Learning a Non-linear Knowledge Transfer Model for Cross-View Action Recognition

Hossein Rahmani, and Ajmal Mian
Computer Science and Software Engineering, The University of Western Australia
hossein@csse.uwa.edu.au, ajmal.mian@uwa.edu.au

Abstract

This paper concerns action recognition from unseen and unknown views. We propose unsupervised learning of a non-linear model that transfers knowledge from multiple views to a canonical view. The proposed Non-linear Knowledge Transfer Model (NKTM) is a deep network, with weight decay and sparsity constraints, which finds a shared high-level virtual path from videos captured from different unknown viewpoints to the same canonical view. The strength of our technique is that we learn a single NKTM for all actions and all camera viewing directions. Thus, NKTM does not require action labels during learning and knowledge of the camera viewpoints during training or testing. NKTM is learned once only from dense trajectories of synthetic points fitted to mocap data and then applied to real video data. Trajectories are coded with a general codebook learned from the same mocap data. NKTM is scalable to new action classes and training data as it does not require re-learning. Experiments on the IXMAS and N-UCLA datasets show that NKTM outperforms existing state-of-the-art methods for cross-view action recognition.

1. Introduction

Action recognition from videos is a significant research problem with applications in human-computer interaction, smart surveillance, and video retrieval. Several techniques have been proposed for discriminative action representation such as 2D shape matching [20, 22, 37], spatio-temporal interest points [5, 16, 27, 35], and trajectory-based representation [30–32, 36]. Especially, dense trajectories-based methods [30–32] have shown impressive results for action recognition by tracking densely sampled points using 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 quite different when observed from different viewpoints [25–27].

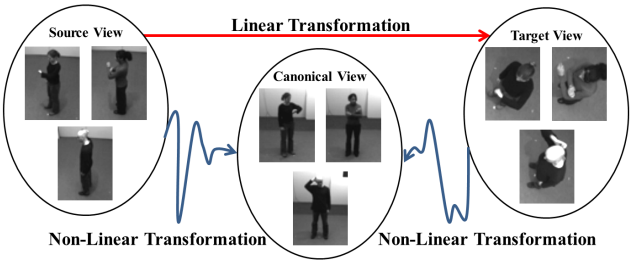

Figure 1: Existing cross-view action recognition techniques [11, 12, 19, 33, 40] connect source and target views with a set of linear transformations that are unable to capture the non-linear manifolds on which real actions lie. Our NKTM finds a shared high-level non-linear virtual path that connects multiple source and target views to the same canonical view.

A practical system should be able to recognize human actions from different unknown and more importantly unseen views. One approach for recognizing actions across viewpoints is to collect data from all possible views and train a separate classifier for each view. However, this approach does not scale well as it requires a large number of labeled samples for each view. To overcome this problem, some techniques infer 3D scene structure and use geometric transformations to achieve view invariance [4, 8, 23, 29, 39]. These methods often require robust joint estimation which is still an open problem in real-world settings. Other methods focus on spatio-temporal features which are insensitive to viewpoint variations [18, 24, 28, 34]. However, the discriminative power of these methods is limited by their inherent structure of view invariant features [38].

Recently, knowledge transfer-based methods [6, 7, 10, 19, 21, 40, 41] have 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. Such methods are either not applicable or perform poorly when recognition is performed on videos from unknown and, more importantly, unseen views. Recently, Wang *et al*. [33] proposed cross-view action recognition by discovering discriminative 3D Poselets and learning the geometric relations among different views. However, they

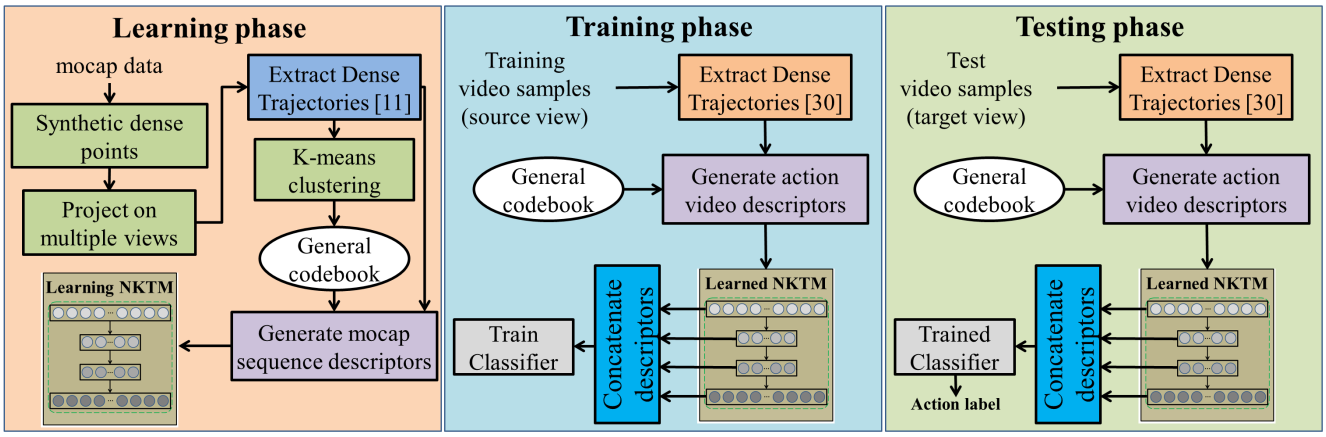

Figure 2: Framework of the proposed algorithm. A single NKTM is learned once only without using action labels. A general codebook is also learned during this phase. The NKTM is used to transfer unknown views to their respective canonical view during training and testing for cross-view action recognition. A linear SVM is used for classification.

