A Feature Extraction Approach to Handle Variations in Camera Viewpoint for Computer Vision Tasks

Dinh Tuan Tran, Hirotake Yamazoe, Nobutaka Shimada, Joo-Ho Lee
College of Information Science and Engineering, Ritsumeikan University, Japan

***Abstract*— In this paper, an action recognition method that can adaptively handle the problems of variations in camera viewpoint is introduced. Our contribution is three-fold. First, a space-sampling algorithm based on affine transform in multiple scales is proposed to yield a series of different viewpoints from a single one. A histogram of dense optical flow is then extracted over each fixed-size patch for a given generated viewpoint as a local feature descriptor. Second, a dimension selection procedure is also proposed to retain only the dimensions that have distinctive information and discard the unnecessary ones in the feature vector space. Third, to adapt to a situation in which video data in multiple viewpoints are used for training, an extended method with a voting algorithm is also introduced to increase the recognition accuracy. By conducting experiments using both simulated and realistic datasets[1], the proposed method is validated. The method is found to be accurate and capable of maintaining its accuracy under a wide range of viewpoint changes. In addition, the method is less sensitive to variations in subject scale, subject position, action speed, partial occlusion, and background. The method is also validated by comparing with state-of-the-art view-invariant action recognition methods using well-known *i3DPost* and *MuHAVi* public datasets.**

I. Introduction

In the fields of computer vision and machine learning, action recognition (AR) is a process of assigning an action data (i.e., video) captured via a single-camera or a multi-camera system into an action label (i.e., action class). The AR scheme has been an important and popular topic in many applications, such as human-computer interaction [1] and healthcare monitoring system [2]. However, AR still remains challenging for realistic environments due to several key problems. In particular, the variations in camera viewpoint remain the main challenge. Almost all previous methods are based on fixed viewpoints. In a real-world environment, the viewpoint may change when the action zone changes. For example, the operating rooms in different hospitals have different features in terms of room design, equipment arrangement, etc. When a surgical workflow analysis system, such as the system in [2], is deployed to various operating rooms, the camera is needed to be re-arranged so that it fits into each room and does not affect the operation adversely. In such a case, the camera cannot be placed in a fixed viewpoint, allowing only an approximate location. Therefore, an AR method that is invariant under a certain range of angles would be more realistic and usable.

In this study, an AR method sequentially using multi-scale affine transform, optical flow (OF) feature, histogram of optical flow (HOF) feature descriptor, *k*-means clustering, and dimension selection, is proposed. The intent is to address the problem of variations in camera viewpoint. Here, each step has its own meaning. There are two central ideas behind this work. The first one is to prevent a significant decrease in AR accuracy when the viewpoint changes by using the affine transform to generate local features (i.e., OFs) for multiple viewpoints from a single one. The second one is to improve the accuracy by proposing the *dimension selection* procedure to preserve only the dimensions that are meaningful for the recognition and reduce the unnecessary information in the feature vector space. The purpose of performing the affine transform in multiple scales is to make the method less sensitive to subject scale variations. The OF feature is utilized to capture local movement information in an action video. The HOF and *k*-means clustering steps make the method more robust to noise and some variations, such as subject position and action speed. This work focuses on the viewpoint variation problem within a certain range of angles, rather than all possible angles due to the practical reason described above. It is also experimentally verified that the proposed method is less sensitive to variations in subject scale, subject position, action speed, partial occlusion, and background. In addition, a voting algorithm is also proposed to adapt to a situation which uses video data in multiple viewpoints for training. Finally, well-known *i3DPost* and *MuHAVi* public datasets are adopted to compare the averaged accuracy of the proposed method with that of state-of-the-art view-invariant action recognition methods.

II. Related Work

There are numerous publications in the field of view-invariant AR [3]–[6]. Zhang et al. [3] overcame the view-variance problem by learning transfer dictionary using a synthetic 3D and 2D video database constructed from human models. Zheng et al. [4] proposed two approaches, which utilize a dictionary for each viewpoint or just a common dictionary shared across multiple viewpoints for the cross-view AR. Chaaraoui et al. [5] introduced an action representation based on the sequences of key poses, which are actually contour points of human silhouettes. Moreover, Yang et al. [6] proposed a feature selection procedure, named multi-view rank minimization-based Lasso (MRM-Lasso). The MRM-Lasso jointly uses Lasso for sparse feature selection, and rank minimization to learn appropriate patterns through viewpoints. In practice, these methods require video data with action labels in multiple viewpoints including a set of test viewpoints for training purposes. Consequently,

[1] http://www.aislab.org/index.php/en/mvar-datasets

when the camera is moved to a different space, first one needs to record the action data in the new viewpoint, and the training phase is then executed again prior to testing. These methods were not evaluated for some key variations, such as partial occlusion–deemed to be an important benchmark. For instance, silhouette extraction in [5] could be significantly affected by the presence of partial occlusion. Other recent works on the view-variance problem have been introduced for AR in [7], [8], and for other topics in [9]–[11].

In [12], Kong et al. proposed a novel deep learning model to learn discriminative view-specified and view-shared features that are robust to viewpoint variations. The view-specified feature extracts the unique dynamics for each viewpoint, while the view-shared feature encodes common patterns between multiple viewpoints. Furthermore, some other methods using deep learning have been proposed to overcome the problem of variations in camera viewpoint for AR in [13], [14], and for other topics in [15]–[18]. However, in almost all deep learning based approaches, a large set of labeled training data is required. In addition, incomplete or inaccurate input data will simply yield wrong results.

The view-unconstrained AR method that is introduced in this paper, does not require such a large set of training data. It employs affine transform, HOF, and *k*-means. Chen et al. [19] adopted the method of affine transform for an encryption algorithm using hyperspectral data in fractional Fourier domain. Gardezi et al. [20] used Affine Scale Invariant Feature Transform (ASIFT) which is an extension of affine transform algorithm to enhance the performance of the spatial correlation filter (SPOT MACH filter). The affine transform method was also employed by Xue et al. [21] for a dissimilarity measure to match anisotropic-scale junctions that are detected by approximately calculating the endpoints of branches across separated indoor frames. Similarly, the HOF feature descriptor is extensively utilized in many motion related investigations. Recently, Xia et al. [22] combined Temporal Convolutional Neural Network (TempCNN) models with optical flow to detect local anomalies by tracking CNN features over time. Happy et al. [23] took into account the OF feature to build fuzzy HOFs for recognizing micro-expression. In essence, this method analyzes the rapid involuntary facial movements to reveal the genuine feelings of an actor. In [24], HOF was incorporated with its orientation, velocity, and entropy to detect unusual events in videos of crowd scenes by recognizing the patterns that might lead to such events. Fuente-Tomas et al. [25] developed a classification based on *k*-means clustering that allocates patients according to their severity for helping clinicians in personalized and shared decision processes of bipolar disorder. Dubey et al. [26] used *k*-means to compare with the fuzzy *c*-means clustering algorithm on breast cancer data–one of the most common cancers in the world. Kant et al. [27] applied *k*-means in a recommender system based on collaborative filtering to rank various products.

III. Proposed Method

A key problem, not only for AR but also for any recognition task in computer vision under a camera viewpoint change is that, the distance between different viewpoints in the same action class is larger than those between different classes. There are two general approaches to solve this problem. The first approach is to decrease the distance of various viewpoints in the same action class. On the other hand, the second approach is to increase the distance of various classes. In this work, two methods are proposed to adopt these approaches. The first method utilizes the affine transform at multiple scales for a single viewpoint with the purpose of increasing the capacity while dealing with a significant viewpoint change. The second method eliminates the dimensions in a K-dimensional feature vector space without which the distance among different classes increases. The details of both methods are given in Sect. III-A and Sect. III-E.

By integrating the above two methods with the HOF feature descriptor and *k*-means clustering algorithm, a viewpoint unconstrained AR method is proposed with the flow charts shown in Figs. 1, 2, and 5. There are training and testing phases. In the training phase, the video data for all action classes in a specified camera viewpoint are processed. Firstly, each video is transformed by using the affine transform technique in multiple scales (Sect. III-A) to generate N_f videos in total. Secondly, dense OF is extracted between two consecutive frames of each video in generated video collection (i.e., bag). Gunnar Farneback's algorithm [28] is adopted for this OF calculation (Sect. III-B). Thirdly, histograms for OFs are built and normalized in a block unit as described in Sect. III-C. The output of this step is the collections of HOF vectors for all action videos in the training viewpoint. Subsequently, *k*-means clustering algorithm is used to organize all training HOF vectors into K clusters (Sect. III-B). Each HOF vector thus corresponds to a cluster index, and each cluster index corresponds to a cluster center. As a result, a collection of *affined* multi-scale features (i.e., cluster indexes) is gained from an input video of a single viewpoint. In reality, each of these collections can be represented by a histogram of clusters as a video descriptor vector. To complete the training phase, all training video descriptor vectors are passed through one more step, namely the *dimension selection* step, to reduce the non-essential dimensions, and to preserve only the useful ones. This step yields the final video descriptor feature for each video in all the classes with the remaining dimensions as well as the trained *k*-means cluster centers for further AR testing. In the testing phase, each video data of an unknown action class in a different viewpoint is processed in the same way–except for the *k-means clustering* and *dimension selection* steps. More specifically, the testing HOF vectors are mapped with the nearest trained cluster center to get a corresponding cluster index. The remaining dimensions in the training phase are utilized to obtain the final video descriptor vector without any new cluster-training or dimension-selecting processes. Finally, the action class for the input testing video is identified to be

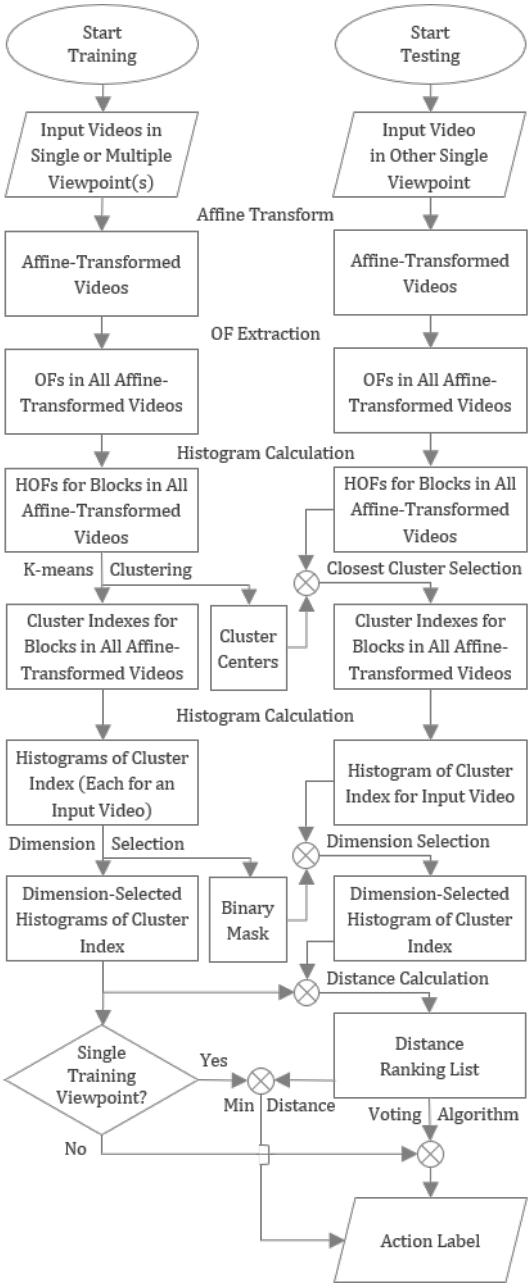

Fig. 1: Overview of the proposed method for both training and testing phases (different camera viewpoint(s) for each phase).

the one, which has the closest video descriptor vector to its own video descriptor vector using the Euclidean distance. In the case of multiple training viewpoints, a voting procedure is used to decide the final action class for the testing video. The details of the proposed method are described in the following subsections.

