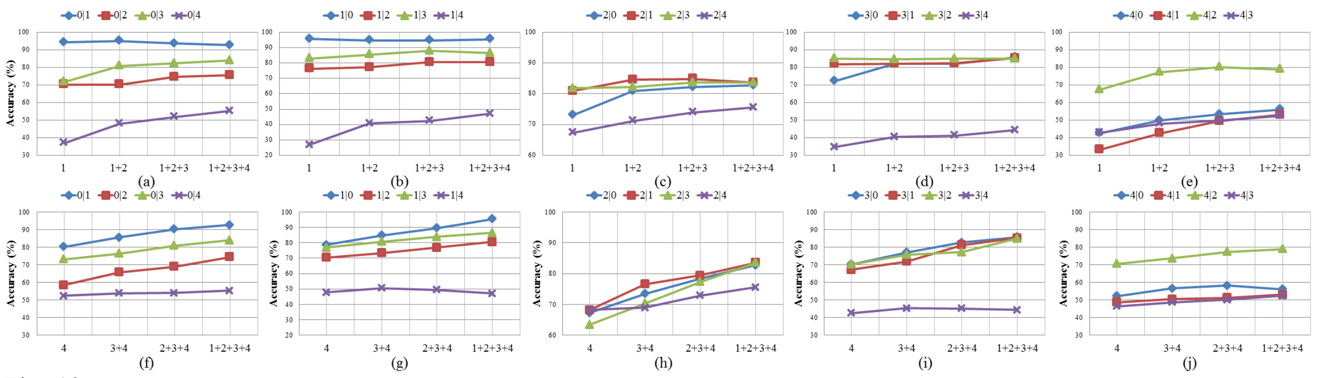

Fig. 13: IXMAS dataset: Effects of adding features from different layers ((a)-(e) starting from the first layer and (f)-(j) starting from the last layer of R-NKTM) to the cross-view action descriptor e.g. $1 + 2 + 3$ means that the descriptor is built by concatenating features from the source view, virtual view 1 and virtual view 2 as shown in Fig. 4.

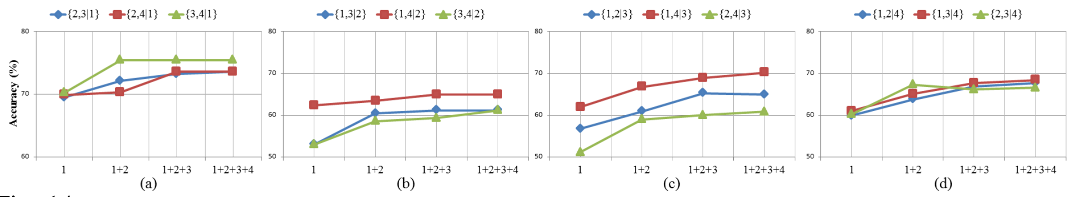

Fig. 14: UWA3DII dataset: Effects of adding features from different layers to the cross-view action descriptor

because the labels used to learn the R-NKTM are just dummy labels and do not correspond to actions that we want to recognize. The proposed R-NKTM is scalable as it needs to be trained only once using synthetic data and generalizes well to real data. We presented a pipeline for generating a large corpus of synthetic training data required for deep learning. The proposed method generates realistic 3D videos by fitting 3D human models to real motion capture data. The 3D videos are projected on 2D plains corresponding to a large number of viewing directions and their dense trajectories are calculated. Using this approach, the dense trajectories are realistic and easy to compute since the correspondence between the 3D human poses is known a priori. A general codebook is learned from these trajectories using k-means and then used to represent the synthetic trajectories for R-NKTM learning as well as the trajectories extracted from real videos during training and testing. The major strength of the proposed R-NKTM is that a single model is learned to transform any action from any viewpoint to its respective high level representation. Moreover, action labels or knowledge of the viewing angles are not required for R-NKTM learning or R-NKTM based representation of real video data. To represent actions in real video sequences, their dense trajectories are coded with the general codebook and forward propagated through the R-NKTM. A simple linear SVM classifier was used to show the strength of our model. Experiments on benchmark multiview datasets show that the proposed approach outperforms existing state-of-the-art.