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 unknown 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*. [11] 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, this method implicitly assumes that the mocap dataset covers a wide range of human actions.

In this paper, we approach cross-view action recognition as a non-linear knowledge transfer learning problem where knowledge from multiple views is transferred to a single canonical view. Our approach consists of three phases as shown in Fig. 2. The first phase is unsupervised learning where a Non-linear Knowledge Transfer Model is learned. The proposed NKTM is a deep network with weight decay and sparsity constraints which finds a shared high-level virtual path that maps action videos captured from different viewpoints to the same canonical (*i.e*. frontal) view. The strongest point of our technique is that we learn a *single* NKTM for mapping all actions from all camera viewpoints to the same canonical view. Thus, action labels are not required while learning the NKTM or while transforming training and test actions to their respective canonical views using the NKTM. In the training phase, actions from unknown views are transformed to their corresponding canonical views using the learned NKTM. Action labels of training data are now required to train the subsequent classifier. In the final phase, actions from unknown and previously unseen views are transformed to their corresponding canonical views using the learned NKTM. The trained classifier is then used to classify the actions. We used a simple linear SVM to show the strength of the proposed NKTM. However, more sophisticated classifiers can also be used.

Our 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 large training data. Our solution is that we learn the NKTM from action trajectories of synthetic points fitted to mocap data. By projecting these points to different views, we can generate a large corpus of synthetic trajectories to learn the 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 testing phases.

The major contribution of our approach is that we learn a *single* non-linear virtual path between all actions and their respective canonical views irrespective of the initial viewing directions of the actions. Thus, the proposed NTKM can bring any action observed from an unknown viewpoint to its canonical view. Moreover, our method encodes action trajectories using a general codebook learned from synthetic data. The same codebook applies to action trajectories of real videos. Thus, new action classes from real videos can easily be added using the same NTKM and codebook. Comparison with five existing cross-view action recognition methods on two standard datasets shows that our method is faster and achieves higher accuracy especially when there are large viewpoint variations.

2. Proposed Technique

The proposed technique comprises three steps: (1) Feature extraction, (2) Non-linear Knowledge Transfer Model (NKTM) learning, and (3) Cross-view action description.

2.1. Feature Extraction

Dense trajectories have shown to be effective for action recognition [11, 30–32]. Our motivation for using dense trajectories is that they can be easily extracted from videos [30–32] as well as mocap data [11]. To extract trajectories from videos, Wang *et al*. [30, 31] 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 as follows:

$$S = (\Delta P_t, ..., \Delta P_{t+L-1}), \tag{1}$$

and then normalized by the sum of the magnitudes of the displacement vectors:

$$S \leftarrow \frac{S}{\|S\|} = \frac{(\Delta P_t, ..., \Delta P_{t+L-1})}{\sum_{i=t}^{t+L-1} \|\Delta P_i\|}. \tag{2}$$

The descriptor S encodes the shape of the trajectory. To extract dense motion trajectories from mocap data with known body joint positions (see Fig. 3(a)), human limbs are approximated by fitting cylinders over bones *i.e*. connections between joints act as axes. Next, a dense grid is laid on the 3D surface of each cylinder as shown in Fig. 3(b). Given a camera viewpoint, the points on the 3D surface which are not visible from the camera are removed by performing back-face culling and hidden point removal [15]. The remaining 3D points are projected orthographically to the $x - y$ plane as shown in Fig. 3(c)-(e). To extract trajectories, these filtered 2D points are connected in time over a fixed horizon of L frames and a sequence S of normalized displacement vectors ΔP_t is calculated for each point (2).

We represent each mocap sequence (and later videos) by a set of motion trajectory descriptors. We construct a codebook of size $k = 2000$ by clustering the trajectory descriptors with k-means. It is important to note that clustering is performed only over the mocap trajectory descriptors. Thus, unlike existing cross-view action recognition techniques [10–12, 19, 21] the codebook we learn does not use the trajectory descriptors of training or test videos from IXMAS [34] or Northwestern-UCLA [33] 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 motion descriptors. Training and test videos are encoded with the same codebook.

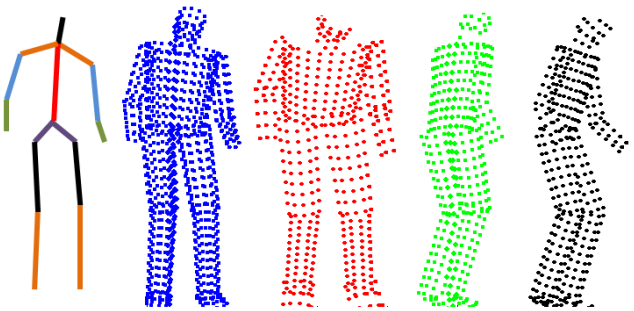

Figure 3: Synthetic data generation from the CMU Motion Capture dataset [1]: (a) mocap skeleton data, (b) human body shape is approximated using cylinders between the joint positions, (c)-(e) sampled body surface points are projected to 3 different views.

