Unlabelled 3D Motion Examples Improve Cross-View Action Recognition

Ankur Gupta
ankgupta@cs.ubc.ca
Alireza Shafaei
shafaei@cs.ubc.ca
James J. Little
little@cs.ubc.ca
Robert J. Woodham
woodham@cs.ubc.ca

Department of Computer Science
University of British Columbia
Vancouver, Canada

Abstract

We demonstrate a novel strategy for unsupervised cross-view action recognition using multi-view feature synthesis. We do not rely on cross-view video annotations to transfer knowledge across views, but use local features generated using motion capture data to learn the feature transformation. Motion capture data allows us to build a feature level correspondence between two synthesized views. We learn a feature mapping scheme for each view change by making a naive assumption that all features transform independently. This assumption along with the exact feature correspondences dramatically simplifies learning. With this learned mapping we are able to "hallucinate" action descriptors corresponding to different viewpoints. This simple approach effectively models the transformation of BoW based action descriptors under viewpoint change and outperforms the state of the art on the INRIA IXMAS dataset.

1 Introduction

A view-invariant representation of human motion is crucial for effective action recognition. Widely popular shape [4, 8, 27] and optical flow-based [6, 23] features, which are commonly used to describe actions in videos, are not specifically designed for viewpoint invariance. The aggregated video descriptors based on these local features are robust to variations in size (with multi-scale features) and temporal shifts. However, the representation in general is not view-invariant. This implies that the method's effectiveness highly depends on availability of training data from different views. One way to deal with this challenge is to gather a large enough training set that includes a wide variety of viewpoints. However, this may not be a feasible approach in many real-life scenarios. When the training view (or source view) is significantly different from the test view (or target view), we need to come up with strategies to either transfer knowledge across views or devise a view-invariant description to recognize actions. These techniques are often referred to as cross-view action recognition techniques.

Many cross-view action recognition approaches transform action descriptors to a view-invariant space where observations from the source and the target view are comparable [10,

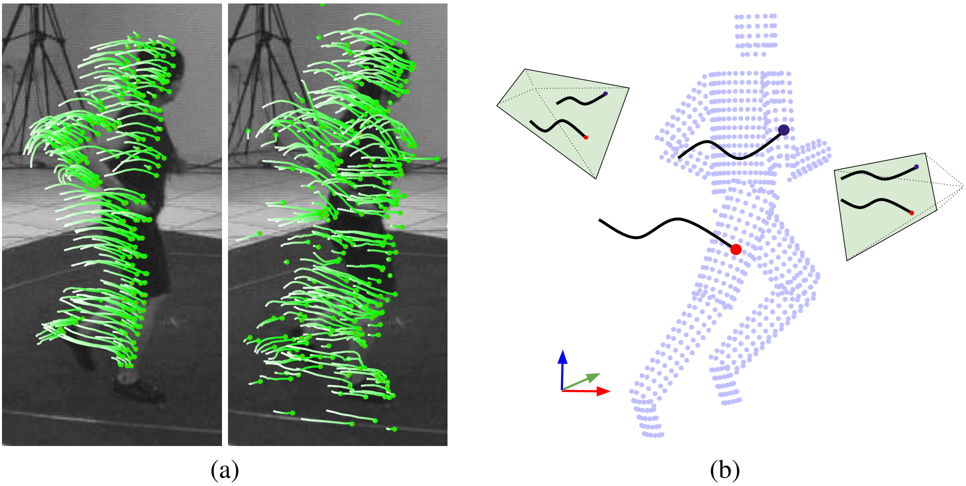


Figure 1: (a) A visual comparison between synthesized mocap trajectories [9] (left) and optical flow-based dense trajectories [23] (right). We exploit the similarity in their shape to learn a feature mapping between different views using mocap trajectories, and use this mapping to transform dense trajectory based features commonly used for action recognition. (b) To generate view correspondence at the local feature level, the human body is represented using cylindrical primitives driven by mocap data. We project the 3D path of the points on the surface to multiple views and learn how the features based on these idealized trajectories transform under viewpoint changes.

