Arbitrary View Action Recognition via Transfer Dictionary Learning on Synthetic Training Data

Jingtian Zhang, *Student Member, IEEE,* Lining Zhang, *Member, IEEE,*
Hubert P. H. Shum, *Member, IEEE,* Ling Shao, *Senior Member, IEEE*

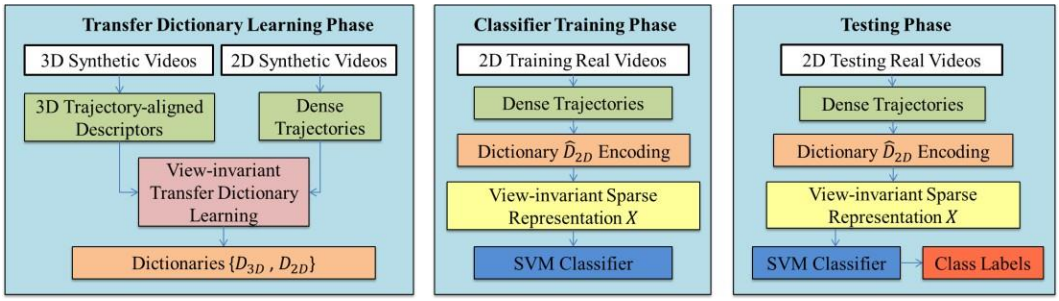


Figure 1. Overview of our view-invariant human action recognition system

***Abstract*—Human action recognition is an important problem in robotic vision. Traditional recognition algorithms usually require the knowledge of view angle, which is not always available in robotic applications such as active vision. In this paper, we propose a new framework to recognize actions with arbitrary views. A main feature of our algorithm is that view-invariance is learned from synthetic 2D and 3D training data using transfer dictionary learning. This guarantees the availability of training data, and removes the hassle of obtaining real world video in specific viewing angles. The result of the process is a dictionary that can project real world 2D video into a view-invariant sparse representation. This facilitates the training of a view-invariant classifier. Experimental results on the IXMAS and N-UCLA datasets show significant improvements over existing algorithms.**

I. Introduction

Machine vision is an important topic in robotics research [1, 2]. In particular, there is a large body of research related to human action recognition, due to its potential applications in developing robots that can understand human behaviors and provide the right assistance.

The main challenge of developing human action recognition techniques in robotics is that the viewpoint from a robot is usually unconstrained. In many domains such as active vision and human machine interaction, the robots have very little information about its viewpoint, as well as the relative direction of the human that performs the action. Therefore, traditional action recognition algorithms that require a predefined view angle fall short in producing satisfactory results. Traditional research in action recognition relies on visual appearance including shape features [3, 4, 5, 6], space-time patterns [7], interest point-based representations [8, 9], as well as motion optical flow patterns [4, 9]. While these features are powerful in recognizing actions from a particular view, visual appearance can be very different from different view angles. As a result, a system trained with a particular view angle may perform poorly in run-time when the viewpoint changes.

To tackle the problem, cross-view action recognition algorithms are proposed. The main idea is to transfer the knowledge from one view to another, allowing a system to recognize actions from a view that it has not been trained on. Li et al. presented a dynamics-based feature called hankelet that can capture the invariant property in viewpoint change using short tracklets for cross-view recognition [21]. Wang et al. used an AND-OR graph representation to compactly express the appearance and motion variance during viewpoint changes [22]. Zhang et al. constructed a continuous virtual path between source view and target view to facilitate cross-view recognition [23]. The weakness of these algorithms is that view information is needed during the training process. However, such information may be difficult to obtain in robotics.

Previous attempts to realize arbitrary view action recognition result in different levels of success. Weinland et al. proposed to recognize human actions by estimating 3D exemplars from a single 2D view angle using the hidden Markov model [10]. However, reconstructing these 3D exemplars from a single view is unreliable. Also, detailed action information may be lost as only discrete samples of silhouette information are used. Yan et al. presented 4D (i.e., 3D spatial and 1D temporal dimensions) action features using time ordered 3D reconstruction of the actors from multi-view video data [11]. However, the recognition accuracy depends heavily on the performance of the 3D reconstruction, and the framework requires training data to be captured in carefully designed viewpoints. Gupta et al. proposed to project the 3D motion capture sequence in the 2D space and explore a best match for each training video using non-linear circular

J. Zhang, L. Zhang, Hubert P. H. Shum and L. Shao are with the Department of Computer Science and Digital Technologies, Northumbria University, Newcastle Upon Tyne, NE1 8ST, U.K. Email: {jingtian.zhang, hubert.shum, lining.zhang, ling.shao}@northumbria.ac.uk.

temporary encoding [12]. However, since discrete 2D projection instead of full 3D information is used for training, the accuracy depends on the number of projected views. Ideally, we would like to have a framework that can provide easy-to-obtain training data, and preform robustly in run-time.