2.2. Non-linear Knowledge Transfer Model

Existing cross-view action recognition methods [6, 11, 19, 21, 33, 40] only seek a set of linear transformations connecting source and target views (see Fig. 1) and are thus unable to capture the non-linear manifolds where realistic action videos usually lie on, especially when actions are captured from different views. Furthermore, these approaches are either not applicable [6, 19, 21, 40] to unseen views or require augmented training samples which cover a wide range of human actions [11]. Moreover, these methods do not scale well to new data and need to repeat the computationally expensive learning/training process when a new action class is to be added. To simultaneously overcome these problems, we propose a Non-linear Knowledge Transfer Model (NKTM) that learns a multi-step virtual path between all possible views and their respective canonical view. Thus the input view is mapped to some intermediate virtual views along the non-linear path before constructing the final canonical view.

As depicted in Fig 4, our NKTM is a deep network, consisting of $Q+1$ layers (where $Q = 3$) and $p^{(q)}$ units in the q-th layer (where $q = 1, 2, \cdots, Q$). For a given 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 $\mathbf{h}^{(1)} = s(\mathbf{W}^{(1)} x_j^i + \mathbf{b}^{(1)}) \in \mathbb{R}^{p^{(1)}}$, 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 $s(\cdot)$ is a non-linear activation function (typically a sigmoid or tangent hyperbolic). The output of the first layer $\mathbf{h}^{(1)}$ is used as the input of the second layer. Similarly, the output of the second layer is computed as $\mathbf{h}^{(2)} = s(\mathbf{W}^{(2)}\mathbf{h}^{(1)} + \mathbf{b}^{(2)}) \in \mathbb{R}^{p^{(2)}}$, where $\mathbf{W}^{(2)} \in \mathbb{R}^{p^{(2)} \times p^{(1)}}$, $\mathbf{b}^{(2)} \in \mathbb{R}^{p^{(2)}}$, and $s(\cdot)$ are the weight matrix, bias, and non-linear activation function of the second layer, respectively. The output of the last layer is computed as:

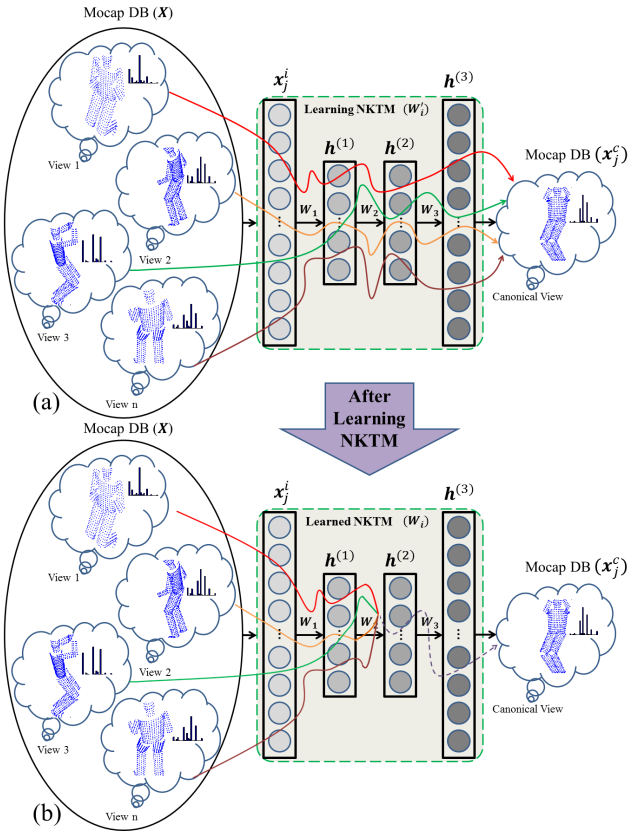

Figure 4: Assume that there are n virtual paths connecting n input views to the canonical view. We show 4 different virtual paths for 4 views in (a). NKTM forces these virtual paths to construct a single non-linear, shared, compact, and high-level virtual path (dotted line in (b)) which connects all views to the canonical view.

$$f(\mathbf{x}_j^i) = \mathbf{h}^{(Q)} = s(\mathbf{W}^{(Q)}\mathbf{h}^{(Q-1)} + \mathbf{b}^{(Q)}) \in \mathbb{R}^{p^{(Q)}} \quad (3)$$

where $f(\cdot)$ is a non-linear transformation function determined by the parameters $\mathbf{W}^{(q)}$ and $\mathbf{b}^{(q)}$ where $q = 1, 2, \cdots, Q$.

We use this structure to learn a single non-linear transformation from all possible views to their respective canonical view. Specifically, in our problem, the inputs to the NKTM are BoW descriptors of mocap action sequences over different views, while the output is BoW descriptors of mocap action sequences from the canonical view. The basic idea of this NKTM is that regardless of the input view, we encourage the output of the NKTM to be close to its canonical view. We explain this idea in the following.

Assume that there is a virtual path which connects any view to its respective canonical view [19]. Therefore, there are n different virtual paths connecting n input views to their canonical view (see Fig. 4(a)). We consider each virtual path as a set of non-linear transformations of action descriptors. Moreover, assume that the videos of the same action over different views share the same high-level feature representation. Given these two assumptions, our objective is to find a virtual path which encodes a shared high-level feature representation in the paths connecting the input views and the canonical view (see Fig. 4(b)). This essentially means that we start with n different virtual paths and the proposed NKTM learning forces them to agree on a single non-linear virtual path.

The learning of the proposed 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 loss function of the reconstruction error, over all samples of the input views:

$$E_1(\theta_K; \mathbf{x}_j^i \in \mathbf{X}) = \frac{1}{2M} \sum_{i=1}^{n} \sum_{j=1}^{m_i} \|\mathbf{x}_j^c - f(\mathbf{x}_j^i)\|^2 \quad (4)$$

where $M = 1/(2n \sum_{i=1}^{n} m_i)$, n is the number of viewpoints and m_i is the number of samples in the i-th view.