12, 13, 15, 29]. However, learning the invariant space requires supervision. Here we present a cross-view action recognition scheme to deal with the unsupervised case, i.e., we have no access to the target view examples. This is a likely scenario when it is not possible to predict the test view in advance. Instead of looking for a view invariant space, we learn a probabilistic function to transform descriptors from one view to another using a large collection of unlabelled motion capture (mocap) data. This allows us to generate multiple new descriptors based on each source-view example to account for many possible target-views. We augment the training data using these new descriptors to make our model resilient to viewpoint changes.

A wide range of techniques have been proposed that utilize partial annotations in the target view to achieve view-invariance (such as [6, 10, 15, 25, 26, 29]). However, very few methods deal with the unsupervised case [9, 12, 13, 28]. A recent approach by Gupta *et al*. [9] transfers knowledge across views without using any target view data. The method temporally aligns each training video to a mocap sequence in a large mocap database (see Figure 3 in [9]). This matching step returns a mocap sequence similar to the video along with the matched viewpoint. The mocap sequence returned is then used to generate additional synthetic training examples. Although this is an unsupervised approach, there is an inherent assumption that the mocap database is large enough to cover a wide range of actions. Also, the quality of generated synthetic examples depends on the accuracy of the matching step.

To deal with these limitations, we propose to learn directly the relationship among motion features observed from different points of view. We achieve this without having access to any multi-view video data (synchronized or otherwise) or a matching step as in [9]. To learn view correspondence between trajectory features, we begin by synthetically generating trajectories using mocap examples from multiple viewpoints (as shown in Figure 1 (b)). We assume that, due to the similarity between synthesized trajectories and dense trajectories, we can learn the view transformations on one and apply it to the other. Figure 1 (a) shows the visual similarity between the dense trajectories generated from a video and corresponding mocap trajectories. We use a video example from the HumanEva dataset [20] which has videos with synchronized mocap sequences.

Trajectory features can be efficiently generated for both video [23] and mocap [9]. Each trajectory descriptor is a concatenation of 2D displacement vectors over a fixed time window normalized by sum of their magnitudes. We use mocap examples to collect a large number of descriptor pairs for each viewpoint change. To simplify the problem, we quantize each descriptor to its closest codeword using a fixed vocabulary. Furthermore, we learn a probabilistic mapping between the codewords in two views. For action recognition, we describe each video with a bag of words (BoW) of dense-trajectory features. Given multiple feature mappings we can "hallucinate" action descriptors from different viewpoints and use them as additional training examples.

This strategy allows us to deal with real-world scenarios in which the viewpoint of the videos cannot be predicted in advance. Furthermore, we relax the assumption from [9] that the mocap database contains the actions we wish to recognize. Thus, we describe a view-independent model of human actions which is flexible, action-independent, and requires no additional labelling effort. We also briefly explore how this representation can be utilized for predicting the elevation angle of a camera given action annotations. To summarize, there are two main contributions of this paper: first, we propose a novel approach to building correspondences between multi-view features using mocap trajectories; second, we describe a simple strategy to learn a feature transformation function using this data and utilize it for cross-view action recognition.

The rest of the paper is organized as follows: in Section 2, we discuss the background and related work. We present our main contributions in Section 3. Section 4 describes the experiments, followed by discussion and future work in Section 5.

2 Background and Related Work

The literature of cross-view action recognition can be divided into three major categories: the first utilizes geometric [11, 16, 19] or dynamical properties [12] of human motion to develop a view-invariant representation. The second, inspired by transfer learning, treats this as a purely statistical learning problem [6, 10, 15, 29], and often does not reason about the geometry. The third relies on using 3D exemplars of motions that can be matched with the video evidence [9, 18, 25, 26] to bring view invariance. Other interesting approaches include bilinear modelling of action with viewpoint [3].