In this paper, we propose a new view-invariant transfer dictionary learning framework to deal with the problem of human action recognition from arbitrary view angles. Our framework utilizes synthetic 3D and 2D training video in order to transfer view-invariant knowledge from 3D to 2D. Such a process does not depend on any real world video and training data can be synthesized effectively. The view-invariant knowledge is represented as a dictionary, which can project real world 2D video into a view-invariant sparse representation. This allows us to train a view-invariant action classifier using a small number of real world 2D video sequences, in which the view-information is not annotated. Experimental results show that our system achieves better accuracy compared with previous work in arbitrary view and cross-view action recognition.

This paper has three main contributions:

- We propose a new transfer dictionary learning framework that utilizes synthetic 2D and 3D training videos to learn a dictionary that can project a real world 2D video into a view-invariant sparse representation, which allows us to train an action classifier for arbitrary views.
- We propose a new 3D feature called Histogram of 3D Velocity (HO3V), which is a 3D equivalent of the state-of-the-art 2D feature *dense trajectories* [9].
- We introduce the first synthetic 2D and 3D training dataset for view-invariant transfer dictionary learning. The database are released for public usage.

The rest of this paper is organized as follows. In Section II, we give an overview of our view-invariant human action recognition frame. In Section III, we present the synthesis and feature extraction process on our 2D and 3D video data. Section IV provides the details of our view-invariant dictionary learning algorithm. Section V presents the experimental results, and Section IV concludes the paper.

II. System Overview

In our system, we synthesize 3D video sequences using motion capture data. $Y_{3D} = [Y_{3D}{}^1, \ldots, Y_{3D}{}^k] \in \mathbb{R}^{m \times k}$ denotes k m-dimensional features extracted from a source 3D video. The synthesized 3D video is projected into different viewpoints to create synthesized 2D videos. $Y_{2D} = [Y_{2D}{}^1, \ldots, Y_{2D}{}^k] \in \mathbb{R}^{n \times k}$ denotes the k n-dimensional features extracted from the target synthetic 2D videos. We train 3D and 2D dictionaries simultaneously from the synthetic data, which are represented as $D_{3D} = [d_{3D}{}^1, \ldots, d_{3D}{}^N] \in \mathbb{R}^{m \times N}$ and $D_{2D} = [d_{2D}{}^1, \ldots, d_{2D}{}^N] \in \mathbb{R}^{n \times N}$. Such dictionaries can project the corresponding video features into a view-invariant sparse representation $X = [X^1, \ldots, X^k] \in \mathbb{R}^{N \times k}$, which is used to train an SVM-based action classifier.

As illustrated in Figure 1 left, in the transfer dictionary learning phase, we learn the dictionaries D_{3D} and D_{2D} simultaneously from synthetic 3D video and synthetic 2D video data. Records belonging to same action class in both 3D and 2D data are encouraged to share the same sparse representation $X \in \mathbb{R}^N$. This allows us to obtain a target dictionary D_{2D} that maps 2D videos to view-invariant 3D videos. It is used to encode a 2D video into a view-invariant representation

Then, as illustrated in Figure 1 middle, in the classifier training phase, we use the learned dictionary D_{2D} to encode the features extracted from real 2D videos. The representation is not sensitive to view changes as the dictionary is trained with 3D video data, and is used to train an SVM classifier.

Finally, in the testing phase as illustrated in Figure 1 right, given any real 2D video, we apply D_{2D} to encode the features into view-invariant sparse representation for classification. Due to the use of the 3D to 2D correlation dictionary, our system can identify actions from an arbitrary view.

III. Video Synthesis and Feature Extraction

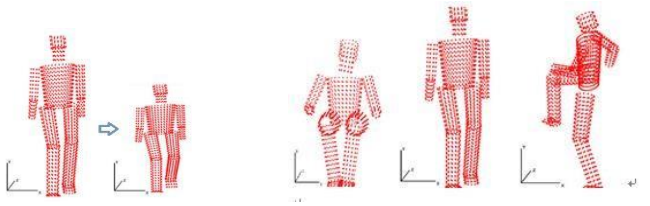
In this section, we explain how we synthesize 3D videos and project them to generate synthetic 2D videos. We then explain how we extract a corresponding set of features in the 3D and 2D space $\{Y_{3D}, Y_{2D}\}$, respectively.

A. Synthesis of 3D and 2D Video

Here, we explain the process of synthesizing 3D and 2D video data. 3D videos provide 3D features that are invariant to viewpoint changes, and 2D videos are used to simulate real videos.

In our system, we implement a design to balance computational cost and system performance. For 3D videos, we utilize simplified cylindrical models for computational efficiency. However, we utilize high quality character meshes for 2D videos such that the extracted features are comparable to that of real 2D videos.