However, due to the high flexibility of the proposed NKTM (*e.g.* number of units in each layer $p^{(q)}$, θ_K), appropriate settings in the configuration of the NKTM are needed to ensure that it learns the underlying structure of the data. Since the input data $\mathbf{x}_j^i \in \mathbb{R}^{p^{(0)}}$, where $p^{(0)} = 2000$, we discard the redundant information in the input data by mapping this high dimensional input data to a compact, high-level and low dimensional representation. This operation is performed by $Q - 1$ hidden layers of the NKTM. Then, the low dimensional representation is mapped back to the high dimensional output data ($\mathbf{h}^{(Q)} \in \mathbb{R}^{2000}$) which is the canonical pose independent representation of the input data.

To reduce over-fitting and improve generalization of the NKTM, we add weight decay J_w and sparsity J_s regularization terms to the training criterion *i.e.* the loss function (4) [2, 13]. Large weights cause highly curved non-smooth mappings. Weight decay keeps the weights small and hence the mappings smooth to reduce over-fitting [17]. Similarly, sparsity helps in selecting the most relevant features to improve generalization.

$$E_2(\theta_K; \mathbf{x}_j^i \in \mathbf{X}) = E_1(\theta_K; \mathbf{x}_j^i \in \mathbf{X}) + \lambda_w J_w + \lambda_s J_s \quad (5)$$

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 $\theta_{\mathbf{W}} = \{\mathbf{W}_1, \mathbf{W}_2, \mathbf{W}_3\}$:

$$J_w = \sum_{q=1}^{Q} \|\mathbf{W}^{(q)}\|_F^2, \quad (6)$$

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

$$\hat{\rho}_t^{(q)} = \frac{1}{M} \sum_{i=1}^{n} \sum_{j=1}^{m_i} \mathbf{h}_t^{(q)}(\mathbf{x}_j^i) \tag{7}$$

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 the $\hat{\rho}_t^{(q)}$ 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 follows:

$$\begin{aligned} J_s &= \sum_{q=1}^{Q} \sum_{t} \mathrm{KL}(\rho \| \hat{\rho}_t^{(q)}) \\ &= \sum_{q=1}^{Q} \sum_{t} \rho \log \frac{\rho}{\hat{\rho}_t^{(q)}} + (1-\rho) \log \frac{1-\rho}{1-\hat{\rho}_t^{(q)}} \end{aligned} \tag{8}$$

The reasons for using these two regularization terms are twofold. Firstly, not all features are equally important. Secondly, sparsity forces the NKTM to find a single, shared and high-level virtual path 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 (5) 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 $\mathbf{x}_j^i \in \mathbf{X}$.

2.3. Cross-View Action Description

So far we have learned an NKTM using mocap data to transfer knowledge from different unknown views to their canonical view. This means that we have a set of non-linear transformation functions $\mathbf{H} = \{\mathbf{h}^{(1)}, \mathbf{h}^{(2)}, \cdots, \mathbf{h}^{(Q)}\}$ which transfer an action descriptor $\mathbf{y}_j^i \in \mathbf{Y}$ from the i-th unknown view to its canonical view $\mathbf{y}_j^c$ as follows:

$$\begin{aligned} &\mathbf{A}_S = \mathbf{y}_j^i \rightarrow \mathbf{A}_1 = \mathbf{h}^{(1)}(\mathbf{y}_j^i) \rightarrow \mathbf{A}_2 = \mathbf{h}^{(2)}(\mathbf{h}^{(1)}(\mathbf{y}_j^i)) \rightarrow \cdots \\ &\cdots \rightarrow \mathbf{A}_T = \mathbf{h}^{(Q)}(\mathbf{h}^{(Q-1)}(\cdots(\mathbf{h}^{(2)}(\mathbf{h}^{(1)}(\mathbf{y}_j^i)))\cdots)) \approx \mathbf{y}_j^c. \end{aligned} \tag{9}$$

We describe an action video as alterations of its feature vector along the virtual path (9). As shown in Fig. 5, a cross-view action descriptor $\hat{\mathbf{y}}_j^i$ is constructed by concatenating the transformed features along the virtual path into a long feature vector $\hat{\mathbf{y}}_j^i = [\mathbf{A}_S, \mathbf{A}_1, \mathbf{A}_2, \cdots, \mathbf{A}_T]$. This new descriptor implicitly incorporates the non-linear high-level changes from the i-th to the canonical view. To perform cross-view action recognition on any action video data, we use the samples with their corresponding labels

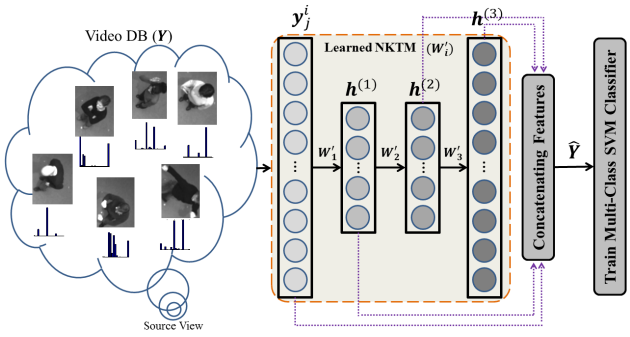

Figure 5: A training action video descriptor $\mathbf{x}_j^i \in \mathbf{X}$ is transformed to its canonical view $\mathbf{x}_j^c$ by performing a set of non-linear transformations $\mathbf{H} = \{\mathbf{h}^{(1)}, \mathbf{h}^{(2)}, \cdots, \mathbf{h}^{(Q)}\}$. We consider each transformation as a virtual view which lies on the non-linear, shared, and high-level virtual path. Therefore, the outputs of these transformation functions $\{\mathbf{A}_S, \mathbf{A}_1, \mathbf{A}_2, \cdots, \mathbf{A}_T\}$ are augmented to form a across-view feature vectors $\hat{\mathbf{x}}_j^i$.