Recently statistical techniques based on domain adaptation [6, 13, 14, 28] have become popular because they do not require detection or tracking of individual body parts. However, some of these methods rely on having access to synchronized multi-view video data or labelled examples in the target-view. Other recent approaches [13, 28], taking inspiration from unsupervised domain adaptation [7], extend these ideas to the case where no labels are available in the target view by utilizing the structure in the data itself. However, they still need unlabelled target view videos which may not be readily available in the general case. Now, we discuss approaches that are more closely related to our method.

2.1 3D exemplar-based approaches

3D representation of motion is an obvious way to achieve view independence. Ramanan and Forsyth [18] track 2D body parts and match them to mocap data annotated with action labels. They also recover 3D pose and camera viewpoint in the process. However, this approach depends on reliable 2D pose estimation. [25, 26] build a voxel-based 3D representation of human motion using multi-view video data. These methods do not require explicit 2D part detectors, but they still need synchronized and calibrated multi-view videos to construct the model. [9] removes the dependency on multi-view video data and pose detection by directly matching motion features from video to idealized trajectories generated using mocap data. Although this approach is effective in transferring knowledge across views in the unsupervised case, it is limited by the availability of motion examples in the mocap database. Hence, it is harder to scale it to a large number of classes. The method is also limited by the accuracy of the intermediate matching step. By learning a direct mapping at the feature level, we are able to address both of these limitations.

2.2 Mapping dictionaries for view invariance

Farhadi *et al.* [6] use frame-level correspondence between descriptors seen from two synchronized views to learn the transformation between feature clusters. Liu *et al.* [14] build separate dictionaries in source and target views and look for correspondence between words using bipartite matching to form bilingual words. These bilingual words act as a mapping from individual dictionaries to a common one. Compared to [6], their approach is more flexible as they use video-level correspondences. Zheng *et al.* [29] improve upon this idea to simultaneously learn a common dictionary between views and individual view-specific dictionaries. They describe each video as a combination of these two factors. The coefficients of these dictionaries are used to transfer knowledge across views.

Again, all the above methods rely on the availability of video examples seen from multiple views during the training. We relax this assumption by using unlabelled mocap data instead. For learning correspondences between features across views, we take a much simpler approach: we learn one dictionary including data from all the views. As we have feature-level correspondences, we can learn a direct mapping between codewords in two views, so we do not require the matching step [14]. However, simultaneous learning of dictionaries while looking for transformations (like [29]), and including a bidirectional data synthesis term (like [10]) can complement our approach. We leave such improvements to future work.

Among related computer vision problems, learning relationships between features across viewpoints is an interesting challenge for simultaneous object detection and 3D pose estimation in images. Some of these approaches use 3D CAD models of objects – analogous to mocap trajectories in our case – to learn the local visual features as well as spatial relationships between them [17, 21]. Another approach [22], similar to ours, solves this problem by establishing connections between features observed from neighboring views (called activation links). However, these methods are not directly applicable to the problem at hand.

3 Methodology

Our cross-view action recognition pipeline can be summarized as three main steps: a) generate a large number of synthetic trajectory features observed from different viewpoints using mocap data; b) use these examples to learn a feature mapping for each view change; and c) apply this model to generate new multi-view descriptors to augment the training data. We describe each of these steps in detail in the following subsections.

3.1 Generating feature correspondences

We use the mocap trajectory generation pipeline by Gupta *et al.* [9]. They use a human model with cylindrical primitives (see Figure 1(b)). This model can be easily driven by mocap data. Each limb consists of a collection of points that are placed on a 3D cylindrical surface. Given a camera viewpoint, these points are projected under orthographic projection and tracked for $L(=15)$ consecutive frames. The resulting displacement vectors are used to generate trajectory features comparable with the dense trajectory features of Wang *et al.* [23]. Since the method assumes orthographic projection, each camera viewpoint can be uniquely defined by only two parameters, the elevation angle θ and the azimuthal angle ϕ.