To synthesize the 3D motion models, we utilize the motion capture data from Carnegie-Mellon Graphics Lab and the Truebones dataset [13]. The motions are represented with captured 3D position information of body joints over times with 25 frames per second (FPS). Following [12], we use cylinders to model body parts and represent surface information.



(a) (b)

Figure 2. Synthesizing 3D video: (a) Motion retargeting (b) Example postures from synthetic 3D video

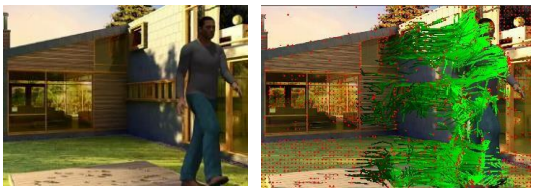
The advantage of using 3D motion data is that we can apply motion retargeting to synthesize the motion performed by characters of different body sizes. As illustrated in Figure 2(a), the motion of a full size character is retargeted into a small size character. During the process, the scale of movement such as step size is scaled. Figure 2(b) shows some postures captured from 3D synthetic videos.

We use a similar process to produce synthetic 2D videos. We first synthesize 3D videos using motion data, and project them uniformly in different viewpoints. Here, in order to enhance the realism of the produced 2D videos such that they can correspond better with real 2D videos, we use high quality character meshes instead of cylinders to model a character's body. Figure 3(a) shows example frames of a generated 2D video, and Figure 3(b) shows the same frames from different viewpoints. Notice that in our system, we do not require any information about the viewpoints.

Figure 3. (a) Example frames of a synthetic 2D video. (b) 2D projections from different viewpoints.

B. *Trajectory-Aligned Descriptor*

Here, we describe the process of extracting features in 2D and 3D videos. 2D video features are extracted using dense trajectories [9]. We then propose a new 3D feature descriptor that is logically similar to the 2D dense trajectories, which we call Histogram of 3D Velocity (HO3V).



(a) (b)

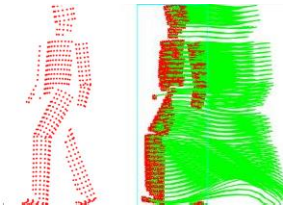
Figure 4. (a) Synthesized 2D video (b) Extracted dense trajectories (red points are interest points, green curves are trajectories)

For both 2D synthetic videos and 2D real videos (which are used for classifier training and testing in Section IV), we employ state-of-the-art action representation *dense trajectories* [9] for feature extraction. It describes both holistic and local information of the motion by combining dense sampling and trajectory tracking. Specifically, it consists of a set of low-level descriptors, including trajectory descriptor, Histogram of Oriented Gradients (HOG), Histogram of Optical Flow (HOF) and Motion Boundary Histogram (MBH). Among them, HOG can extract the static appearance of the videos while HOF and MBH can extract the motion information. Figure 4 shows an example of dense trajectories extracted from a video.

An advantage of synthetic 3D videos is that vertices geometry on the cylinder surface, as well as their correspondence across frames, are available. We first sample interest points from the vertices as shown in Figure 5(a). For each point, we extract the motion trajectory across frames. That is, (P_t, P_{t+1}, P_{t+2}, ...), where P_t is the 3D Cartesian coordinate of the vertex at frame t. We then define a normalized trajectory as:

$$Tr' = \frac{(\Delta P_t, \Delta P_{t+1}, \ldots, \Delta P_{t+L-1})}{\sum_{j=t}^{t+L-1} \|\Delta P_j\|} \quad (1)$$

where L is a user-defined value that represents the number of frames to be considered in a trajectory, and $\Delta P_t = (P_{t+1} - P_t)$ indicates the displacement across two frames. The denominator is the total length of the trajectory, which is used for normalization. Figure 5(b) shows an example of the extracted trajectories.



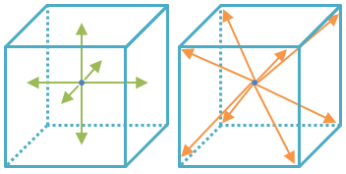
(a) (b)

Figure 5. 3D trajectory-aligned descriptor: (a) Uniformly-sampled interest points in a 3D video (b) 3D extracted trajectories

Our transfer learning involves transferring 3D features to 2D ones, and hence it is preferable that both 3D and 2D features have similar logical meanings. We therefore propose a new feature called Histogram of 3D Velocity (HO3V), which is a velocity field and is logically similar to HOF in 2D dense trajectories. We first define the velocity of a vertex as:

$$V_t = \frac{\Delta P_t}{1/FPS}, \quad (2)$$

where FPS means frame per second of the 3D video, which is set as 25 in our experiments.