from a source view and extract their cross-view action descriptors $\hat{\mathbf{Y}}$. Then, we train a classifier such as SVM to classify these actions. For a given sample at test time (*i.e.* samples from target view), we simply extract its descriptor using (9) and feed it to the trained classifier to find its label.

3. Experiments

The proposed algorithm was evaluated on two benchmark datasets including the INRIA Xmas Motion Acquisition Sequences (IXMAS) [34] and the Northwestern-UCLA Multiview Action3D (N-UCLA) [33] datasets. We compare our performance to the state-of-the-art cross-view action recognition methods including Hankelets [18], Discriminative Virtual Views (DVV) [19], Continuous Virtual Path (CVP) [40], Non-linear Circulant Temporal Encoding (nCTE) [11], and AND-OR Graph (AOG) [33].

To learn the NKTM, we use the CMU Motion Capture dataset [1] which contains about 2600 mocap sequences of different subjects performing a variety of daily-life actions. It is important to note that we do not use the action labels provided with this dataset. Moreover, we can generate as many different views from the data as we desire. We report action recognition results of our method for unseen and unknown views *i.e.* unlike DVV [19] and CVP [40] we assume that no videos, labels or correspondences from the target view are available at training time. More importantly, unlike Hankelets [18], nCTE [11], DVV [19], CVP [40] and AOG [33] we learn our NKTM and build the motion trajectories codebook using only mocap sequences. Therefore, the NKTM and the codebook are general and can be used for cross-view action recognition on any action videos.

3.1. Implementation Details

It is important to emphasize that unlike existing methods [11, 19, 33, 40], we use the same learned NKTM to eval-

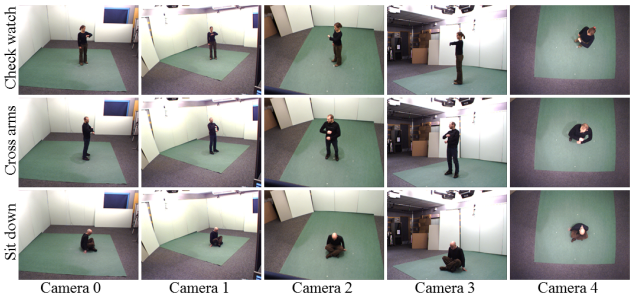

Figure 6: Sample frames from the IXMAS [34] dataset.

uate our algorithm on both IXMAS [34] and N-UCLA [33] datasets. More precisely, nCTE [11], DVV [19], CVP [40] and AOG [33] need to learn different models to transfer knowledge between two views for different datasets. On the other hand, we learn only one model that is applicable to all data and requires no re-learning. For a fair comparison, we feed the same motion trajectory descriptors, instead of spatio-temporal interest point descriptors, to DVV [19] and CVP [40]. Moreover, we use 10 virtual views, each with a 30-dimensional features. We used the code supplied by the authors of nCTE [11] and DVV [19] and carefully implement CVP [40] as its code is not public. The remaining accuracies are reported from their original papers. MATLAB code and video presentation of our method are freely available online.[1]

Dense Trajectories Extraction: The first step is to extract the dense trajectory descriptors and build the BoW histograms of all mocap sequences from all possible view-points. For the sake of computation we project each 3D reconstructed mocap sequence under orthographic projection for a few number ($n = 18$) of view-points (azimuthal angle $\phi \in \Phi = \{0, \pi/3, 2\pi/3, \pi, 4\pi/3, 5\pi/3\}$, and zenith angle $\theta \in \Theta = \{\pi/6, \pi/3, \pi/2\}$). We define $(\phi, \theta) = (\pi, \pi/2)$ as the canonical view. We cluster the mocap trajectories into $k = 2000$ clusters using k-means to make the general codebook. We extract dense trajectories from videos using Wang *et al*. [30] method. We take the length of each trajectory $L = 15$ for both mocap and video sequences and the dense sampling step size 5 for video samples.

Weights Initialization and NKTM Configuration: For NKTM learning, we train a deep network with four layers. The first step in the training of our NKTM is the initialization of $\theta_{\mathbf{W}} = \{\mathbf{W}^{(1)}, \mathbf{W}^{(2)}, \mathbf{W}^{(3)}\}$ and $\theta_{\mathbf{b}} = \{\mathbf{b}^{(1)}, \mathbf{b}^{(2)}, \mathbf{b}^{(3)}\}$. Random initialization and unsupervised pre-training [14] are two popular initialization methods for deep training. In our experiments, due to small number of hidden layers ($= 2$), we use a simple random initialization