In this work, we also assign a unique ID to each point on the 3D surface. Now given two arbitrary viewpoints, we can find a correspondence between features that originate from the same point on the surface (see Figure 1(b)). One of the challenges in generating corresponding feature pairs is occlusion. We may not get enough pairs when the camera view changes significantly. To deal with this problem, we further define an equivalence region for each point based on the geodesic distance on the model surface. If a feature correspondence is not available due to occlusion, we relax the search to a region of radius R_m around the point on each limb. Finally, we add these feature pairs to our training set. We choose $R_m = \mathit{model\ height}/10$ for all our experiments.

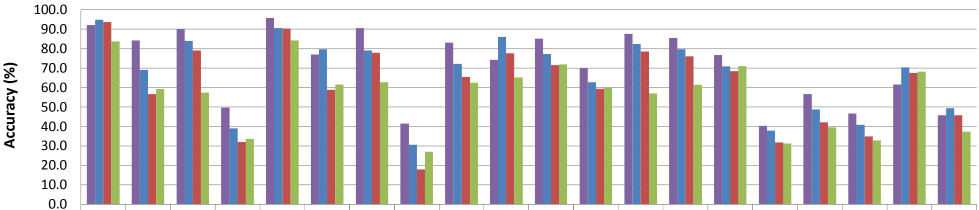
3.2 Learning codeword transformations

We quantize the mocap trajectory features using a fixed codebook $\mathcal{C}$ of size n. Given a source elevation angle θ and relative change in viewpoint given by $\Delta = (\delta\theta, \delta\phi)$, we define the training set $\mathcal{D}_\theta^\Delta = \{(f_i, g_i)\}_1^m$ to be the set of m pairs $(f, g) \in \mathcal{C} \times \mathcal{C}$, where f_i and g_i are the codewords for two corresponding trajectory features. Note that we do not need a specific source azimuthal angle ϕ for data collection because, while elevation angles in this case can be uniquely defined with respect to the ground plane (angle made with the normal to the ground plane), azimuthal angle has no fixed reference. Hence, to generate trajectory correspondences we consider different source elevations while using all the angles in $\pi/6$ intervals as source azimuthal angles.

Given the training data $\mathcal{D}_\theta^\Delta$, the relationship between codewords f_i and g_i can be modeled within a probabilistic framework. Since we assume each feature transforms independently, we can learn a joint probability mass function $\mathrm{P}(F, G)$ which captures the probability of having feature pairs (f_i, g_i) in $\mathcal{D}_\theta^\Delta$. We train the model using maximum likelihood estimation. We calculate the empirical probability by counting the co-occurrences of (f_i, g_i) in $\mathcal{D}_\theta^\Delta$ followed by normalization. The conditional probability distribution of G, given an observation of codeword f_i in the source domain can be written as

$$\mathrm{P}(G|F = f_i) = \frac{\mathrm{P}(F = f_i, G)}{\mathrm{P}(F = f_i)} = \frac{\mathrm{P}(F = f_i, G)}{\sum_{c \in \mathcal{C}} \mathrm{P}(F = f_i, G = c)} \tag{1}$$

After observing an instance of codeword f_i in the source view, $\mathrm{P}(G|F = f_i)$ allows us to infer the possible outcomes in the target view. In the next section, we use this probability distribution to map source BoW features to a target view.



	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
■ Ours	92.1	**84.2**	**90.0**	**49.7**	**95.8**	77.0	**90.6**	**41.5**	**83.0**	74.2	**85.2**	**70.0**	**87.6**	**85.5**	**76.7**	**40.3**	**56.7**	**46.7**	61.5	45.8
■ nCTE	**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**
■ w/o aug.	93.6	56.7	79.1	32.1	90.3	58.8	77.9	17.9	65.5	77.6	71.5	59.4	78.5	76.1	68.5	31.8	42.1	34.8	67.6	45.8
■ Hankelets	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

Train camera - Test camera

Figure 2: Accuracy comparison per train-test view pair. We compare with nCTE [9], our method without augmentation, and Hankelets [12].