A. Multi-Scale Affine Transform

Firstly, a set of various viewpoints is generated from a single one using the affine transform A that includes four parameters as presented in (1). In addition, the geometric relationship of these parameters is illustrated in the left of Fig. 3. Here, ω_r^o is the camera rotation angle according to its roll axis. Note that this work focuses on the AR tasks in which the camera is fixed while performing an action. When the action is performed in a different space (e.g., room), the camera is needed to be moved and re-arranged. It is difficult to have the exact same camera viewpoint due to the design of that space. However, the camera can be placed in parallel or nearly parallel with the space. In other words, the roll angle ω_r^o of the camera can be assumed to be always zero, and it can be ignored for the affine transform. The remaining three parameters are scale factor λ, tilted subject rotation angle t, and longitude subject rotation angle ϕ. These are needed to be considered for a full affine

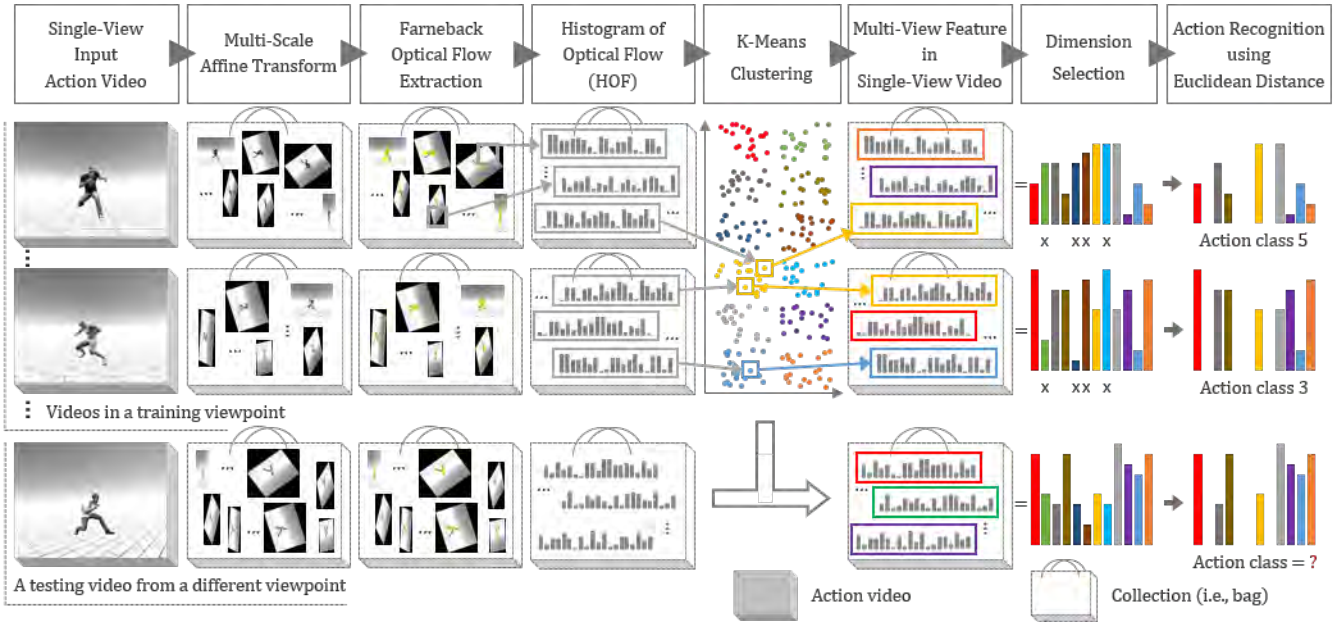

Fig. 2: Overview of the proposed method for both training and testing phases in the case of single training viewpoint (different camera viewpoint for each phase).

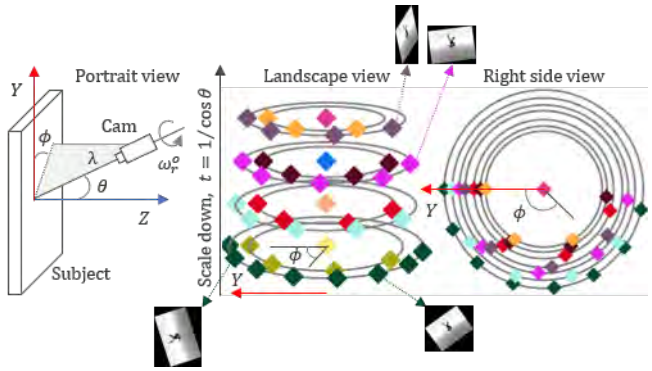

Fig. 3: Overview of *multi-scale affine transform*. Geometric relationship of affine decomposition on the left, and sampling process illustrations in the middle and on the right. Each circle corresponds to a tilt t value, each colored diamond indicates a pair of t and longitude ϕ, and generated viewpoints having the same t are illustrated by the same color.

invariant. The latitude θ in Fig. 3 is equivalent to the tilt t, and can be calculated via $\theta = \arccos 1/t$. From a single viewpoint, a sampling process is performed by changing these parameters such that there is a good compromise between accuracy and sparsity.

$$A = \lambda \begin{bmatrix} \cos\omega_r^o & -\sin\omega_r^o \\ \sin\omega_r^o & \cos\omega_r^o \end{bmatrix} \begin{bmatrix} t & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\phi & -\sin\phi \\ \sin\phi & \cos\phi \end{bmatrix}. \quad (1)$$

More specifically, every frame in each action video is resized by multiple pairs of λ_x and λ_y scale factors. This process is to make the method more robust to variations in subject scale. While λ_y varies from $\lambda_y^{min} = 0.2$ to $\lambda_y^{max} = 0.5$ by a sampling step of $\lambda_y^{step} = 0.1$, λ_x changes from $\lambda_y - \Delta\lambda_x$ to $\lambda_y + \Delta\lambda_x$ by a step of $\lambda_x^{step} = 0.1$, where $\Delta\lambda_x = 0.1$ is a constant. At each pair of λ_x and λ_y, the frame after being resized is denoted as f. Here, w and h correspond to its width and height, respectively. The interpolation method, which is used for resizing frames is *inter-area*, applies a resampling technique using pixel area relation. The *inter-area* gives moire'-free results, and is preferred for downsampling (i.e., downscaling) frames. After that, the frame f is converted to grayscale and then *affined* by the steps outlined below. In addition, this sampling process is illustrated in Fig. 3.

1) Calculate the tilt factor t via (2). Each t corresponds to a circle in Fig. 3.
2) Loop over the longitude rotation ϕ from 0 to $a * b/t$ by a sampling step of b/t such that $a * b/t < 180\,deg$, where $b = 72.0\,deg$ is a constant indicating rotation angle.
3) At each pair of t and ϕ (i.e., a colored diamond in Fig. 3, each color corresponds to a t value), 2x3 transform matrix is calculated based on the rotation factor ϕ, and then the affine transform is applied to the frame f using the calculated matrix with the *bilinear* interpolation and *border-replicate* pixel extrapolation, in which the row or column at the very edge of the original frame f is replicated to the extra border.
4) Finally, a set of N_f affined frames f_i, $i\{0, ..., N_f - 1\}$ (i.e., N_f diamonds in Fig. 3) is obtained from the original frame f. Each frame f_i has a size of (w_{f_i}, h_{f_i}) and the frame f_0 (i.e., $i = 0$) is exactly the original f. In other words, a collection of N_f multi-scale, multi-view videos can be yielded from a single-view video, i.e., $N_f = 49$ in this work.

$$t = \frac{\lambda_y + \Delta\lambda_x}{\lambda_x}. \quad (2)$$

B. Farneback Optical Flow Extraction

In computer vision based AR tasks, one of the most common and effective approaches is to extract the motion information from every movement of the subject performing an action. Optical flow (OF) is a well-known local motion feature. In this research, after generating N_f affined videos for each single-view one, a dense OF is computed from every pair of two consecutive grayscale affined frames in an affined video using Gunnar Farneback's algorithm [28]. Unlike a sparse OF extraction, estimation OFs on a dense grid of pixels or all pixels in the frame, was proposed in [28]. Here, all parameters in the Gunnar Farneback's algorithm are experimentally configured with

- three pyramid layers including the initial frame,
- each next layer is twice smaller than the previous one,
- the averaging window size is 15,
- the number of iterations the algorithm does at each pyramid level is three,
- the pixel neighborhood size used to find polynomial expansion in each pixel is five,
- the Gaussian standard deviation that is utilized to smooth derivatives used as a basis for the polynomial expansion is 1.2, and
- the initial OF approximation is the output of the previous OF computation.

From the local dense OF features (i.e., yellow vectors in Fig. 4) calculated above, all the OFs close to the boundary of the affined frame f_i support (i.e., parallelogram in Fig. 4) are then removed by a distance threshold. The distance thresholds for both width and height dimensions are set to $0.2{*}w_{f_i}$ and $0.2{*}h_{f_i}$, respectively. Moreover, the OFs whose magnitude $l_{i,j}$ is smaller than a threshold $\epsilon_l = 0.5$, are eliminated for noise abatement purposes.

C. Histogram of Optical Flow

As mentioned earlier, the main idea behind this work is to generate local features for multiple viewpoints from a single one. The local feature is an OF as described in Sect. III-B. A feature descriptor is also needed for that OF. In this research, the histogram of optical flow (HOF) is adopted for the local feature descriptor. Indeed, the human body is not planar as it contains significant depth variations. Therefore, a local descriptor like HOF can adapt to this situation by separating the frame into smaller patches called *blocks*, and treating them as flat surfaces. In this way, the affine transform described in Sect. III-B can work well for a significant change in camera viewpoint.