[1] http://www.csse.uwa.edu.au/∼ajmal/code.html

Table 1: Average accuracies (%) on the IXMAS [34] dataset e.g. C_0 is the average accuracy when camera 0 was used for training or testing. Each time, only one camera view is used for training and testing. 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
Hankelets [18]	59.7	59.9	65.0	56.3	41.2
DVV [19]	44.7	45.6	31.2	42.0	27.3
CVP [40]	50.0	49.3	34.7	45.9	31.0
nCTE [11]	72.6	72.7	73.5	70.1	47.5
Proposed NKTM	**77.8**	**75.2**	**80.3**	**74.7**	**54.6**

method [3, 9] which initializes the bias $\mathbf{b}^{(q)}$ as $\mathbf{0}$ and the weight matrix $\mathbf{W}^{(q)}$ as the following uniform distribution:

$$\mathbf{W}^{(q)} \sim U\left[-\frac{4\sqrt{6}}{\sqrt{p^{(q)}+p^{(q-1)}}}, +\frac{4\sqrt{6}}{\sqrt{p^{(q)}+p^{(q-1)}}}\right] \quad (10)$$

where $U[-a, +a]$ is the uniform distribution in the interval $(-a, +a)$, and $p^{(q)}$ denotes the number of units in the q-th layer, where $p^{(0)} = 2000$. We use sigmoid as the non-linear activation function and multi-resolution search [2] to find optimal values of the NKTM hyper-parameters. We set the weight decay parameter $\lambda_w = 0.0001$, the sparsity parameter $\lambda_s = 0.5$, and the sparsity target $\rho = 0.05$. The NKTM consists 2000 units at the input/output layers and 1000 units at the two hidden layers. Thus our view invariant action representation is a 6000 dimensional vector.

3.2. IXMAS Dataset [34]

This dataset consists of synchronized videos observed from 5 different views, four side and one top view. It contains 11 daily-life action classes: *check watch*, *cross arms*, *scratch head*, *sit down*, *get up*, *turn around*, *walk*, *wave*, *punch*, *kick*, and *pick up*. Each action is performed three times by 10 subjects. Fig. 6 shows examples from this dataset.

We follow the same evaluation protocol as in [11, 18, 19] 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.

The proposed algorithm outperforms the state-of-the-art methods on most view pairs. It is interesting to note that our method can perform much better (about 7% on average) than the nearest competitor nCTE [11] when camera 4 is considered as either source or target view (see Table 1). As shown in Fig. 6, 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*. cam-

Table 2: Accuracy (%) comparison with state-of-the-art methods under 20 combinations of source (training) and target (test) views on the IXMAS [34] 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 [33] cannot be applied to this dataset at it does not have 3D joint positions. 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
Hankelets [18]	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 [19]	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 [40]	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 [11]	**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
Proposed 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**

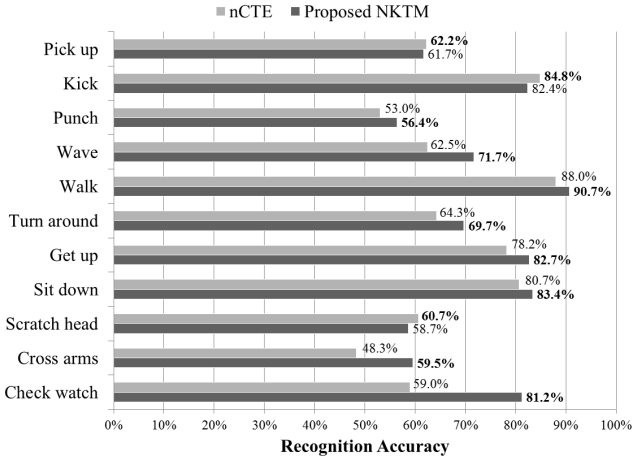

Figure 7: Individual action recognition accuracies of our method and nCTE [11] on IXMAS [34] dataset.

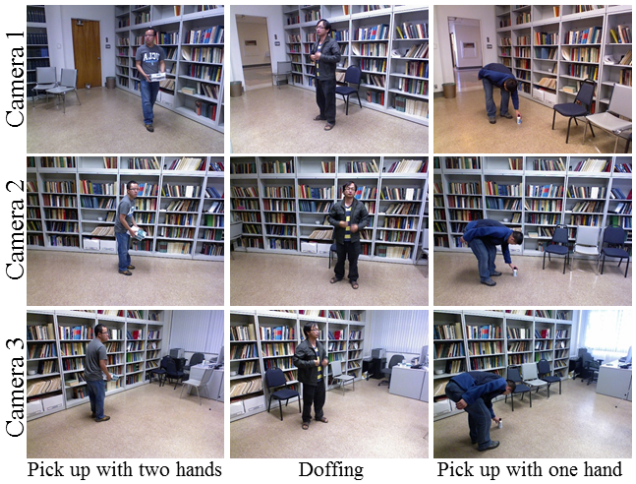

Figure 8: Sample frames from Northwestern-UCLA Multiview dataset [33]. Each column shows a different action.

era 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 mocap dataset [11]. However, our method achieves 66.5% average recognition accuracy on these three actions which is about 11% higher than nCTE [11] (see Fig. 7). This demonstrates that the proposed NKTM is able to transfer knowledge between views without requiring all action classes in the NKTM learning phase. A more thorough comparison of the proposed algorithm with the state-of-the-art methods is shown in Table 2. Our technique outperforms all methods and achieves 72.5% average recognition accuracy which is about 5% higher than the nearest competitor nCTE [11].

3.3. Northwestern-UCLA Dataset [33]

This dataset contains RGB, depth and human skeleton positions 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 is performed by 10 subjects from 1 to 6 times. Fig. 8 shows some examples. We compare our method to four state-of-the-art algorithms (Hankelets [18], DVV [19], CVP [40], nCTE [11]).