3.3 Synthesizing cross-view action descriptors

Given an l_2 normalized BoW descriptor of an action, we wish to synthesize a new descriptor as seen from the viewpoint $\Delta = (\delta\theta, \delta\phi)$ away from the original view. Let $\mathbf{x} = [x_1, \dots, x_n]^{\mathrm{T}}$ be the BoW descriptor in the source view, and $\mathbf{y} = [y_1, \dots, y_n]^{\mathrm{T}}$ be the descriptor we want to estimate. As seen in the last section, we have a probabilistic mapping between codewords across views. Using this mapping, we return an average descriptor by taking the expectation

$$\bar{\mathbf{y}} = [\mathbb{E}[y_1], \dots, \mathbb{E}[y_n]]^{\mathrm{T}} \tag{2}$$

$$\mathbb{E}[y_j] = \sum_{i=1}^{n} x_i \cdot \mathrm{P}(G = f_j | F = f_i) \tag{3}$$

By organizing $\mathrm{P}(G|F)$ in the form of a matrix (say N) where the i-th row is the categorical distribution $\mathrm{P}(G|F = f_i)$, we can rewrite the above formulation as a matrix multiplication

$$\bar{\mathbf{y}} = \mathrm{N}^{\mathrm{T}}\mathbf{x} \tag{4}$$

We further l_2 normalize $\bar{\mathbf{y}}$ to make it consistent with the original descriptor. We refer to N as the transition matrix. We get one such matrix per transition, i.e., per training set $\mathcal{D}_{\theta}^{\Delta}$.

4 Experiments

We use the INRIA Xmas Motion Acquisition Sequences (IXMAS) [24] to test our approach. This dataset contains 11 action classes, such as *check watch*, *sit down*, *get up*, and *turn around*, performed by 10 different actors. The dataset consist of synchronized videos observed from 5 cameras with diverse viewpoints. Each sequence also has reconstructed volumes and silhouettes of the human subjects, but we do not use this information.

For 3D motion examples, we use the CMU Motion Capture Database [1] which contains over 2600 mocap sequences of human subjects performing a variety of actions. We use the 2200 shortest sequences for training the model. Though the CMU dataset contains some action labels for each file, we do not use these annotations.

Method	Average accuracy
Ours	**71.7%**
nCTE based matching [9]	67.4%
w/o aug.	62.1%
Hankelets [12]	56.4%

Table 1: Comparison of overall recognition accuracy averaged over all view pairs.

4.1 Unsupervised cross-view action recognition

For the task of cross-view action recognition, we follow the same evaluation scheme as in [9, 28]. At each round, we use all the examples from one camera for training and then evaluate the performance on the other four cameras. We compute dense trajectory descriptors of length 15 frames on all the IXMAS videos using the publicly available code from Wang *et al*. [23][1]. We generate a 2000 word codebook $\mathcal{C}$ using a random subset of these trajectories and k-means clustering. Note that Wang *et al*. use HOG, HOF, and MBH descriptors along with trajectories. However, we only use trajectory descriptors.

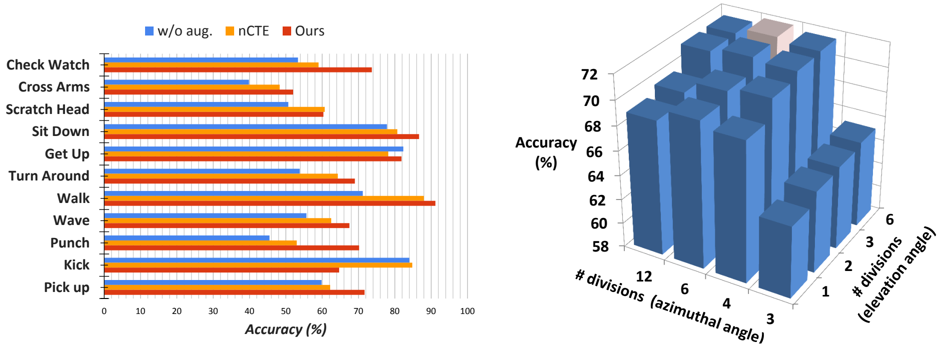
To learn the mapping between codewords, we take a random subsample of the mocap data, keeping 10% of the frames. Since each mocap file returns a large number of feature correspondences, subsampling helps us to keep the amount of training data manageable. We generate mocap trajectories from multiple viewpoints and quantize them using the same codebook $\mathcal{C}$. For this experiment, we quantized the elevation angle θ to $\{30, 60, 90\}$ degrees and the azimuthal angle ϕ to 6 equally spaced angles in $[0\ 360)$. Thus, for each source elevation θ, we have 17 possible viewpoint transitions (excluding transition to itself). Given a training example from IXMAS, we generate one synthesized descriptor per transition as described in Section 3.3. Since we do not know the source elevation θ for the training set, we consider transitions from all the possible elevation angles. This gives us 51 synthesized descriptors per training example.