Fig. 4 illustrates the details of the HOF calculation process. HOF for each affined frame f_i is calculated based on the previously computed dense OFs. Firstly, a frame f_i with a size of (w_{f_i}, h_{f_i}) pixels is divided into $(n^{i,x}_{block}, n^{i,y}_{block})$ (orange) blocks and $(n^{i,x}_{cell}, n^{i,y}_{cell})$ (purple) cells by a grid of vertical and horizontal (blue) lines. Each block consists of multiple cells. For convenience, the cell size is denoted as (s^x_{cell}, s^y_{cell}) (e.g., $(6,6)$ in Fig. 4 and $(16,16)$ for experiments in Sect. IV) pixels, the block size is denoted as $(s^x_{block}, s^y_{block})$ (e.g., $(2,2)$ in Fig. 4 and $(4,4)$ for experiments in Sect. IV) cells. Here, $(n^{i,x}_{cell}, n^{i,y}_{cell})$ and $(n^{i,x}_{block}, n^{i,y}_{block})$ are calculated using (3) and (4), respectively. As the remainder of a division between the frame size and the cell size may not always be equal to zero, all the pixels in four borders (i.e., red rectangle in Fig. 4) are symmetrically cropped such that the remainder of the division is forced to zero. Secondly, all OFs remaining after Sect. III-B are normalized in a clockwise direction into n_{bin} (e.g., five bins in Fig. 4 and four bins for experiments in Sect. IV) using (5). After the normalization is applied, a j-th OF in frame f_i has two information: flow magnitude $l_{i,j}$ and normalized direction $\alpha_{i,j}$. Thirdly, HOF with a fixed number of bins (i.e., n_{bin}) is calculated for each cell. For example, a cell HOF in Fig. 4 contains five bins. After that, all cell HOFs in each block are concatenated to a block HOF that is essentially a vector of $s^x_{block} * s^y_{block} * n_{bin}$ dimensions. The block HOF vector is finally normalized by dividing its vector length into each of its bin value (i.e., L_2 norm).

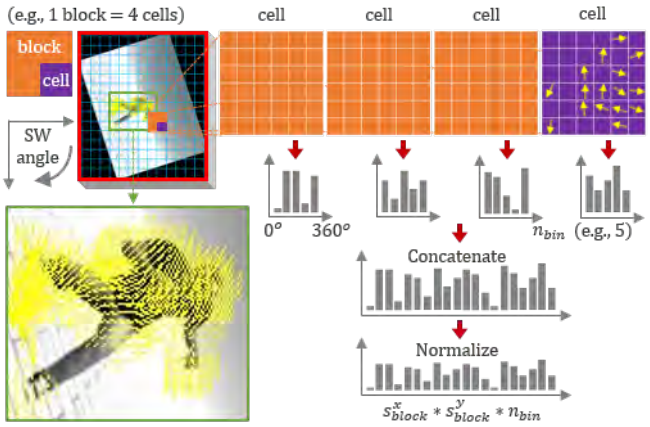

Fig. 4: An example of the calculation procedure for histogram of optical flow. Cell size is $(6,6)$ pixels, block size is $(2,2)$ cells, and dense optical flow is normalized into five bin dimensions in the clockwise direction.

$$n^{i,x}_{cell} = w_{f_i}/s^x_{cell},\ n^{i,y}_{cell} = h_{f_i}/s^y_{cell}, \quad (3)$$

$$n^{i,x}_{block} = n^{i,x}_{cell} - s^x_{block} + 1,\ n^{i,y}_{block} = n^{i,y}_{cell} - s^y_{block} + 1, \quad (4)$$

$$\alpha_{i,j} = \arg_d \left(d{*}\frac{360}{n_{bin}} \le \alpha_{i,j} < (d+1){*}\frac{360}{n_{bin}} \right),\ d\{0, ..., n_{bin}-1\}. \quad (5)$$

Because the individual dense OFs may contain noise, and are sensitive to overall lighting, calculating a HOF over each patch makes this descriptor more robust to noise and subject position variations. Especially, normalizing the block HOFs makes them less sensitive to variations in action speed and lighting. Noise can be eliminated more by ignoring the blocks where the number of valid OFs is smaller than a threshold ϵ_n.

$$\epsilon_n = \min\left(s^x_{cell}, s^y_{cell}\right) * \min\left(s^x_{block}, s^y_{block}\right). \quad (6)$$

D. K-Means Clustering

K-means clustering is a method of vector classification that solves a well-known clustering problem in data mining. In unsupervised learning, k-means is one of the simplest algorithms where it follows a straightforward procedure for classifying a given set of feature vectors into a certain number of clusters (i.e., K clusters) where each feature vector belongs to the cluster with the nearest center.

In this work, k-means is utilized to map a vector in $s^x_{block} * s^y_{block} * n_{bin}$ dimensional space into an one-dimensional vector (i.e., number). This can be treated as a normalization process to reduce noise and handle movement variations in the same action class. More specifically, k-means trains K cluster centers corresponding to K clusters (i.e., groups) in the training phase. All normalized block HOF vectors in both the training and testing phases are then mapped with the nearest trained cluster center. Thus, it is able to represent each HOF vector via a corresponding cluster index. After this step, a collection of multi-scale and multi-view features which is actually a bag of cluster indices, is obtained from a single viewpoint input video. Each of these collections is essentially a distribution (i.e., histogram) vector of clusters that is treated as a video feature descriptor (see Fig. 2). For convenience, the number of action classes is denoted as N, the number of dimensions in the final feature vector space that is utilized for video descriptors is denoted as K. Here, K is actually the number of clusters in k-means. The descriptor vector $v_{i,j}$ is for the j-th video in the i-th action class, where $i\{0, ..., N-1\}$ is presented as (7).

E. Dimension Selection

Ideally, the distance among all different action classes should be increased to be as long as possible. Several dimensions in the feature vector space is thus discarded to accomplish this. This process is called *dimension selection*. Note that it differs from *dimension reduction* processes, such as Principal Component Analysis (PCA) [29] which is often used to reduce dimensions of a vector space by transforming that vector space into a smaller one that still maintains most of the information in the original vector space. More specifically, in the training phase, after achieving the K-dimensional descriptor vector $v_{i,j}$ for each j-th video in the i-th action class, the k-th dimension where $k\{0, ..., K-1\}$ is sequentially gotten out. The $(K-1)$ dimensional vector is then re-normalized by dividing its length into the value in each dimension to yield a new vector $v_{i,j,-k}$. An example of identifying the first dimension (i.e., $k = 0$) is illustrated in (8). Subsequently, the sum of distances among all pairs of different classes is re-calculated as presented in (9) below. If the original sum of distances before identifying the dimensions is s as given in (10), and a K-dimensional binary mask vector m is also declared to indicate whether a dimension is retained (i.e., $m^k = 1$) or not (i.e., $m^k = 0$) as in (11), then each element m^k in the mask m will be decided using (12). Here, ϵ_s is a threshold (e.g., $\epsilon_s = 10^{-6}$ in Sect. IV-A, $\epsilon_s = 1.6 * 10^{-3}$ in Sect. IV-B, and $\epsilon_s = 2 * 10^{-3}$ in Sect. IV-C). Finally, the descriptor vector $v_{i,j}$ considering only K' retained dimensions is re-normalized by using L_2 norm to obtain the final K'-dimensional descriptor vector (i.e., $v'_{i,j}$) for each j-th video in the i-th action class. This binary mask vector m will be further used to ignore dimensions (i.e., $m^k = 0$) for input video in the testing phase (see 13).

$$v_{i,j} = (v^0_{i,j}, v^1_{i,j}, ..., v^{K-1}_{i,j}), \tag{7}$$

$$v_{i,j,-0} = (v^1_{i,j,-0}, v^2_{i,j,-0}, ..., v^{K-1}_{i,j,-0}),\ |v_{i,j,-0}| = 1, \tag{8}$$

$$s_{-0} = \sum_{i,j,i',j'} ||v_{i,j,-0} - v_{i',j',-0}||,\ i, i'\{0, ..., N-1\}, i \neq i', \tag{9}$$

$$s = \sum_{i,j,i',j'} ||v_{i,j} - v_{i',j'}||,\ i, i'\{0, ..., N-1\}, i \neq i', \tag{10}$$

$$m = (m^0, m^1, ..., m^{K-1}), \tag{11}$$

$$m^0 = \begin{cases} 1 & \text{if } \frac{s_{-0} - s}{s} \leq \epsilon_s \\ 0 & \text{otherwise} \end{cases}. \tag{12}$$

Consequently, these K'-dimensional video descriptor vectors in the training phase are then used to compare with the K'-dimensional video descriptor vector of an input video in the testing phase to output an action label (i.e., class) for that video. In other words, the output action label c_{v_t} for a testing video v_t is the nearest one that is calculated using the Euclidean distance as shown in (13) bellow. This is known as the AR process.

$$c_{v_t} = \text{argmin}_i \sum_{k=0}^{K-1} m^k * (v^k_t - v^k_{i,j})^2. \tag{13}$$

F. Multiple Training Viewpoints

There is a fact that a room is often equipped with multiple cameras instead of just one camera. In such a case, the cameras are often located around the room to fully cover the actor performing actions from multiple viewpoints. Therefore, the AR accuracy can be further improved by using the recognition result from each camera and the relative position relationship between these cameras wisely. In this section, an extended method is proposed to adapt to this situation as shown in Fig. 5. There are two changes to the method proposed in Fig. 2. The first one is that the action video data in all available viewpoints instead of just the data in one viewpoint, are utilized for training. The second one is the last two steps (i.e., *distance ranking based on Euclidean distance* and *action recognition using voting algorithm*) in Fig. 5. They are used instead of the last step (i.e., *action recognition using Euclidean distance*) in Fig. 2.

More specifically, for each K'-dimensional video descriptor vector of an input video v'_t in the testing phase, R video descriptor vectors from all viewpoints in the training phase that have closest Euclidean distances to the testing video v'_t, are considered as candidates to decide the final output action label. The output label is voted by using the voting procedure proposed in Algorithm 1. It requires $R + 2$ scoring coefficients and three scoring steps to accomplish this. In the first step (from line 4 to line 6 in Algorithm 1), R first coefficients from $\mu_{0,0}$

Fig. 5: Overview of the proposed method for both training and testing phases in the case of multiple training viewpoints (different camera viewpoint(s) for each phase).

TABLE I: Parameter setting in Algorithm 1 for experiments on *i3DPost* dataset in Sect. IV-B and *MuHAVi* dataset in Sect. IV-C.

Dataset	R	$\mu_{0,0}$	$\mu_{0,1}$	$\mu_{0,2}$	$\mu_{0,3}$	$\mu_{0,4}$	$\mu_{0,5}$	$\mu_{0,6}$	$\mu_{0,7}$	$\mu_{0,8}$	μ_1	μ_2
i3DPost with 6 actions	6	1.0	0.4	0.2	0.1	0.0	0.0	-	-	-	1.2	0.6
i3DPost with 10 actions	7	1.0	0.4	0.2	0.1	0.0	0.0	0.0	-	-	1.2	0.6
MuHAVi with 17 actions	9	2.0	0.8	0.4	0.2	0.1	0.0	0.0	0.0	0.0	2.4	1.2

to $\mu_{0,R-1}$ are used to score actions that appeared in the distance ranking list $\mathbb{L}_R$ based on their positions in $\mathbb{L}_R$. This means that, if a training video $v'_{i,j}$ is closer to the testing video v'_t than another $v'_{i',j'}$, then the corresponding action class c_i will gain more score than $c_{i'}$. In the second step (from line 7 to line 9), a score μ_1 is given to an action class c_i for each pair of videos in $\mathbb{L}_R$ that have the same class c_i and were recorded from the same viewpoint. This is because from our point of view, if there exist such videos in the ranking list, action class and viewpoint of the testing video could be similar to those of such videos. Similarly, if there is a group of three training videos in $\mathbb{L}_R$ that has the same class but was recorded from three neighbor viewpoints, the testing video could has the same class and viewpoint with the video in the middle viewpoint of that group. Hence, a score μ_2 should be given to that class in such a case (see the third step from line 10 to line 12 in Algorithm 1). All parameters in Algorithm 1 are configured for experiments on *i3DPost* dataset in Sect. IV-B and *MuHAVi* dataset in Sect. IV-C as shown in Table I. An example of the input and output for Algorithm 1 is shown in Fig. 6.