References

[1] M. Blank, L. Gorelick, E. Shechtman, M. Irani, and R. Basri, “Actions as space-time shapes,” in *ICCV*, 2005.
[2] Z. Lin, Z. Jiang, and L. Davis, “Recognizing actions by shape-motion prototype trees,” in *ICCV*, 2009.
[3] F. Lv and R. Nevatia, “Single view human action recognition using key pose matching and viterbi path searching,” in *CVPR*, 2007.
[4] S. Xiang, F. Nie, Y. Song, and C. Zhang, “Contour graph based human tracking and action sequence recognition,” *Pattern Recognition*, 2008.
[5] P. Dollár, V. Rabaud, G. Cottrell, and S. Belongie, “Behavior recognition via sparse spatio-temporal features,” in *ICCV*, 2005.
[6] I. Laptev, “On space-time interest point,” *IJCV*, 2005.
[7] G. Willems, T. Tuytelaars, and L. Gool, “An efficient dense and scale-invariant spatio-temporal interest point detector,” in *ECCV*, 2008.
[8] H. Rahmani, A. Mahmood, D. Q Huynh, and A. Mian, “HOPC: Histogram of oriented principal components of 3D pointclouds for action recognition,” in *ECCV*, 2014.
[9] J. Liu and M. Shah, “Learning human actions via information maximization,” in *CVPR*, 2008.
[10] J. Liu, Y. Yang, and M. Shah, “Learning semantic visual vocabularies using diffusion distance,” in *CVPR*, 2009.
[11] S. Wu, O. Oreifej, and M. Shah, “Action recognition in videos acquired by a moving camera using motion decomposition of lagrangian particle trajectories,” in *ICCV*, 2011.
[12] H. Wang, A. Klser, C. Schmid, and C. Liu, “Dense trajectories and motion boundary descriptors for action recognition,” *IJCV*, 2013.
[13] H. Wang, A. Kläser, C. Schmid, and C. Liu, “Action recognition by dense trajectories,” in *CVPR*, 2011.
[14] H. Wang and C. Schmid, “Action recognition with improved trajectories,” in *ICCV*, 2013.
[15] A. Gupta, J. Martinez, J. J. Little, and R. J. Woodham, “3D pose from motion for cross-view action recognition via non-linear circulant temporal encoding,” in *CVPR*, 2014.
[16] A. Gupta, A. Shafaei, J. J. Little, and R. J. Woodham, “Unlabelled 3D motion examples improve cross-view action recognition,” in *BMVC*, 2014.
[17] R. Gopalan, R. Li, and R. Chellapa, “Domain adaption for object recognition: An unsupervised approach,” in *ICCV*, 2011.

0162-8828 (c) 2016 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2017.2691768, IEEE Transactions on Pattern Analysis and Machine Intelligence