Following [2, 9], we augment our original training data using these new descriptors. In such augmentation schemes, the synthesized data is often weighted less compared to the original data [2]. We empirically set the weight of the augmented data to 0.01. We train an SVM with χ^2 kernel using one-vs-all strategy. The final source code as well as the preprocessed data is publicly available [2].

To evaluate our method quantitatively, we compare against the work of Li *et al*. [12] and Gupta *et al*. [9]. The results for comparison are taken directly from the cited papers. We also measure the performance without feature augmentation, referred to as *w/o aug.*. Figure 2 compares the accuracy per train-test view pair. Our method outperforms the state of art on most view combinations. The accuracy averaged over all views is 71.7%, a 4.3% improvement over [9] (see Table 1). Notably, we do not need the expensive matching step for synthesizing multi-view descriptors. Instead, we can generate new examples by simple matrix multiplications. As a result, our training is significantly more efficient than [9]. Figure 3(a) compares our performance with the *w/o aug.* and [9] for recognizing particular action classes.

[1] http://lear.inrialpes.fr/people/wang/dense_trajectories

[2] http://cs.ubc.ca/research/motion-view-translation/



(a) (b)

Figure 3: (a) A comparison of per class recognition accuracy of our method with *w/o aug.* and nCTE [9]. (b) Effect of number of augmented views (division along elevation and azimuthal angles) on the overall action recognition accuracy. The best result is highlighted.

Effect of number of augmented views In the above experiments, we used the same quantization for elevation and azimuthal angles for augmentation as in [9] for a fair comparison. Now, we study the effect of varying these parameters on the overall accuracy. However, changing quantization changes the number of synthetic examples. To balance the cumulative weight of synthetic data: a) we fix the weight 0.01 for 18 viewing angles as used above; b) for the rest of the trials, we change the weight inversely proportional to the number of examples so that that the cumulative weight of synthetic examples remains constant, e.g., we set the per example weight to 0.005 for 36 viewing angles. The rest of the pipeline remains the same. We report the overall accuracy numbers as a function of the number of divisions considered along azimuthal and elevation angles in Figure 3 (b). We observe a sharp increase in accuracy as we go from 3 to 4 divisions in azimuthal angle. However, there are no significant changes once we increase the azimuthal divisions to 6 and elevation divisions to 3. This shows that approximately 18 views are sufficient to capture the variation in the viewpoint.

4.2 Predicting elevation angles with synchronized views

In the previous experiments, we saw how augmenting the training data with synthetic examples corresponding to multiple viewpoint transitions improves cross-view action recognition accuracy. We are also interested in the question: how specific are these feature mappings to a particular viewpoint change? This motivates us to consider the problem of estimating the target elevation angle given the source elevation angle and action annotations in both the source and the target view. We achieve this by comparing the action recognition performance gained by different possible transitions. In contrast to the experiments above: a) we augment the training data with the synthetic data generated using one transition at a time and b) weigh synthetic examples the same as the original descriptors. We measure the recognition accuracy on the target view examples and find the transition with the highest gain. Using the relative elevation angle change corresponding to the best transition, we estimate the target view elevation angle. For this experiment we consider a denser grid of viewpoints. We choose a 15 degrees (in the range [15 90] degrees) division in elevation angle and 30 degrees (in the range [0 360) degrees) in azimuthal angle giving rise to a total of 142 transitions.