(a) (b)

Figure 6. The 14 velocity bins visualized with a 3D cube: (a) 6 directions pointing towards the faces of the cube (b) 8 directions pointing towards the corners of the cube

We quantize the 3D velocity orientations into 14 bins $H(h_1, h_1, \ldots, h_{14})$ as shown in Figure 6. HO3V is defined as the binned histogram along each vertex trajectory:

$$h_i = \frac{\sum_{t \in T_i} \|V_t\|}{\sum_{j=t}^{t+L-1} \|V_t\|}, \quad (3)$$

where T_i is a set that contains the frame's number in which the velocity direction of the interest point belongs to i on a L -frame trajectory. $\|V_t\|$ is the magnitude of the velocity. Notice that the magnitudes of the 3D velocity are used for weighting.

IV. View-invariant Action Classification

In this section, we explain how we apply synthesized 3D and 2D video data to enhance action recognition in real 2D videos through dictionary learning.

A. View-invariant Dictionary Learning

Here, we introduce the basic theory of dictionary learning [15], and explain how we learn the view-invariant transfer dictionary for the 3D and 2D synthetic videos. Dictionary learning generates a sparse representation for a high dimensional signal using linear projection with a projection dictionary. Let $Y \in \mathbb{R}^n$ denote an n-dimensional input signal that can be reconstructed by the N-dimensional projection coefficient $X \in \mathbb{R}^N$ via a linear projection dictionary $D = [d^1, \dots, d^i, \dots, d^N] \in \mathbb{R}^{n \times N}$. To obtain an over-completed dictionary, N should be much larger than n. Assuming the reconstruction error to be $E(X)$, the projection process is formulated as:

$$Y = DX + E(X). \quad (4)$$

The objective function is defined as:

$$\langle X, D \rangle = \arg\min_{X,D} \|Y - DX\|_2^2 \ s.t. \|X\|_0 \le T, \quad (5)$$

where $\|Y - DX\|_2^2$ denotes the reconstruction error and $\|X\|_0 \le T$ is the sparsity constraint.

We design a transfer dictionary learning system to transfer the view-invariance of the synthetic 3D videos to the synthetic 2D videos. We train two dictionaries simultaneously, with one for 3D (i.e., source - D_{3D}) and one for 2D (i.e., target - D_{2D}). The main idea is that we optimize the dictionaries such that the same action in both 3D and 2D videos has similar sparse representations, as visualized in Figure 7. Upon successful training, D_{2D} is able to project the feature vector of a 2D video into a sparse representation that is similar to that of a 3D video. In other words, such a sparse representation is view-invariant.

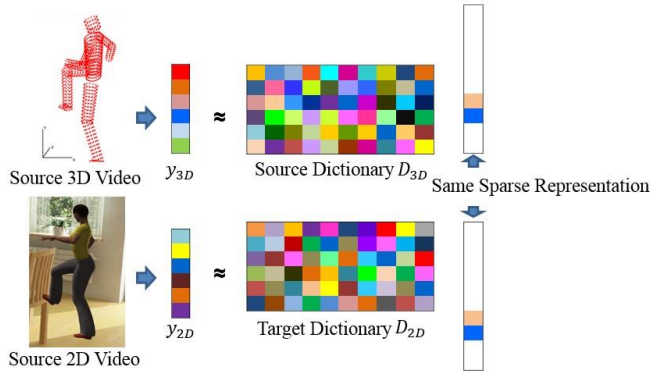


Figure 7. Optimizing the 3D (source) and 2D (target) dictionaries to encourage that the same action in synthetic 3D and 2D videos has the same sparse representations.

We divide the dictionary into some disjoint subsets and each subset is used for one action category. 3D and 2D videos with the same action category are therefore represented with the same subset of the dictionary. Those with different action categories are represented with disjoint subsets of the dictionary. With this design, the 3D and 2D videos with the same action category tend to share the same sparse representation. Conversely, those with different action categories tend to have different representations.

Specifically, the dictionary optimization function is designed as:

$$\langle X, D, A \rangle = \arg\min_{X,D,A} \alpha \|Y_{3D} - D_{3D} X\|_2^2 + \|Y_{2D} - D_{2D} X\|_2^2 + \beta \|Q - AX\|_2^2 \ s.t. \forall i, \|x^i\|_0 \le T, \quad (6)$$

where α and β are tradeoff parameters, $\|Y_{3D} - D_{3D} X\|_2^2$ and $\|Y_{2D} - D_{2D} X\|_2^2$ are two terms to minimize the error of the 3D and 2D dictionaries respectively, and $\|Q - AX\|_2^2$ is a label consistent regularization term to minimize the difference in sparse representation for the same class of action as introduced in [17,18]. The matrix $Q = [q_1, \dots, q_k] \in \mathbb{R}^{N \times k}$ and each vector $q_i = [q_i^1, \dots q_i^N] = [0 \dots 1, 1 \dots 0] \in \mathbb{R}^N$ is the discriminative sparse code of one shared video pair $\{y_{3D}^i, y_{2D}^i\}$. The non-zero values of q_i indicate that the pair and the dictionary item d^i share the same action label. Notice that due to our dictionary design, d_{2D}^i and d_{3D}^i always have the same action label. A is a transformation matrix that can transform the sparse code X to the most discriminative space based on label consistent matrix Q.