We follow [33] and use the samples from two cameras for training and samples from the remaining camera for testing. This dataset is very challenging because the subjects performed some walking within most actions and the motion of some actions such as *pick up with one hand* and *pick up with two hands* are very similar. The comparison of the recognition accuracy for three possible combinations of selecting training and test cameras is shown in Table 4.

Fig. 9 compares the per action class recognition accuracy of our method with nCTE. Our method achieves higher accuracy than nCTE [11] 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 website [1] used to learn our NKTM. However, our method still achieves 76.8% average recognition accuracy on these three actions which is about 10% higher than nCTE [11].

Table. 3 shows the performance of the proposed method after adding the output of each layer to the action descriptor.

3.4. Computation Time

It is interesting to note that our technique outperforms the current cross-view action recognition methods on both

Table 3: The recognition accuracy of our method when the outputs of different number of layers are added to action descriptors.

	1	$1+2$	$1+2+3$	$1+2+3+4$
IXMAS	62.7	68.0	71.9	**72.5**
N-UCLA	61.9	66.8	69.1	**69.4**

Table 4: Accuracy (%) on the N-UCLA Multiview dataset [33] when two views are used for training and one for testing. 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.

{Sources}|Target	$\{1,2\}\|3$	$\{1,3\}\|2$	$\{2,3\}\|1$	Mean
Hankelets [18]	45.2	-	-	-
DVV [19]	58.5	55.2	39.3	51.0
CVP [40]	60.6	55.8	39.5	52.0
AOG [33]	73.3	-	-	-
nCTE [11]	68.6	68.3	52.1	63.0
Proposed NKTM	**75.8**	**73.3**	**59.1**	**69.4**

IXMAS [34] and N-UCLA [33] datasets by transferring knowledge using the same NKTM learned without supervision (without action labels). Therefore, compared to existing cross-view action recognition techniques, the proposed 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 NKTM in an on-line system is equal to training a multi-class SVM classifier. On the other hand, this situation is computationally expensive for most existing techniques [11, 33] as shown in Table 5. For instance nCTE [11] requires to perform computationally expensive spatio-temporal matching for each video sample of the new action class. Similarly, AOG [33] needs to train the AND/OR structure and tune its parameters. Table 5 compares the computational complexity of the proposed method with AOG [33] and nCTE [11]. Compared to AOG [33] and nCTE [11], the training time of the proposed method is negligible. Thus, it can be used in an on-line system. Moreover, the test time of the proposed method is more faster than AOG [33] and comparable to nCTE. However, nCTE requires 30GB memory to store mocap samples.

4. Conclusion

We presented an algorithm for unsupervised learning of a Non-linear Knowledge Transfer Model (NKTM) for cross-view action recognition. The proposed NKTM is scalable as it needs to be trained only once using synthetic data and generalizes well to real data. Moreover, it learns a single model to transform unknown views to their respective canonical views. Action labels are not required to learn the

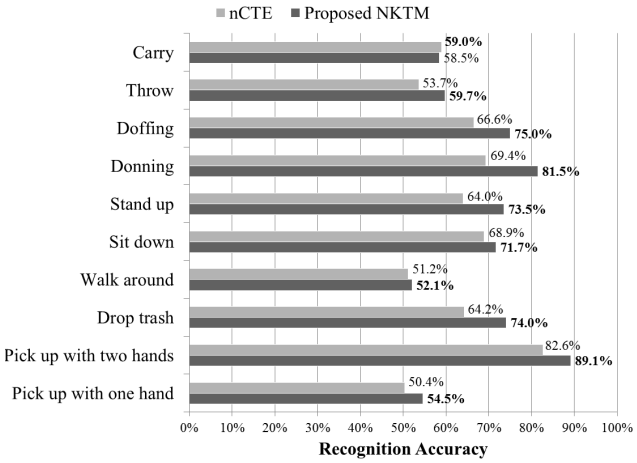

Figure 9: Individual action recognition accuracies of our method and nCTE [11] on Northwestern-UCLA Multiview dataset [33].

Table 5: Computation time in minutes on the N-UCLA dataset when videos from cameras 1 and 2 are used as source views and videos from camera 3 are used as target view. Train+1 means the time required to add a new action class (on-line system) after training with 20 action classes. All timings include feature extraction time. Testing time is for classifying 429 action videos.

Method	Training	Train+1	Testing
AOG [33]	780	780	240
nCTE [11]	600	19	12
Proposed NKTM	26	0.52	12

NKTM. Moreover, knowledge of the viewing angles is not required during training or testing. To represent actions, we extract their trajectories and code them with a general codebook. For learning the NKTM, we extracted dense trajectories of synthetic points fitted to mocap data. We generated a large corpus of synthetic data to learn the NKTM by projecting the synthetic points on 18 viewing directions before trajectory extraction. A general codebook was learned from these trajectories using k-means and then used to represent the synthetic trajectories during learning as well as the trajectories extracted from real videos during training and testing. A linear SVM was used to classify actions. Experiments on two standard datasets show that the proposed approach outperforms existing state-of-the-art.

Acknowledgment

We would like to thank the authors of [11, 15, 19, 30] for making their codes publicly available and the authors of [33] for providing the Northwestern-UCLA dataset. This research was supported by ARC Discovery Grant DP110102399.