To obtain the ground truth elevation angle for each camera view, we use the calibration

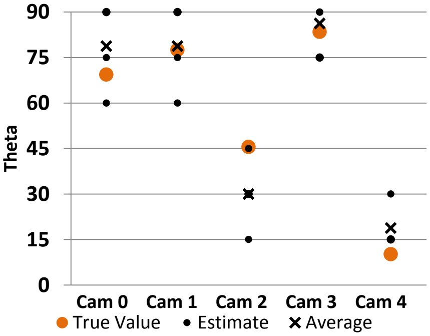


Figure 4: Elevation angle prediction given annotated source and target view data. We show all the predictions along with their mean and the ground truth.

data provided with the IXMAS dataset. We decompose the camera matrix to obtain the camera center in world coordinates. Ideally we would like the elevation angle to be the angle made by line joining the camera center and center of the body of the actor. However, this angle can change as actors move. To obtain a reasonable approximation of elevation angle for each camera view, we calculate the angle between the vertical and the line joining the camera center to a point which is $d = 0.8$m meters above the origin lying near the center of the ground plane. We assume the actual elevation angle does not change significantly as actors move around.

In the IXMAS dataset, for each camera, we can obtain four estimates for elevation angle by treating other cameras as source views. We show our results in Figure 4. We observe that we are able to get a reasonable estimate based on this simple approach of choosing the best transition. We note that the estimation also includes the quantization error for both source and target views. These results suggest that our approach not only boosts action recognition accuracy when synthetic examples are aggregated, but individual transition matrices are also specific to the relative change in the elevation angle.

5 Discussion and Future Work

In this paper, we described a simple yet effective approach to add view-invariance to an action recognition pipeline without any additional supervision. We also show that given the labelled examples in two views we can predict the relative viewpoint change along elevation angle. Instead of relying on pose estimation, we learn a probabilistic model for feature transformations due to viewpoint changes. For training, we rely on 3D human motion examples in the form of unlabelled mocap data. However, we do not require mocap data with subjects performing specific actions, which makes our approach flexible and widely applicable. In summary, using synthetic motion examples as a proxy for real trajectories allows us to establish a correspondence between features across views, which is a difficult problem given video data alone.

Given this correspondence data, we use a straightforward method to learn a model to transform an action descriptor. More advanced approaches may help improve the results. For instance, rather than using a fixed vocabulary, we can jointly learn the vocabulary and transformations to build a more flexible model. We also make the assumption that all features transform independently. However, we expect the features in the same frame to be correlated with each other. Furthermore, accounting for correspondences which are lost due to occlusion and extending synthetic feature generation beyond trajectories are interesting subjects for further exploration.

Acknowledgements. Authors would like to thank Julieta Martinez for the efficient implementation of mocap trajectory generation pipeline. This work was supported in part by the Natural Sciences and Engineering Research Council of Canada (NSERC) and the Institute for Computing, Information and Cognitive Systems (ICICS) at UBC, and enabled in part by WestGrid and Compute Canada.

References

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

[2] Chao-Yeh Chen and Kristen Grauman. Watching Unlabeled Video Helps Learn New Human Actions from Very Few Labeled Snapshots. In *CVPR*, 2013.

[3] Fabio Cuzzolin. Using bilinear models for view-invariant action and identity recognition. In *CVPR*, 2006.

[4] Navneet Dalal and Bill Triggs. Histograms of Oriented Gradients for Human Detection. In *CVPR*, 2005.

[5] Navneet Dalal, Bill Triggs, and Cordelia Schmid. Human Detection Using Oriented Histograms of Flow and Appearance. In *ECCV*. 2006.

[6] Ali Farhadi and Mostafa Kamali Tabrizi. Learning to Recognize Activities from the Wrong Viewpoint. In *ECCV*, 2008.