IV. Experiments

In this section, the details of the experiments that were conducted to evaluate the proposed method on both custom-made and public datasets are provided. On the custom-made datasets, the effectiveness of each contribution in this work as well as the invariant range in viewpoint for the proposed method using singular training viewpoint in Fig. 2 were evaluated. On the other hand, comprehensive comparison results with state-of-the-art view-invariant AR methods are given by using the proposed method for multiple training viewpoints in Fig. 5 with well-known public datasets named *i3DPost* [30] and *MuHAVi* [31].

A. Experiments on Custom-Made Datasets[1]

Because there are two major contributions for the method of singular training viewpoint in this work (*multi-scale affine transform* and *dimension selection*), the experiments on four methods were performed to assess the impact of these steps on the AR accuracy under a significant change in the viewpoint. As denoted in Table III, the first method is the simplest method

Algorithm 1: Proposed voting algorithm for multiple training viewpoints.

Input: Scoring coefficients $\mu_{0,0}$, $\mu_{0,1}$, ..., $\mu_{0,R-1}$, μ_1, μ_2, and a distance ranking list $\mathbb{L}_R = \left\{(c_{i,j,r}, \curlyvee_{i,j,r}, \boldsymbol{d}_{i,j,r})_r\right\}$ sorted by ascending $\boldsymbol{d}$ value, in which

- N is the number of action classes,
- R is the number of videos in $\mathbb{L}_R$,
- $i\{0, ..., N-1\}$, $r\{0, ..., R-1\}$,
- $c_{i,j,r}$ is the i-th action class (i.e., ground truth) for its j-th training video (i.e., $v'_{i,j}$),
- $\curlyvee_{i,j,r}$ is the camera viewpoint from which the video $v'_{i,j}$ was recorded, and
- $\boldsymbol{d}_{i,j,r}$ is the Euclidean distance from the testing video v'_t to the training video $v'_{i,j}$.

Output: Scoring list $\mathbb{L}_N = \{(c_i, z_i)_i\}, i\{0, ..., N-1\}$ and an action class label c_i that has highest score z_i in $\mathbb{L}_N$.

Initialization: for $i = 0$ *to* $N-1$ **do**
  Set z_i to zero
end
Step 1: for $r = 0$ *to* $R-1$ **do**
  Increase the corresponding z_i by the scoring coefficient $\mu_{0,r}$
end
Step 2: if *exist a pair of r and r′ that* $c_{i,j,r}$ *is equal to* $c_{i',j',r'}$ *(i.e., i is equal to i′) and* $\curlyvee_{i,j,r}$ *is equal to* $\curlyvee_{i',j',r'}$ **then**
  Increase the corresponding z_i by the scoring coefficient μ_1
end
Step 3: if *exist three r, r′, and r″ that i is equal to both i′ and i″;* $\curlyvee_{i,j,r}$, $\curlyvee_{i',j',r'}$, *and* $\curlyvee_{i'',j'',r''}$ *are three consecutive (i.e., neighbor) viewpoints* **then**
  Increase the corresponding z_i by the scoring coefficient μ_2
end

TABLE II: Scenarios of variations and their video sizes in simulated *MVAR-Unity3D Attack* (C_0 to C_7) and realistic *MVAR-GoPro Standing Exercise* (C_R) datasets.

	Variation	Videos		Variation	Videos
C_0	None	1,520	C_4	Lighting condition	1,520
C_1	Subject scale	1,520	C_5	Partial occlusion	1,520
C_2	Subject position	1,520	C_6	Background	1,520
C_3	Action speed	1,520	C_7	Combined	1,520
C_R	*MVAR-GoPro*	480		Total	12,640

called *NONE*, which does not use either *affine transform* or *dimension selection* steps (shown in Fig. 2). In other words, the number of affined viewpoints N_f in this method is one. The remaining steps including OF extraction, HOF calculation, and k-means clustering are performed only on the original input video. In addition, all dimensions in the final feature vector space are retained for recognizing the action. The second proposed method (i.e., *DS*) excludes the *affine transform* step. The essential dimensions are selected but no new viewpoint is generated. Conversely, the third one named *AT* has the *affine transform* but excludes the *dimension selection*. The last one (proposed exclusively in this work, called *BOTH*) includes all the steps shown in Fig. 2.

The aim of these experiments is to monitor how the accuracy changes when the camera is moved around the action subject. To accomplish this, of custom-made datasets, action videos in only one viewpoint are used for training, videos in each other viewpoint are sequentially used for testing. As explained previously, the camera angle on the roll axis was assumed to be always zero. Since the remaining two axes are equivalent and have the same properties, only the viewpoint difference problem for the yaw axis was considered. The experimental results for the pitch axis would be the same. For more specific evaluations, the camera should be moved to as many places as possible, and also separately measure the impact of other variations including subject scale, subject position, action speed, lighting condition, partial occlusion, as well as the background. However, it is difficult to record a dataset with such requirements in a realistic environment. To accomplish this, a simulated dataset named *MVAR-Unity3D Attack*[1] (*MVAR* stands for *Multiple Viewpoints Action Recognition*) was first built with many camera viewpoints and scenarios of variations (e.g., scenario of variations in partial occlusion), and the proposed method was evaluated on it. The results were then confirmed by using a much smaller self-recorded realistic dataset named *MVAR-GoPro Standing Exercise*[1].

The parameter settings as described in each step of Sect. III were used. In addition, the number of k-means clusters K utilized for each of the four methods is shown in Table III. These values of K were experimentally derived after testing a wide range of K from 200 to 20,000 by a step of 200. In both datasets, the videos have a resolution of 640x480 pixels. The number of action classes N is eight (see Tables IV and V). Videos in the viewpoint with the roll angle $\omega_y^o = 0$ (i.e., *Cam* 0 in Fig. 10) were used for the training phase. The remaining in one another viewpoint were sequentially used for testing.

1) Experiments on MVAR-Unity3D Attack Dataset: To record this large and complex dataset, a simulation on Unity3D software based on 3D models made by Studio New Punch [32], Niandrei [33], Explosive [34], and RockVR [35] was conducted. The experimental setting is shown in Fig. 7. More specifically, a camera was utilized and rotated around the roll axis (i.e., Y axis) by a step $\Delta\omega_y^o$ of 5 deg from 0 to 90 deg to have 19 viewpoints. The viewpoint differences that are greater 90 deg and less than 270 deg were ignored in these experiments. This is because, in almost all recognition tasks, the camera is practically located in the front of the action subject. The viewpoints from 270 deg were also ignored as they are on the opposite side and would yield similar results. In this dataset,

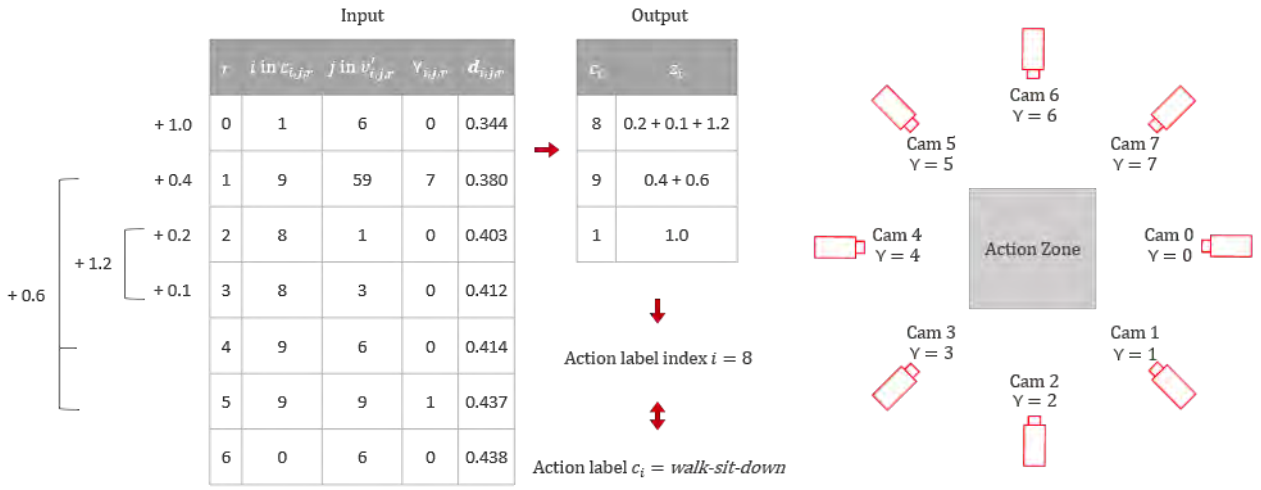

Fig. 6: An example of input and output for Algorithm 1 when the number of viewpoints is 8, the number of action classes is $N = 10$, the number of videos in distance ranking list is $R = 7$, scoring coefficients $\mu_{0,0} = 1.0$, $\mu_{0,1} = 0.4$, $\mu_{0,2} = 0.2$, $\mu_{0,3} = 0.1$, $\mu_{0,4} = 0.0$, $\mu_{0,5} = 0.0$, $\mu_{0,6} = 0.0$, $\mu_1 = 1.2$, and $\mu_2 = 0.6$.

TABLE III: Methods and their cluster sizes K, ignored dimension sizes in all scenarios of variations from C_0 to C_R.

	Method	K	C_0	C_1	C_2	C_3	C_4	C_5	C_6	C_7	C_R
NONE	None	1000	-	-	-	-	-	-	-	-	-
DS	Dimension selection	1000	349	362	389	370	288	378	349	395	317
AT	Affine transform	5000	-	-	-	-	-	-	-	-	-
BOTH	*DS* + *AT*	5000	572	577	551	580	556	560	540	574	379

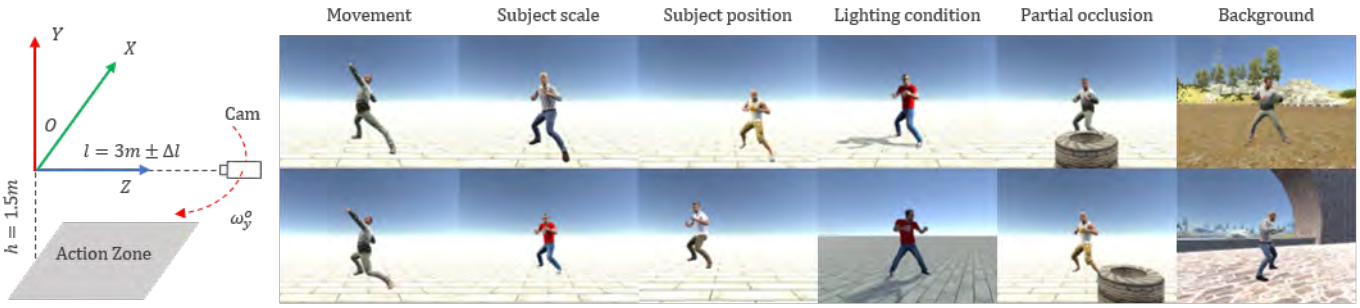

Fig. 7: Experimental setting and several examples of variations for simulated *MVAR-Unity3D Attack* dataset.