[18] A. Farhadi and M. K. Tabrizi, "Learning to recognize activities from the wrong view point," in *ECCV*, 2008.
[19] A. Farhadi, M. K. Tabrizi, I. Endres, and D. A. Forsyth, "A latent model of discriminative aspect," in *ICCV*, 2009.
[20] J. Liu, M. Shah, B. Kuipersy, and S. Savarese, "Cross-view action recognition via view knowledge transfer," in *CVPR*, 2011.
[21] J. Wang, X. Nie, Y. Xia, Y. Wu, and S. Zhu, "Cross-view action modeling, learning and recognition," in *CVPR*, 2014.
[22] Z. Zhang, C. Wang, B. Xiao, W. Zhou, S. Liu, and C. Shi, "Cross-view action recognition via a continuous virtual path," in *CVPR*, 2013.
[23] R. Li and T. Zickler, "Discriminative virtual views for cross-view action recognition," in *CVPR*, 2012.
[24] A. Yilmaz and M. Shah, "Action sketch: a novel action representation," in *CVPR*, 2005.
[25] T. Syeda-Mahmood, A. Vasilescu, and S. Sethi, "Action recognition from arbitrary views using 3D exemplars," in *ICCV*, 2007.
[26] D. Gavrila and L. Davis, "3D model-based tracking of humans in action: a multi-view approach," in *CVPR*, 1996.
[27] T. Darrell, I. Essa, and A. Pentland, "Task-specific gesture analysis in real-time using interpolated views," *PAMI*, 1996.
[28] B. Li, O. Camps, and M. Sznaier, "Cross-view activity recognition using hankelets," in *CVPR*, 2012.
[29] V. Parameswaran and R. Chellappa, "View invariance for human action recognition," *IJCV*, 2006.
[30] C. Rao, A. Yilmaz, and M. Shah, "View-invariant representation and recognition of actions," *IJCV*, 2002.
[31] D. Weinland, R. Ronfard, and E. Boyer, "Free viewpoint action recognition using motion history volumes," *CVIU*, 2006.
[32] I. N. Junejo, E. Dexter, I. Laptev, and P. Patrick, "Cross-view action recognition from temporal self-similarities," in *ECCV*, 2008.
[33] X. Yang and Y. Tian, "A survey of vision-based methods for action representation, segmentation and recognition," *CVIU*, 2011.
[34] J. Zheng and Z. Jiang, "Learning view-invariant sparse representations for cross-view action recognition," in *ICCV*, 2013.
[35] H. Rahmani, A. Mahmood, D. Q Huynh, and A. Mian, "Histogram of oriented principal components for cross-view action recognition," *PAMI*, 2016.
[36] M. Rodriguez, J. Ahmed, and M. Shah, "Action MACH a spatio-temporal maximum average correlation height filter for action recognition," in *CVPR*, 2008.
[37] H. Rahmani and A. Mian, "Learning a non-linear knowledge transfer model for cross-view action recognition," in *CVPR*, 2015.
[38] M. Marszaek, I. Laptev, and C. Schmid, "Actions in context," in *CVPR*, 2009.
[39] H. Rahmani, A. Mahmood, D. Q. Huynh, and A. Mian, "Real time action recognition using histograms of depth gradients and random decision forests," in *WACV*, 2014.
[40] H. Rahmani, D. Q. Huynh, A. Mahmood, and A. Mian, "Discriminative human action classification using locality-constrained linear coding," *Pattern Recognition Letters*, 2015.
[41] A. Shahroudy, T.-T. Ng, Q. Yang, and G. Wang, "Multimodal multipart learning for action recognition in depth videos," *PAMI*, 2016.
[42] H. Rahmani, A. Mahmood, D. Huynh, and A. Mian, "Action classification with locality-constrained linear coding," in *ICPR*, 2014.
[43] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, 2006.
[44] G. E. Hinton, S. Osindero, and Y. Teh, "A fast learning algorithm for deep belief nets," *Neural Computation*, 2006.
[45] Y. Bengio, "Learning deep architectures for AI," *Foundations and Trends in Machine Learning*, 2009.
[46] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *NIPS*, 2012.
[47] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *CVPR*, 2014.
[48] S. Ji, W. Xu, M. Yang, and K. Yu, "3D convolutional neural networks for human action recognition," *PAMI*, 2013.
[49] K. Simonyan and A. Zisserman, "Two-stream convolutional networks for action recognition in videos," in *NIPS*, 2014.
[50] G. Gkioxari and J. Malik, "Finding action tubes," in *CVPR*, 2015.
[51] J. Donahue, L. A. Hendricks, S. Guadarrama, M. Rohrbach, S. Venugopalan, K. Saenko, and T. Darrell, "Long-term recurrent convolutional networks for visual recognition and description," in *CVPR*, 2015.
[52] F. Bogo, J. Romero, M. Loper, and M. J. Black, "FAUST: Dataset and evaluation for 3D mesh registration," in *CVPR*, 2014.
[53] "MakeHuman: an open source 3D computer graphics software," http://www.makehuman.org/.
[54] M. Loper, N. Mahmood, and M. J. Black, "MoSh: motion and shape capture from sparse markers," *ACM Transactions on Graphics*, 2014.
[55] D. Anguelov, P. Srinivasan, D. Koller, S. Thrun, J. Rodgers, and J. Davis, "SCAPE: Shape completion and animation of people," *ACM Transactions on Graphics*, 2005.
[56] "Blender: a 3D modelling and rendering package," http://www.blender.org/.
[57] S. Katz, A. Tal, and R. Basri, "Direct visibility of point sets," *ACM Transactions on Graphics*, 2007.
[58] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," in *International Conference on Artificial Intelligence and Statistics*, 2011.
[59] CMU Motion Capture Database, http://mocap.cs.cmu.edu/,.
[60] Y. Bengio, "Practical recommendations for gradient-based training of deep architectures," *Neural Network: Tricks of the Trade*, 2012.
[61] G. Hinton, "A practical guide to training restricted boltzmann machines," *Neural Network: Tricks of the Trade*, 2012.
[62] S. Lawrence, C. L. Giles, and A. C. Tsoi, "What size neural network gives optimal generalization? convergence properties of backpropagation," Tech. Rep., 1996.
[63] A. Vedaldi and K. Lenc, "MatConvNet - Convolutional Neural Networks for MATLAB," in *ACM International Conference on Multimedia*, 2015.
[64] H. Larochelle, D. Erhan, A. Courville, J. Bergstra, and Y. Bengio, "An empirical evaluation of deep architectures on problems with many factors of variation," in *ICML*, 2007, pp. 473–480.

Hossein Rahmani received his B.Sc. degree in Computer Software Engineering in 2004 from Isfahan University of Technology, Isfahan, Iran and the M.S. degree in Software Engineering in 2010 from Shahid Beheshti University, Tehran, Iran. He has published several papers in conferences and journals such as CVPR, ECCV, and TPAMI. He is currently working towards his PhD degree in computer science from The University of Western Australia.

Ajmal Mian completed his PhD from The University of Western Australia in 2006 with distinction and received the Distinguished Doctoral Dissertation Award. He has secured seven nationally competitive grants from the Australian Research Council and the National Health and Medical Research Council. He is currently with the School of Computer Science and Software Engineering, The University of Western Australia. His research interests include computer vision, action recognition, 3D shape analysis, 3D facial morphometrics and machine learning.

Mubarak Shah , the Trustee chair professor of computer science, is the founding director of the Center for Research in Computer Vision at the University of Central Florida (UCF). He was an editor-in-chief of Machine Vision and Applications journal, and an associate editor of ACM Computing Surveys journal. His research interests include video surveillance, visual tracking, human activity recognition, visual analysis of crowded scenes and video registration. He is an ACM distinguished speaker. He is a fellow of the IEEE, AAAS, IAPR, and SPIE.

0162-8828 (c) 2016 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.