B. Optimization

Here, we explain how we obtain the solution for Eq. (6). Since the three terms on the right hand side of Eq. (6) belong to the same format, we can first rewrite Eq. (6) as follows:

$$\langle X, D_0 \rangle = \arg\min_{X,D_0} \|Y_0 - D_0 X\|_2^2 \ s.t. \forall i, \|x^i\|_0 \le T, \quad (7)$$

where $Y_0 = \begin{pmatrix} \sqrt{\alpha} Y_{3D} \\ Y_{2D} \\ \sqrt{\beta} Q \end{pmatrix}$, $D_0 = \begin{pmatrix} \sqrt{\alpha} D_{3D} \\ D_{2D} \\ \sqrt{\beta} A \end{pmatrix}$. Such an objective function is then sharing the same form as Eq. (5), which can be optimized using K-SVD algorithm [16].

D_{3D}, D_{2D} and A are required to be initialized before optimization. In our system, for D_{3D} and D_{2D}, we run a few iterations of K-SVD within each action class and initialize the label of the dictionary items based on their action labels. For A, we use the multivariate ridge regression model [19] with the L_2- norm:

$$A = (XX^t + \lambda I)^{-1}, \quad (8)$$

where X is calculated with the initialized D_{3D} or D_{2D}.

C. Action Classifier

Here, we explain how we apply our trained dictionary to perform view-invariant action classification.

Since D_{3D}, D_{2D} and A are jointly $L2$-normalized during the optimization process, we will need another step of normalization before they can be used for classification. We follow [17] for this process. The desired dictionaries $\hat{D}_{3D}$, $\hat{D}_{2D}$ and the desired transformation matrix $\hat{A}$ are calculated as:

$$\hat{D}_{3D} = \{ \frac{d_{3D}^1}{\|d_{3D}^1\|_2}, \frac{d_{3D}^2}{\|d_{3D}^2\|_2}, \ldots, \frac{d_{3D}^N}{\|d_{3D}^N\|_2} \}$$

$$\hat{D}_{2D} = \{ \frac{d_{2D}^1}{\|d_{2D}^1\|_2}, \frac{d_{2D}^2}{\|d_{2D}^2\|_2}, \ldots, \frac{d_{2D}^N}{\|d_{2D}^N\|_2} \}$$

$$\hat{A} = \{ \frac{a^1}{\|a^1\|_2}, \frac{a^2}{\|a^2\|_2}, \ldots, \frac{a^N}{\|a^N\|_2} \} \quad (9)$$

During the training phase, we first extract the features from real 2D videos as explained in Section III-B. We then calculate the sparse representation of each training samples using the trained 2D dictionary. This allows us to project the features in a real 2D video into a view-invariant sparse representation:

$$x = \arg\min_{x'} \| y - \hat{D}_{2D} x' \|_2^2 \quad s.t. \forall i, \|x^i\|_0 \leq T. \quad (10)$$

Finally, we train a non-linear multi-class SVM with χ^2 kernel for action classification:

$$K(x,y) = \exp(-\sum_{i=1}^{M} D(x,y)). \quad (11)$$

V. Experimental Results

In this section, we first provide experiment setup details. We then evaluate the performance of our method with two public multi-view datasets including the IXMAS database and the N-UCLA database.

The synthetic 3D and 2D datasets we used for transfer dictionary learning are open to public. They can be found at our project website.

All experiments were performed with a desktop computer with an Intel i7-4790k CPU, an NVIDIA Quadro K2200 graphics card and 16GB RAM.

A. Transfer Dictionary Learning

We used the software package Poser 2014 [14] to animate and retarget motion capture data files, as well as render the scenes into 2D videos. The 3D videos were synthesized with 5 cylindrical characters with 18 motion data files per action class. The 2D videos were synthesized with 5 high quality characters with a uniformly sampled 18 viewpoints per action class, with the motion files randomly selected. The azimuthal angle of the projection was uniformly sampled as {0°, 60°, 120°, 180°, 240°, 300°} and the polar angle of the projection was sampled as {0°, -30°, -60°}. This setup allowed us to generate the same number of 3D and 2D videos for transfer dictionary learning as required by K-SVD.