TABLE IV: Action classes in *MVAR-Unity3D Attack* dataset.

0	Right-hand-hook	4	Left-kick-attack
1	Right-hand-cross	5	Right-kick-attack
2	Left-hand-uppercut	6	Turn-left
3	Right-hand-uppercut	7	Turn-right

there are five characters with characteristic variables, such as height, head, body, and clothes. Each character performed each of the eight short action classes twice in each of eight scenarios of variations as listed in Tables II and IV. In total, the dataset consists of $19{*}5{*}8{*}8{*}2 = 12,160$ videos. There is $1,520$ videos for each scenario. Especially, for each frame of every video, a random function was used to make a change within a specified range of movements (for all parts, e.g., hands, legs, and head) even though these videos belonged to the same action class (see movement variations in the second column of Fig. 7). Note that these movement variations were applied to all frames in every video of all eight scenarios. The random function was also utilized to change the subject scale, position where the action was performed, action speed, lighting condition, location of partial occlusion, and different backgrounds. Fig. 7 also shows two examples for each of some variations.

Fig. 8 shows the change in accuracy for all four methods with various scenarios of variations when the camera was moved. Firstly, the results of the *NONE* and *BOTH* methods are considered. The figure indicates that the *BOTH* method

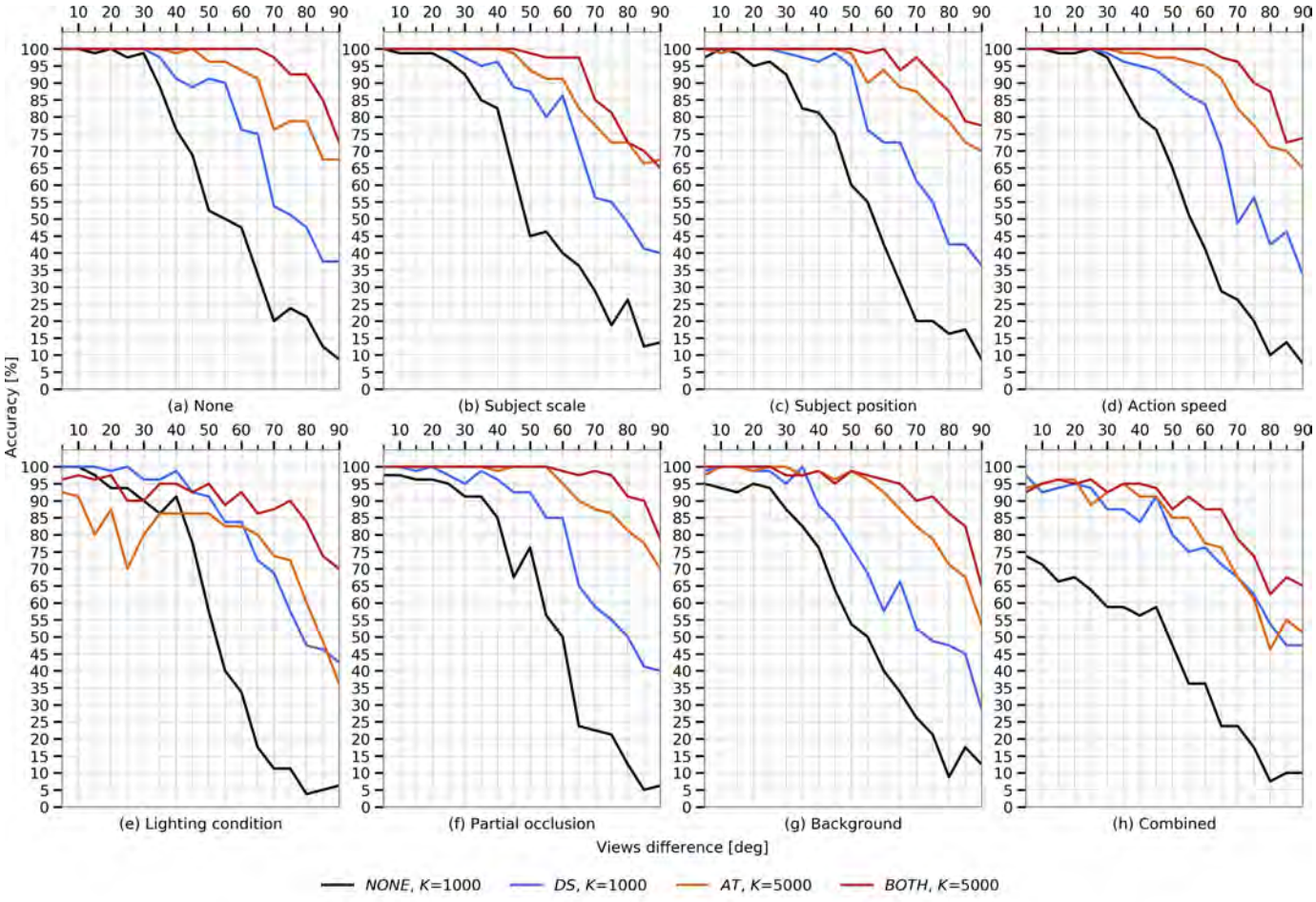

Fig. 8: Accuracy of four methods with eight scenarios of variations when the camera viewpoint changed in *MVAR-Unity3D Attack* dataset.

yielded much better results than the *NONE* method in almost all scenarios except for the very first viewpoint differences (0 to 30 *deg*) in the scenario of lighting condition variations. While the *NONE* method tended to decrease very quickly when the camera moved, the *BOTH* method was not only accurate, but also capable to maintain its accuracy from significantly falling down under a wide range of viewpoint changes (about 65 *deg* on average). It is observed from Fig. 8 that the blue line (*DS*) is always above the black one (*NONE*). This means that the *dimension selection* is an effective procedure to improve the accuracy. Furthermore, because the orange line (*AT*) is less steep than both blue (*DS*) and black (*NONE*) lines in all scenarios, the *multi-scale affine transform* is confirmed to make a major contribution by preventing the accuracy from dropping off in the *BOTH* method.

It is also observed from the very first viewpoint differences that the *NONE* method was significantly affected by variations, such as subject scale, subject position, partial occlusion, and background. The *BOTH* was only slightly affected (often unaffected) by such variations. This indicates that the *multi-scale affine transform* and *dimension selection* steps made the proposed method more robust to these variations. In particular, the *BOTH* method was less sensitive to the subject scale variations, since the local features (i.e., OFs) were extracted in multiple scales (Fig. 8b). It was robust to the variations of the subject position as every frame was separated into smaller blocks. Instead of the absolute position information, only the HOF local feature descriptor for each block was employed (Fig. 8c). The *AT* method also had a positive impact on the accuracy when the partial occlusion appeared (Fig. 8f). However, the impact was not clear in the lighting condition variations, whereas the *DS* method was effective in this case (Fig. 8e). It was quite similar for the variations in the background because the lighting condition might have changed according to the background (Fig. 8g). In addition, the variations in action speed did not affect both the *NONE* method and the *BOTH* method (Fig. 8d). This could be explained via the vector normalization in the HOF calculation.

Fig. 9 shows the confusion matrices for the four methods (with all scenarios of variations) when the viewpoint difference was 60 *deg*. Per our observation, in such a large viewpoint change, the *NONE* method was not unable to distinguish

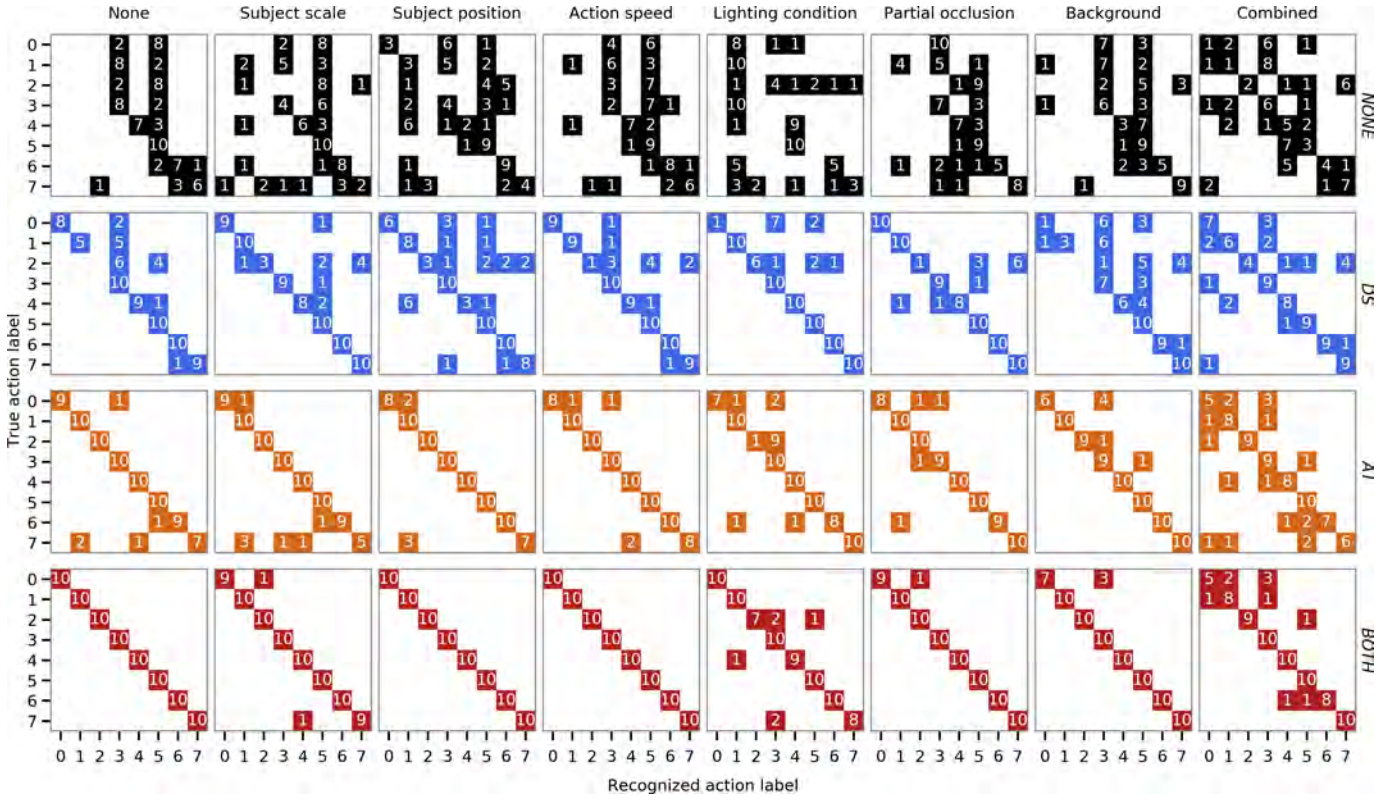

Fig. 9: Confusion matrices for four methods with eight scenarios of variations when the camera viewpoint changed 60 *deg* on the roll axis in *MVAR-Unity3D Attack* dataset.