References

[1] CMU Motion Capture Database, http://mocap.cs.cmu.edu/,. 3, 5, 7

[2] Y. Bengio. Practical recommendations for gradient-based training of deep architectures. In *Neural Network: Tricks of the Trade*, 2012. 4, 6

[3] Y. Bengio. Practical recommendations for gradient-based training of deep architectures. In *Neural Networks: Tricks of the Trade*, 2012. 6

[4] T. Darrell, I. Essa, and A. Pentland. Task-specific gesture analysis in real-time using interpolated views. In *PAMI*, 1996. 1

[5] P. Dollár, V. Rabaud, G. Cottrell, and S. Belongie. Behavior recognition via sparse spatio-temporal features. In *ICCV*, 2005. 1

[6] A. Farhadi and M. K. Tabrizi. Learning to recognize activities from the wrong view point. In *ECCV*, 2008. 1, 3

[7] A. Farhadi, M. K. Tabrizi, I. Endres, and D. A. Forsyth. A latent model of discriminative aspect. In *ICCV*, 2009. 1

[8] D. Gavrila and L. Davis. 3D model-based tracking of humans in action: a multi-view approach. In *CVPR*, 1996. 1

[9] X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In *AISTATS*, 2010. 6

[10] R. Gopalan, R. Li, and R. Chellapa. Domain adaption for object recognition: An unsupervised approach. In *ICCV*, 2011. 1, 3

[11] 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. 1, 2, 3, 5, 6, 7, 8

[12] A. Gupta, A. Shafaei, J. J. Little, and R. J. Woodham. Unlabelled 3D motion examples improve cross-view action recognition. In *BMVC*, 2014. 1, 3

[13] G. Hinton. A practical guide to training restricted boltzmann machines. In *Neural Network: Tricks of the Trade*, 2012. 4

[14] G. E. Hinton, S. Osindero, and Y.-W. Teh. A fast learning algorithm for deep belief nets. In *Neural Computation*, 2006. 6

[15] S. Katz, A. Tal, and R. Basri. Direct visibility of point sets. In *TOG*, 2007. 3, 8

[16] I. Laptev. On space-time interest point. In *IJCV*, 2005. 1

[17] S. Lawrence, C. L. Giles, and A. C. Tsoi. What size neural network gives optimal generalization? convergence properties of backpropagation. In *Technical Report*, 1996. 4

[18] B. Li, O. Camps, and M. Sznaier. Cross-view activity recognition using hankelets. In *CVPR*, 2012. 1, 5, 6, 7, 8

[19] R. Li and T. Zickler. Discriminative virtual views for cross-view action recognition. In *CVPR*, 2012. 1, 3, 4, 5, 6, 7, 8

[20] Z. Lin, Z. Jiang, and L. Davis. Recognizing actions by shape-motion prototype trees. In *ICCV*, 2009. 1

[21] J. Liu, M. Shah, B. Kuipersy, and S. Savarese. Cross-view action recognition via view knowledge transfer. In *CVPR*, 2011. 1, 3

[22] F. Lv and R. Nevatia. Single view human action recognition using key pose matching and viterbi path searching. In *CVPR*, 2007. 1

[23] F. Lv and R. Nevatia. Single view human action recognition using key pose matching and viterbi path searching. In *CVPR*, 2007. 1

[24] V. Parameswaran and R. Chellappa. View invariance for human action recognition. In *IJCV*, 2006. 1

[25] H. Rahmani, A. Mahmood, D. Huynh, and A. Mian. Action classification with locality-constrained linear coding. In *ICPR*, 2014. 1

[26] H. Rahmani, A. Mahmood, A. Mian, and D. Huynh. Real time action recognition using histograms of depth gradients and random decision forests. In *WACV*, 2014. 1

[27] 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. 1

[28] C. Rao, A. Yilmaz, and M. Shah. View-invariant representation and recognition of actions . In *IJCV*, 2002. 1

[29] T. Syeda-Mahmood, A. Vasilescu, and S. Sethi. Action recognition from arbitrary views using 3D exemplars. In *ICCV*, 2007. 1

[30] H. Wang, A. Klser, C. Schmid, and C. Liu. Action recognition by dense trajectories. In *CVPR*, 2011. 1, 3, 6, 8

[31] H. Wang, A. Klser, C. Schmid, and C. Liu. Dense trajectories and motion boundary descriptors for action recognition. In *IJCV*, 2013. 1, 3

[32] H. Wang and C. Schmid. Action recognition with improved trajectories. In *ICCV*, 2013. 1, 3

[33] J. Wang, X. Nie, Y. Xia, Y. Wu, and S. Zhu. Cross-view action modeling, learning and recognition. In *CVPR*, 2014. 1, 3, 5, 6, 7, 8

[34] D. Weinland, R. Ronfard, and E. Boyer. Free viewpoint action recognition using motion history volumes. In *CVIU*, 2006. 1, 3, 5, 6, 7, 8

[35] G. Willems, T. Tuytelaars, and L. Gool. An efficient dense and scale-invariant spatio-temporal interest point detector. In *ECCV*, 2008. 1

[36] 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. 1

[37] S. Xiang, F. Nie, Y. Song, and C. Zhang. Contour graph based human tracking and action sequence recognition. In *Pattern Recognition*, 2008. 1

[38] X. Yang and Y. Tian. A survey of vision-based methods for action representation, segmentation and recognition. In *CVIU*, 2011. 1

[39] A. Yilmaz and M. Shah. Action sketch: a novel action representation. In *CVPR*, 2005. 1

[40] 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. 1, 3, 5, 6, 7, 8

[41] J. Zheng and Z. Jiang. Learning view-invariant sparse representations for cross-view action recognition. In *ICCV*, 2013. 1