As a result, for the experiments with the IXMAS dataset with 11 action classes, we synthesized 990 pairwise 3D and 2D videos. For the experiments with the N-UCLA dataset with 10 action classes, we synthesized 900 pairwise videos. From our experience, a larger synthetic dataset would perform better. We decided the size such that we can obtain good system accuracy with a reasonable training time.

We extracted dense trajectories from 2D synthetic videos, as well as 2D real videos from the IXMAS and the N-UCLA datasets. Similar with [9], we constructed a codebook for each of the four descriptors in the dense trajectories separately. For each descriptor, we applied k-means to cluster a subset of 100,000 dense trajectory features into 500 visual words. This resulted in a 2D feature Y_{2D} of 2,000 dimensions.

For 3D synthetic models, similar with [10], we set the trajectory sample step to 5 frames, and the trajectory length to 15 samples. We constructed codebooks for 3D trajectories and HO3V descriptors respectively. We applied k-means to cluster a subset of 100,000 3D trajectory-aligned descriptors into 500 visual words. This resulted in a 3D feature Y_{3D} of 1,000 dimensions.

When constructing the transfer dictionary learning, to initialize the dictionary pair D_{3D} and D_{2D}, we employed k-means 5 times on the features Y_{3D} and Y_{2D} respectively. We set the dictionary sizes N to 1000 for both D_{3D} and D_{2D}. The 3D dictionary trade-off parameter α was set to 2.0. The label consistent trade-off parameter β was set to be 4.0. Finally, the number of iterations for the K-SVD algorithm was set to 60.

B. Experiments on the IXMAS Dataset

The IXMAS dataset [20] contains 11 action classes captured in 5 different viewpoints with 10 different actors. Figure 8 shows some examples.

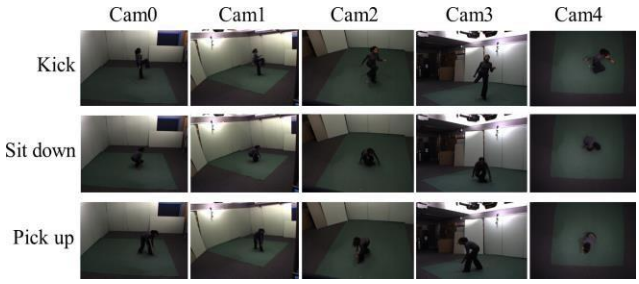
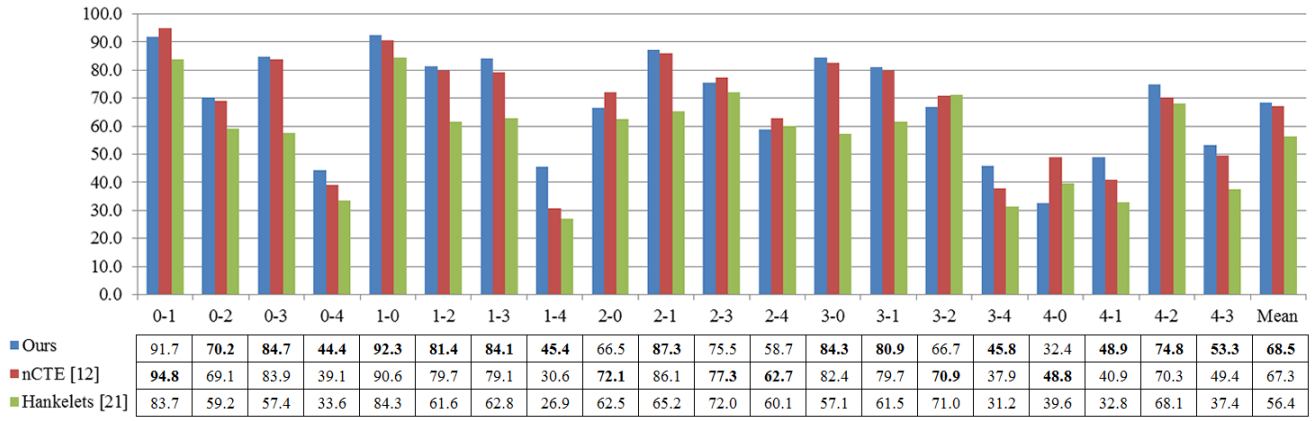


Figure 8. Sample frames form the IXMAS dataset

A major strength of our system is that it works on arbitrary views, which means that we do not require view labels (or camera labels) in any of our system training. In this experiment, we combined the 2D real videos from different viewpoints in IXMAS without retaining the view labels for training and testing the action classifier. We followed the leave-one-actor-out cross validation strategy from [10, 11] to obtain the recognition accuracy. As shown in Table 1, our system outperforms existing methods including 3D Exemplar and 4D-AFMS, mainly because we used synthetic 3D and 2D videos for transfer learning to enhance the accuracy. Table 2 shows the performance of our system under different cross validation strategies. The accuracy shows a downward trend with the reduction of the number of the training samples.