TABLE V: Action classes in *MVAR-GoPro Standing Exercise* dataset.

0	Ankle-circles	4	Obliques-stretches
1	Neck-rolls	5	Knee-circles
2	Hip-circles	6	Torso-twists
3	Big-arm-circles	7	Jumping-jacks

some actions, such as *right-hand-hook* (0), *right-hand-cross* (1), and *right-hand-uppercut* (3), which are very similar and are hard to recognize. In addition, it did not work properly with other quite distinctive actions. For examples, it commonly confused *right-hand-hook* (0), *right-hand-cross* (1), and *left-hand-uppercut* (2) actions which use a hand, with *right-kick-attack* (5) which uses right leg; or *turn-left* (6) with *turn-right* (7), which are on opposite sides. In contrast, the *BOTH* method was able to accomplish this distinctions well, especially perfectly recognizing the *none* (C_0), *subject position* (C_2), and *action speed* (C_3) scenarios of variations.

2) Experiments on MVAR-GoPro Standing Exercise Dataset: Next, experiments on a realistic dataset were conducted to validate the overall results achieved from the simulated dataset as described in Sect. IV-A.1. The experimental setup is shown in Fig. 10. In this dataset, five GoPro cameras located at five viewpoints from 0 to 90 *deg* were utilized. The average difference between two neighbor viewpoints was 22.5 *deg*.

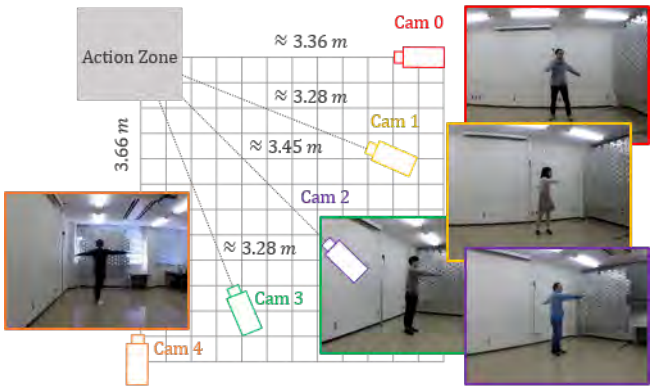

Fig. 10: Experimental setting for realistic *MVAR-GoPro Standing Exercise* dataset.

There were six actors and eight action classes of standing exercises as listed in Table V. Each action was performed twice by each actor. Hence, there were 480 videos in total. In these experiments, videos recorded via the *Cam* 0 in Fig. 10 was used for training. The final results are presented in Fig. 11. While the left side of the figure presents the performances of all methods for various viewpoint differences, the right side shows four confusion matrices for the four methods when the camera was located at the *Cam* 2, which has a viewpoint difference of 45 *deg*. These results are equivalent to the results

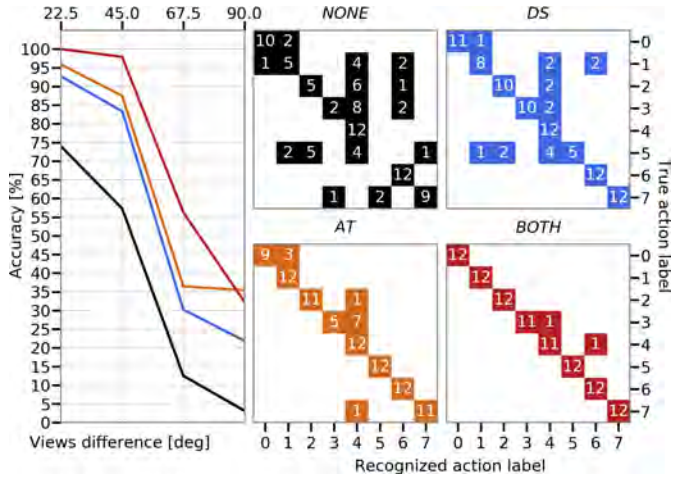

Fig. 11: Accuracy and confusion matrices for four methods when the camera viewpoint changed 45 *deg* on the roll axis in *MVAR-GoPro Standing Exercise* dataset.

TABLE VI: The number of clusters for each method in both subsets of *i3DPost* dataset and *MuHAVi* dataset.

Dataset	Method	K
i3DPost with 6 actions	*NONE*	600
	DS	600
	AT	3000
	BOTH	3000
i3DPost with 10 actions	*NONE*	1000
	DS	1000
	AT	5000
	BOTH	5000
MuHAVi with 17 actions	*BOTH*	5000

for the *combined* (C_7) scenario of variations in the simulated *MVAR-Unity3D Attack* dataset above. The *dimension selection* significantly enhanced the AR accuracy, while the *multi-scale affine transform* made the method more robust to the variations in camera viewpoint. From our observation, when the viewpoint difference was 22.5 *deg*, the *BOTH* method recognized all the actions perfectly, whereas the accuracy for the *NONE* method was just 74 %. The accuracy of the *BOTH* method decreased by only 2 % from 100 % to 98 % when the difference increased up to 45 *deg*, while in the *NONE* method it dropped 17 % from 74 % to 57 %. The view-invariant range in this dataset was about 45 *deg*. This is a promising result because as mentioned in Sect. I, this work focuses on the problem of viewpoint variations in a certain range of angles, rather than all possible angles. This is because when the cameras are re-arranged in a different environment, the viewpoint of each camera is usually changed but the change is often small.

B. Experiments on i3DPost Public Dataset

Next, the proposed method using multiple training viewpoints was compared with various other methods published from 2011 to 2018 including Holte et al. [36], Azary et al. [37], Iosifidis et al. [38], Castro et al. [39], Hilsenbeck et al. [40], Zang et al. [3], [41], and Angelini et al. [42], [43]. There are some state-of-the-art methods among them, such as Zang et al. [3] (2018) and Angelini et al. [42] (2018). While Zang et al. [3] learns transfer dictionary using a synthetic 3D and 2D video database constructed from human models to recognize actions from arbitrary viewpoints, the view-invariant AR method in Angelini et al. [42] adopted OpenPose [44], Long Short-Term Memory (LSTM) Neural Network [45], and 1D Convolutional Neural Network (1D CNN) [46], which are advanced and well-used technologies in recent years. Here, OpenPose is a realtime 2D pose estimator for multiple persons by providing 70 face landmarks, 18 body landmarks, 42 hands landmarks, and six feet landmarks for each target.

In these experiments, a well-known public dataset named *i3DPost* [30] was used for the comparisons. This dataset consists of eight actors with different body sizes, clothing, nationalities, and sex (i.e., six males and two females), performed each of 10 action classes once. In 10 classes, there are six single actions (i.e., *walk*, *run*, *jump-forward*, *bend*, *one-hand-wave*, and *jump-in-place*) and four combined actions (i.e., *sit-down-stand-up*, *run-fall*, *walk-sit-down*, and *run-jump-walk*). Eight calibrated and synchronized cameras corresponding to eight viewpoints were utilized to capture action videos in a full high definition resolution of 1920x1080 pixels. However, all videos in the dataset were symmetrically cropped and then resized to have a resolution of 640x480 pixels. This is because such a high resolution is unnecessary in this work and a smaller resolution will reduce the processing time. Moreover, although the dataset also includes actor 3D mesh models and camera calibration parameters, they were not used. The major challenges here are that each actor performed actions in arbitrary directions (see two example videos of the same action class in *View 0* of Fig. 5), action classes are quite similar (e.g., *walk*, *run*, and *jump-forward*), and there exist both single actions and their combined ones (e.g., *run-jump-walk*). Several frame examples for 10 action classes, eight actors, and eight viewpoints in the dataset are shown in Fig. 12.

The *i3DPost* dataset is usually tested with leave-one-actor-out (LOAO) strategy, therefore this strategy was adopted in these experiments. In other words, the video data of one actor in all available viewpoints was used for testing, while the remaining data in the dataset were for training. The AR accuracy for every testing actor was then averaged to obtain the final accuracy as shown in Table VII. Because some methods tested with a subset of six single action classes, while others performed experiments on 10 action classes dataset, the proposed method was evaluated on both subsets. Other parameter settings were the same as in Sect. IV-A, excepted the numbers of k-means clusters K utilized for each method in both subsets were vary from 600 to 5000 (see Table VI), and the threshold ϵ_s for *dimension selection* in Sect. III-E was $1.6 * 10^{-3}$.

Table VII shows comparative results of the proposed method and other works in two middle columns. It is observed that the

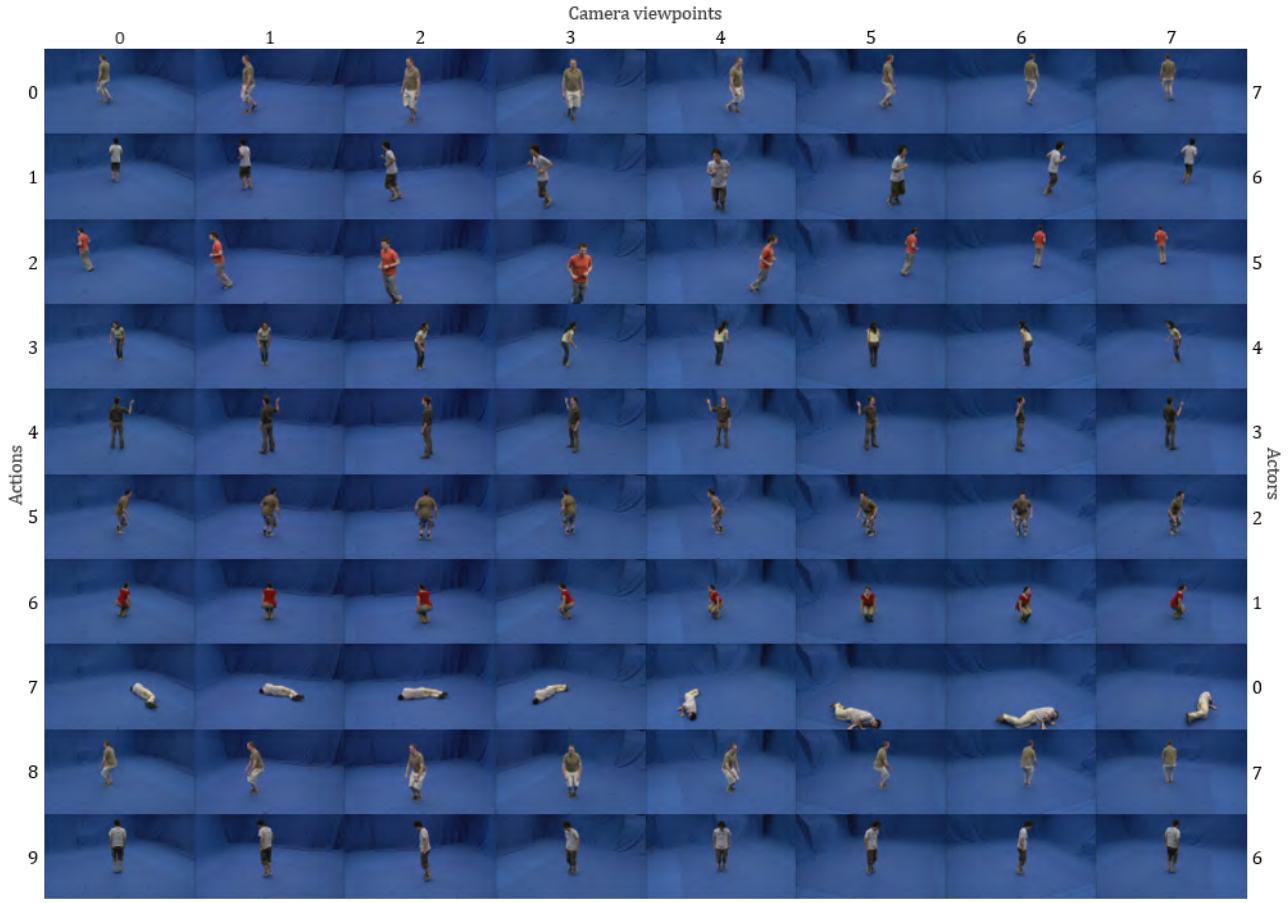

Fig. 12: Frame examples for 10 action classes, eight actors, and eight viewpoints in *i3DPost* dataset. Actions from top to down rows: *walk*, *run*, *jump-forward*, *bend*, *one-hand-wave*, *jump-in-place*, *sit-down-stand-up*, *run-fall*, *walk-sit-down*, and *run-jump-walk*.