[7] Boqing Gong, Yuan Shi, Fei Sha, and K. Grauman. Geodesic Flow Kernel for Unsupervised Domain Adaptation. In *CVPR*, 2012.

[8] Lena Gorelick, Moshe Blank, Eli Shechtman, Michal Irani, and Ronen Basri. Actions as Space-Time Shapes. *TPAMI*, 29(12), 2007.

[9] Ankur Gupta, Julieta Martinez, James J. Little, and Robert J. Woodham. 3D Pose from Motion for Cross-view Action Recognition via Non-linear Circulant Temporal Encoding. In *CVPR*, 2014.

[10] De-An Huang and Yu-Chiang Frank Wang. Coupled Dictionary and Feature Space Learning with Applications to Cross-Domain Image Synthesis and Recognition. In *ICCV*, 2013.

[11] Imran N. Junejo, Emilie Dexter, Ivan Laptev, and Patrick Pérez. View-independent Action Recognition from Temporal Self-similarities. *TPAMI*, 33(1), 2011.

[12] Binlong Li, Octavia I. Camps, and Mario Sznaier. Cross-view Activity Recognition using Hankelets. In *CVPR*, 2012.

[13] Ruonan Li and Todd Zickler. Discriminative Virtual Views for Cross-View Action Recognition. In *CVPR*, 2012.

[14] Jingen Liu and Mubarak Shah. Cross-view Action Recognition via View Knowledge Transfer. In *CVPR*, 2011.

[15] Behrooz Mahasseni and Sinisa Todorovic. Latent Multitask Learning for View-Invariant Action Recognition. In *ICCV*, 2013.

[16] Vasu Parameswaran and Rama Chellappa. View Invariance for Human Action Recognition. *IJCV*, 66(1), 2006.

[17] Bojan Pepik, Michael Stark, Peter Gehler, and Bernt Schiele. Teaching 3D Geometry to Deformable Part Models. In *CVPR*, 2012.

[18] Deva Ramanan and David A Forsyth. Automatic Annotation of Everyday Movements. In *NIPS*, 2003.

[19] Cen Rao, Alper Yilmaz, and Mubarak Shah. View-invariant Representation and Recognition of Actions. *IJCV*, 50(2), 2002.

[20] Leonid Sigal, Alexandru O. Balan, and Michael J. Black. Humaneva: Synchronized video and motion capture dataset and baseline algorithm for evaluation of articulated human motion. *IJCV*, 87(1-2), 2010.

[21] Michael Stark, Michael Goesele, and Bernt Schiele. Back to the Future: Learning Shape Models from 3D CAD Data. In *BMVC*, 2010.

[22] Alexander Thomas, Vittorio Ferrari, Bastian Leibe, Tinne Tuytelaars, Bernt Schiel, and Luc Van Gool. Towards Multi-view Object Class Detection. In *CVPR*, 2006.

[23] Heng Wang, Alexander Klaser, Cordelia Schmid, and Cheng-Lin Liu. Action recognition by dense trajectories. In *CVPR*, 2011.

[24] Daniel Weinland, Remi Ronfard, and Edmond Boyer. Free viewpoint action recognition using motion history volumes. *CVIU*, 104(2–3), 2006.

[25] Daniel Weinland, Edmond Boyer, and Remi Ronfard. Action Recognition from Arbitrary Views Using 3D Exemplars. In *ICCV*, 2007.

[26] Pingkun Yan, Saad M. Khan, and Mubarak Shah. Learning 4D Action Feature Models for Arbitrary View Action Recognition. In *CVPR*, 2008.

[27] Alper Yilmaz and Mubarak Shah. Actions sketch: A novel action representation. In *CVPR*, 2005.

[28] Zhong Zhang, Chunheng Wang, Baihua Xiao, Wen Zhou, Shuang Liu, and Cunzhao Shi. Cross-View Action Recognition via a Continuous Virtual Path. In *CVPR*, 2013.

[29] Jingjing Zheng and Zhuolin Jiang. Learning View-Invariant Sparse Representations for Cross-View Action Recognition. In *ICCV*, 2013.