	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
Ours	91.7	**70.2**	**84.7**	**44.4**	**92.3**	**81.4**	**84.1**	**45.4**	66.5	**87.3**	75.5	58.7	**84.3**	**80.9**	66.7	**45.8**	32.4	**48.9**	**74.8**	**53.3**	**68.5**
nCTE [12]	**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	67.3
Hankelets [21]	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

Figure 9: Cross-view recognition accuracy of all possible viewpoint combinations. The horizontal axis labels are formatted as "source view - target view" in cross-view training.

Table 1. Accuracy of view-invariant action recognition with the IXMAS dataset

	Accuracy (%)
Ours	**84.3**
3D Exemplar [10]	81.3
4D-AFMS [11]	78.0

Table 2. Accuracy obtained with different cross validation strategies

	Training samples/Total samples (%)	Accuracy (%)
Leave-one-actor-out	90	**84.3**
Leave-two-actors-out	80	82.6
Leave-three-actors-out	70	77.6
Leave-four-actors-out	60	69.7
Leave-five-actors-out	50	58.4

In order to compare with existing works on cross-view action recognition that utilize view labels including nCTE [12] and Hankelets [21], we conducted another experiment considering view labels. Here, we grouped the videos in the IXMAS dataset into different views, and evaluated the accuracy of transferring one view to another. We followed the leave-one-action-out cross validation strategy from [12, 21]. Figure 9 shows that our algorithm outperforms the state-of-the-art method nCTE in most cross-view pairs, as well as the average system accuracy. Figure 10 shows that our algorithm outperforms nCTE in most action classes. These demonstrate that our system can realize cross-view action recognition by transferring the view-invariance from 3D models. Notice that in our ideal setup, the system does not require view information. This experiment is designed for the sake of comparison only.

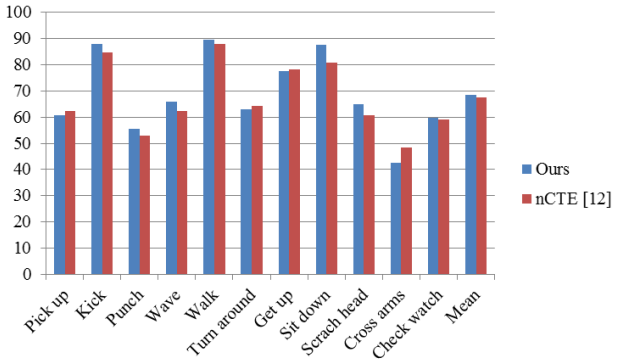


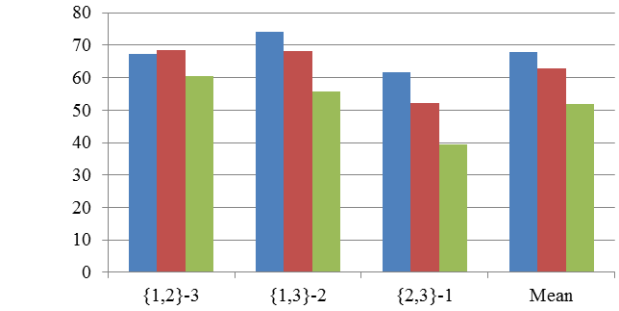
Figure 10. Cross-view recognition accuracy per action class in IXMAS

C. *Experiments on the N-UCLA Dataset*

The N-UCLA dataset [22] contains 10 action classes captured in 3 different viewpoints with 10 different actors.

Figure 11. Example frames from the N-UCLA dataset

We evaluated our system accuracy in cross-view action recognition and compare with existing work including nCTE [12] and CVP [23]. We followed the experimental setup in [12, 23] that utilizes videos captured from two cameras for training and the other one for testing. The accuracy was calculated using leave-one-action-out cross validation. As shown in Figure 12, our method outperforms existing algorithms in most of the cross-view setups and the overall result.



	{1,2}-3	{1,3}-2	{2,3}-1	Mean
Ours	67.3	**74.2**	**61.8**	**67.8**
nCTE [12]	**68.6**	68.3	52.1	63.0
CVP [23]	60.6	55.8	39.5	52.0

Figure 12. Cross-view accuracy using N-UCLA, with the horizontal axis formatted as "two source views - target view"

VI. Conclusion and Discussions

In this paper, we have proposed a view-invariant human action recognition framework. Unlike previous work, the view-invariance is obtained from synthetic 2D and 3D training data through transfer dictionary learning. The trained dictionary is used to project real world a 2D video into a view-invariant sparse representation, facilitating an arbitrary view action classifier. Due to the abundant availability of training data and the use of 3D information, our system performs robustly and its accuracy suppresses that of previous work.