TABLE VII: Methods, their recognition accuracy on *i3DPost* dataset (two middle columns), and other comparisons between them (two right columns).

Method	Published year	Subset of 6 actions (%)	Subset of 10 actions (%)	Not required multiple training viewpoints	Separately measured variations impact
Holte et al. [36]	2011	89.58	80.00	✗	✗
Azary et al. [37]	2012	92.97	86.72	✗	✗
Iosifidis et al. [38]	2012	98.16	-	-	✗
Castro et al. [39]	2015	99.00	-	-	✗
Hilsenbeck et al. [40]	2016	92.42	-	✗	✗
Zang et al. [41]	2016	-	93.75	✗	✗
Zang et al. [3]	2018	-	**94.60**	✗	✗
Angelini et al. [42]	2018	98.95	-	-	✗
Angelini et al. [43]	2018	**99.74**	-	-	✗
NONE		64.84	47.97	✓	✓
DS		89.58	82.19	✓	✓
AT		96.61	91.87	✓	✓
BOTH		**99.74**	**96.72**	✓	✓

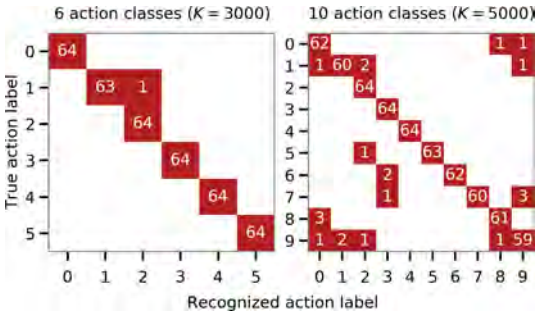

Fig. 13: Confusion matrices for the proposed method when the number of action classes were six and 10 in *i3DPost* dataset.

BOTH method with multiple training viewpoints yielded better results for both subsets, especially for the 10 classes dataset. More specifically, with the subset of six classes, the accuracy 99.74% of the proposed method is equal to that of Angelini et al. [43] and higher than that of Angelini et al. [42] (i.e., 98.95%) which adopted OpenPose, LSTM, and 1D CNN. In addition, these methods did not evaluate on the 10 classes dataset, did not test with singular training viewpoint, as well as did not independently measure variations impact as in Sect. IV-A (see last two columns in Table VII). On the other hand, the proposed method achieved better accuracy (i.e., 96.72%) than Zang et al. [3] with 94.60% for the 10 classes dataset. Moreover, the method in Zang et al. [3] requires video data or human models in multiple viewpoints for training, then it might be impossible with singular training viewpoint, while the proposed method just needs data in one viewpoint for training.

Fig. 13 shows confusion matrices of the *BOTH* for both subsets. It is observed from the matrix on the left that the proposed method mis-recognized only one video and correctly recognized other 383 videos. It confused *jump-forward* (2) with *run* (1) which are actually difficult to be distinguished (see second and third rows in Fig. 12). Furthermore, It is indicated from the matrix on the right that, the method sometimes confused *walk* (0) and *jump-forward* (2) with *run* (1) because they are three quite similar actions; or *jump-forward* (2) with *jump-in-place* (5) because global spatial features are not considered in this work; or single actions (e.g., *walk* (0), *run* (1), and *jump-forward* (2)) with their combined actions (e.g., *run-jump-walk* (8)).

C. Experiments on MuHAVi Public Dataset

Finally, the proposed method using multiple training viewpoints was also compared with other methods published from 2013 to 2019 using another well-known public dataset named *MuHAVi* [31]. The methods compared include Chaaraoui et al. [5], Orrite et al. [47], Murtaza et al. [31], and Nida et al. [48]. Among them, Nida et al. [48] adopted Convolutional Neural Network (CNN) which is advanced and well-used technologies in recent years for spatio-temporal feature learning.

MuHAVi-Uncut version of the *MuHAVi* dataset was used for the comparisons in these experiments. This is because this version contains much longer sequences also involving more actors and more camera viewpoints. In *MuHAVi* dataset, eight cameras were used to record 17 actions performed by seven actors. Each actor repeated an action three or four times. The 17 actions include *walk-turn-back*, *run-stop*, *pull-heavy-object*, *pickup-throw-object*, *punch*, *kick*, *shot-gun-collapse*, *walk-fall*, *look-in-car*, *crawl-on-knees*, *wave-arms*, *draw-graffiti*, *jump-over-fence*, *drunk-walk*, *smash-object*, *jump-over-gap*, and *climb-ladder*. Action videos have a resolution of 720x576 or 704x576 pixels. However, all videos in the dataset were symmetrically cropped and then resized to have a resolution of 640x480 pixels. This dataset is particularly challenging because of salt-pepper noise, shadows, and large number of action classes. Some frame examples for eight viewpoints in the dataset are shown in Fig. 14.

Leave-one-camera-out (LOCO) and leave-one-actor-out (LOAO) strategies were adopted for these experiments. In the LOCO strategy, action videos from one viewpoint are used for testing while the remaining videos are used for training. Similarly, in the LOCO strategy, the video data of one actor in all available viewpoints was used for testing, while the remaining data in the dataset were for training. The AR accuracy for every testing camera (i.e., viewpoint) or actor was then averaged to obtain the final accuracy as shown in Table VIII. Parameter settings were the same as in Sect. IV-A, excepted the number of k-means clusters K for both strategies was 5000 (see Table VI), and the threshold ϵ_s for *dimension selection* in Sect. III-E was $2 * 10^{-3}$. Parameter setting in Algorithm 1 for this dataset was different from it in the *i3DPost* dataset as shown in Table I.

Table VIII shows comparative results of the proposed method and other works in two middle columns. It is observed that although the *BOTH* method with multiple training viewpoints yielded a slightly lower accuracy for LOAO strategy comparing with Nida et al. [48], the proposed method was much more accurate than others for LOCO strategy. As mentioned in Sect. I, the camera viewpoint often changes when the action zone changes. In such a case, acquiring training data for new viewpoint (e.g., LOAO strategy) is a time-consuming task, except for online training methods. Therefore, a method that is more accurate for LOCO strategy would be more realistic and usable. In addition, these methods did not test with singular training viewpoint, as well as did not independently measure variations impact as in Sect. IV-A (see last two columns in Table VIII).

D. Parallel Processing

More affined viewpoints will need more computer resources to process the data. Each process among one of the N_f viewpoints is independent from each other in almost all steps (except for the last *action recognition using Euclidean distance*). Therefore, they can be implemented in parallel. As a result, the processing time is not a problem for the proposed method.

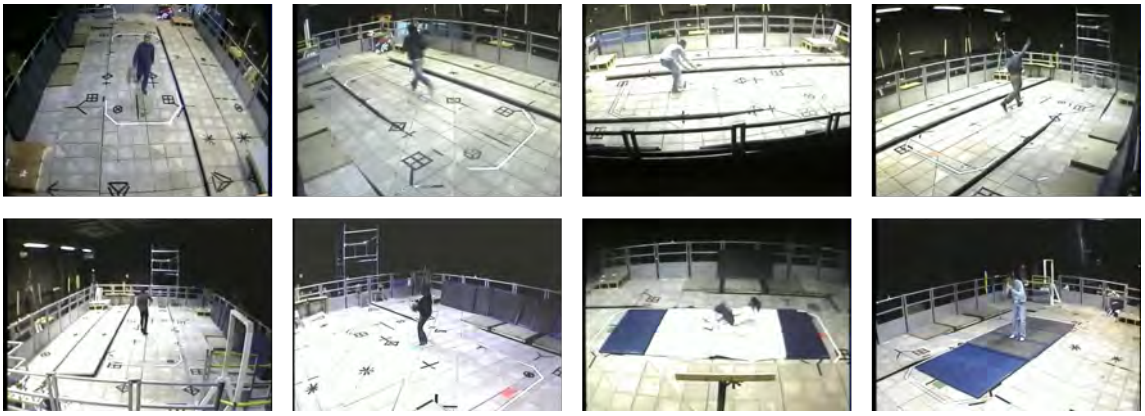

Fig. 14: Frame examples for eight viewpoints in *MuHAVi* dataset.

TABLE VIII: Methods, their recognition accuracy on *MuHAVI* dataset (two middle columns), and other comparisons between them (two right columns).

Method	Published year	LOCO (%)	LOAO (%)	Not required multiple training viewpoints	Separately measured variations impact
Chaaraoui et al. [5]	2013	50.40	81.50	-	✗
Orrite et al. [47]	2014	-	83.9	-	✗
Murtaza et al. [31]	2016	52.20	84.10	-	✗
Nida et al. [48]	2019	**82.04**	**93.66**	-	✗
BOTH		**85.16**	**92.94**	✓	✓

V. Conclusion

In this paper, a viewpoint-unconstrained method for the AR was presented. Our contribution was three-fold. First, a multi-scale space-sampling algorithm based on the affine transform was proposed to prevent a significant drop in the accuracy of recognition. Second, a dimension selection algorithm that determines the dimensions in the feature vector space for use in the recognition algorithm - improving the accuracy, was also proposed. Finally, a voting algorithm was introduced to adapt the situation of multiple training viewpoints. Multiple experiments on the *MVAR-Unity3D Attack* (a simulated dataset using the Unity3D software) were performed to individually evaluate the effect on each type of variations. The results indicated that the method is accurate and robust to the variations in subject scale, subject position, action speed, partial occlusion, and background. Other experiments on the recorded realistic *MVAR-GoPro Standing Exercise* dataset also verified that the proposed method is capable of maintaining its accuracy under a wide range of viewpoint changes. The invariant range was about 65 *deg* in the simulated dataset, and 45 *deg* in the realistic case. Additional experiments were performed on well-known *i3DPost* and *MuHAVi* public datasets to verify the superiority of the proposed method compared to state-of-the-art view-invariant action recognition methods. For future work, it is planned to enhance the method robustness to variations in lighting condition, consider global spatio-temporal features, and increase the ability to distinguish between single and combined actions. In addition, the affined HOF and dimension selection methods will be integrated into surgical workflow analysis systems proposed in [2].

Acknowledgment

This research is supported by Otsuka Toshimi Scholarship Foundation (2017-2019), and JSPS KAKENHI Grant Number 17K00372 (2017-2020).

References

[1] J.-H. Lee, "Human centered ubiquitous display in intelligent space," *The 33rd Annual Conference of the IEEE Industrial Electronics Society (IECON)*, pp. 22–27, 2007.

[2] D. T. Tran, R. Sakurai, H. Yamazoe, and J.-H. Lee, "Phase segmentation methods for an automatic surgical workflow analysis," *International Journal of Biomedical Imaging*, vol. 2017, 2017.

[3] J. Zhang, H. P. H. Shum, J. Han, and L. Shao, "Action recognition from arbitrary views using transferable dictionary learning," *IEEE Transactions on Image Processing*, vol. 27, no. 10, pp. 4709–4723, Oct 2018.

[4] J. Zheng, Z. Jiang, and R. Chellappa, "Cross-view action recognition via transferable dictionary learning," *IEEE Transactions on Image Processing*, vol. 25, no. 6, pp. 2542–2556, June 2016.

[5] A. A. Chaaraoui, P. Climent-Pérez, and F. Flórez-Revuelta, "Silhouette-based human action recognition using sequences of key poses," *Pattern Recognition Letters*, vol. 34, no. 15, pp. 1799–1807, 2013.

[6] W. Yang, Y. Gao, Y. Shi, and L. Cao, "Mrm-lasso: A sparse multiview feature selection method via low-rank analysis," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 26, no. 11, pp. 2801–2815, 2015.

[7] J. Liu, G. Wang, L. Duan, K. Abdiyeva, and A. C. Kot, "Skeleton-based human action recognition with global context-aware attention lstm networks," *IEEE Transactions on Image Processing*, vol. 27, no. 4, pp. 1586–1599, April 2018.

[8] C. Zhang, H. Zheng, and J. Lai, "Cross-view action recognition based on hierarchical view-shared dictionary learning," *IEEE Access*, vol. 6, pp. 16 855–16 868, 2018.

[9] N. Jia, V. Sanchez, and C. Li, "On view-invariant gait recognition: a feature selection solution," *IET Biometrics*, vol. 7, no. 4, pp. 287–295, 2018.

[10] X. Ben, P. Zhang, Z. Lai, R. Yan, X. Zhai, and W. Meng, "A general tensor representation framework for cross-view gait recognition," *Pattern Recognition*, vol. 90, pp. 87 – 98, 2019.

[11] X. You, J. Xu, W. Yuan, X.-Y. Jing, D. Tao, and T. Zhang, "Multi-view common component discriminant analysis for cross-view classification," *Pattern Recognition*, 2019.

[12] Y. Kong, Z. Ding, J. Li, and Y. Fu, "Deeply learned view-invariant features for cross-view action recognition," *IEEE Transactions on Image Processing*, vol. 26, no. 6, pp. 3028–3037, 2017.

[13] P. Zhang, C. Lan, J. Xing, W. Zeng, J. Xue, and N. Zheng, "View adaptive neural networks for high performance skeleton-based human action recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 1–1, 2019.

[14] H. Rahmani, A. Mian, and M. Shah, "Learning a deep model for human action recognition from novel viewpoints," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 3, pp. 667–681, March 2018.

[15] A. Kumar, G. Gupta, A. Sharma, and K. M. Krishna, "Towards view-invariant intersection recognition from videos using deep network ensembles," *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1053–1060, Oct 2018.

[16] D. Thapar, A. Nigam, D. Aggarwal, and P. Agarwal, "Vgr-net: A view invariant gait recognition network," *2018 IEEE 4th International Conference on Identity, Security, and Behavior Analysis (ISBA)*, pp. 1–8, Jan 2018.

[17] H. Zhan, B. Shi, L.-Y. Duan, and A. C. Kot, "Deepshoe: An improved multi-task view-invariant cnn for street-to-shop shoe retrieval," *Computer Vision and Image Understanding*, vol. 180, pp. 23 – 33, 2019.

[18] Y. Wang, C. Song, Y. Huang, Z. Wang, and L. Wang, "Learning view invariant gait features with two-stream gan," *Neurocomputing*, vol. 339, pp. 245 – 254, 2019.

[19] H. Chen, Z. Liu, C. Tanougast, and J. Ding, "Optical hyperspectral image cryptosystem based on affine transform and fractional fourier transform," *Applied Sciences*, vol. 9, no. 2, 2019.

[20] A. Gardezi, U. Malik, S. Rehman, R. C. D. Young, P. M. Birch, and C. R. Chatwin, "Enhanced target recognition employing spatial correlation filters and affine scale invariant feature transform," in *Pattern Recognition and Tracking XXX*, vol. 10995, 2019.

[21] N. Xue, G. Xia, X. Bai, L. Zhang, and W. Shen, "Anisotropic-scale junction detection and matching for indoor images," *IEEE Transactions on Image Processing*, vol. 27, no. 1, pp. 78–91, Jan 2018.

[22] H. Xia, T. Li, W. Liu, X. Zhong, and J. Yuan, "Abnormal event detection method in surveillance video based on temporal cnn and sparse optical flow," in *Proceedings of the 2019 5th International Conference on Computing and Data Engineering*, ser. ICCDE' 19. New York, NY, USA: ACM, 2019, pp. 90–94.

[23] S. L. Happy and A. Routray, "Fuzzy histogram of optical flow orientations for micro-expression recognition," *IEEE Transactions on Affective Computing*, pp. –, 2017.

[24] R. V. H. M. Colque, C. Caetano, M. T. L. de Andrade, and W. R. Schwartz, "Histograms of optical flow orientation and magnitude and entropy to detect anomalous events in videos," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 27, no. 3, pp. 673–682, March 2017.

[25] L. d. l. Fuente-Tomas, B. Arranz, G. Safont, P. Sierra, M. Sanchez-Autet, A. Garcia-Blanco, and M. P. Garcia-Portilla, "Classification of patients with bipolar disorder using k-means clustering," *PLOS ONE*, vol. 14, no. 1, pp. 1–15, 01 2019. [Online]. Available: https://doi.org/10.1371/journal.pone.0210314

[26] A. K. Dubey, U. Gupta, and S. Jain, "Comparative study of k-means and fuzzy c-means algorithms on the breast cancer data," *International Journal on Advanced Science, Engineering and Information Technology*, vol. 8, no. 1, pp. 18–29, 2018.

[27] S. Kant, T. Mahara, V. K. Jain, D. K. Jain, and A. K. Sangaiah, "Leaderrank based k-means clustering initialization method for collaborative filtering," *Computers and Electrical Engineering*, vol. 69, pp. 598–609, 2018.

[28] G. Farnebäck, "Two-frame motion estimation based on polynomial expansion," *Image Analysis*, pp. 363–370, 2003.

[29] H. Abdi and L. J. Williams, "Principal component analysis," *WIREs Comput. Stat.*, vol. 2, no. 4, pp. 433–459, July 2010.

[30] N. Gkalelis, H. Kim, A. Hilton, N. Nikolaidis, and I. Pitas, "The i3dpost multi-view and 3d human action/interaction database," *2009 Conference for Visual Media Production*, pp. 159–168, Nov 2009.

[31] F. Murtaza, M. H. Yousaf, and S. A. Velastin, "Multi-view human action recognition using 2d motion templates based on mhis and their hog description," *IET Computer Vision*, pp. 758–767, 2016.

[32] S. N. Punch, *Human Males Pack*. Unity Asset Store, 2017.

[33] Niandrei, *Lake Race Track*. Unity Asset Store, 2018.

[34] Explosive, *RPG Character Mecanim Animation Pack Free*. Unity Asset Store, 2019.

[35] RockVR, *Video Capture*. Unity Asset Store, 2017.

[36] M. B. Holte, T. B. Moeslund, N. Nikolaidis, and I. Pitas, "3d human action recognition for multi-view camera systems," *2011 International Conference on 3D Imaging, Modeling, Processing, Visualization and Transmission*, pp. 342–349, May 2011.

[37] S. Azary and A. Savakis, "Multi-view action classification using sparse representations on motion history images," *2012 Western New York Image Processing Workshop*, pp. 5–8, Nov 2012.

[38] A. Iosifidis, A. Tefas, and I. Pitas, "Multi-view action recognition based on action volumes, fuzzy distances and cluster discriminant analysis," *Signal Processing*, vol. 93, no. 6, pp. 1445 – 1457, 2013, special issue on Machine Learning in Intelligent Image Processing.

[39] G. Castro-Muñoz and J. Martínez-Carballido, "Real time human action recognition using full and ultra high definition video," *2015 International Conference on Computational Science and Computational Intelligence (CSCI)*, pp. 509–514, Dec 2015.

[40] B. Hilsenbeck, D. Münch, H. Kieritz, W. Hübner, and M. Arens, "Hierarchical hough forests for view-independent action recognition," *2016 23rd International Conference on Pattern Recognition (ICPR)*, pp. 1911–1916, Dec 2016.

[41] Jingtian Zhang, Lining Zhang, H. P. H. Shum, and Ling Shao, "Arbitrary view action recognition via transfer dictionary learning on synthetic training data," *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1678–1684, May 2016.

[42] F. Angelini, Z. Fu, Y. Long, L. Shao, and S. M. Naqvi, "Actionxpose: A novel 2d multi-view pose-based algorithm for real-time human action recognition," *CoRR*, vol. abs/1810.12126, 2018. [Online]. Available: http://arxiv.org/abs/1810.12126

[43] F. Angelini, Z. Fu, S. A. Velastin, J. A. Chambers, and S. M. Naqvi, "3d-hog embedding frameworks for single and multi-viewpoints action recognition based on human silhouettes," *IEEE SigPort*, 2018.

[44] Z. Cao, T. Simon, S.-E. Wei, and Y. Sheikh, "Realtime multi-person 2d pose estimation using part affinity fields," *Computer Vision and Pattern Recognition (CVPR)*, pp. 7291–7299, 2017.

[45] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[46] K. Simonyan and A. Zisserman, "Two-stream convolutional networks for action recognition in videos," *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 1*, pp. 568–576, 2014.

[47] C. Orrite, M. Rodriguez, E. Herrero, G. Rogez, and S. A. Velastin, "Automatic segmentation and recognition of human actions in monocular sequences," *22nd International Conference on Pattern Recognition (ICPR)*, pp. 4218–4223, 2014.

[48] N. Nida, M. H. Yousaf, A. Irtaza, and S. Velastin, "Instructor activity recognition through deep spatiotemporal features and feedforward extreme learning machines," *Mathematical Problems in Engineering*, vol. 2019, pp. 1–13, 04 2019.