One advantage of our framework is that it separates the training of the transfer dictionary and classifier into two separate processes using different sources of training data. The view-invariant transfer dictionary is trained solely with synthesized data. As a result, it is possible to produce an even larger synthetic 2D dataset to cover all possible viewpoints. On the other hand, the action classifier is trained with real world 2D video data, and we do not require view information to be known, which relaxes the requirement of obtaining real world data.

Dictionary learning can be considered as a linear projection algorithm, and can be limited in representing the view-invariance of 3D video. In the future, we are interested in apply non-linear algorithms such as Neural Networks with synthetic training data to achieve better results.

Another piece of future work is to investigate the impact of 2D projection selection when creating the synthetic 2D video. Instead of uniformly sample the projection angle, it may be possible to learn an optimal set of projections per action class.

Acknowledgement

This project was supported by the Engineering and Physical Sciences Research Council (EPSRC) (Ref: EP/M002632/1).

References

[1] G. Chen, M. Giuliani, D. Clarke, and A. Gaschler, A. Knoll, "Action recognition using ensemble weighted multi-instance learning," in Proc. IEEE ICRA, 2014.

[2] N. Hu, G. Englebienne, Z. LouZ, and B. Krose, "Learning latent structure for activity recognition," in Proc. IEEE ICRA, 2014.

[3] G. Cheung, S. Baker, and T. Kanade, "Shape-from-silhouette of articulated objects and its use for human body kinematics estimation and motion capture," in Proc. IEEE CVPR, 2003.

[4] Z. Lin, Z. Jiang, and L. S. Davis, "Recognizing actions by shape-motion prototype trees," in Proc. IEEE ICCV, 2009.

[5] J. Liu, Saad Ali, and Mubarak Shah. "Recognizing human actions using multiple features," in Proc. IEEE ICCV, 2008.

[6] F. Lv and Ramakant Nevatia. "Single view human action recognition using key pose matching and viterbi path searching," in Proc. IEEE ICCV, 2008.

[7] M. Blank, L. Gorelick, E. Shechtman, M. Irani, and R. Basri, "Actions as space-time shapes," in Proc. IEEE ICCV, 2005.

[8] A. Kovashka and K. Grauman. "Learning a hierarchy of discriminative space-time neighborhood features for human action recognition," in Proc. IEEE CVPR, 2010.

[9] H. Wang, A. Kläser, C. Schmid and L. L. Chen, "Action recognition by dense trajectories," in Proc. IEEE CVPR, 2011.

[10] D. Weinland, E. Boyer and R. Ronfard, "Action recognition from arbitrary views using 3d exemplars," in Proc. IEEE ICCV, 2007.

[11] P. Yan, S. M. Khan and M. Shah, "Learning 4d action feature models for arbitrary view action recognition," in Proc. IEEE CVPR 2008.

[12] A. Gupta, J. L. Martinez, J. J. Little and R. J. Woodham, "3D pose from motion for cross-view action recognition via non-linear circulant temporal encoding," in Proc. IEEE CVPR, 2014.

[13] Carnegie Mellon University Motion Capture Database. URL http://mocap.cs.cmu.edu/

[14] Poser2014. URL http://my.smithmicro.com/poser-pro-2014.html

[15] J. Mairal, F. Bach and J. Ponce, "Online dictionary learning for sparse coding," in Proc. 26th Annual International Conference on Machine Learning. ACM, 2009, pp. 689-696.

[16] M. Aharon, M. Elad, and A. Bruckstein, "K-svd: An algorithm for designing overcomplete dictionries for sparse representation," IEEE Trans. on Signal Processing, 54(1):4311–4322, 2006.

[17] F. Zhu and L. Shao, "Weakly-supervised cross-domain dictionary learning for visual recognition," International Journal of Computer Vision, vol. 109, no. 1-2, pp. 42-59, 2014.

[18] Z. Jiang, Z. Lin, L. Davis, "Learning a discriminative dictionary for sparse coding via label consistent k-svd," in Proc. IEEE CVPR, 2011.

[19] G. Golub, P. Hansen, and D. O'leary. "Tikhonov regularization and total least squares," SIM J. Matrix Anal. Appl., 21(1):185–194, 1999

[20] D. Weinland, R. Ronfard, and E. Boyer, "Free viewpoint action recognition using motion history volumes," In *CVIU*, 2006.

[21] B. Li, O. Camps, and M. Sznaier. "Cross-view activity recognition using hankelets," in Proc. IEEE CVPR, 2012.

[22] J. Wang, X. Nie, Y. Xia, Y. Wu, and S. Zhu. "Cross-view action modeling, learning and recognition," in Proc. IEEE CVPR, 2014.

[23] Z. Zhang, C. Wang, B. Xiao, W. Zhou, S. Liu, and C. Shi. "Cross-view action recognition via a continuous virtual path," in Proc. IEEE CVPR, 2013.