J Real-Time Image Proc
DOI 10.1007/s11554-013-0370-1

ORIGINAL RESEARCH PAPER

Real-time human action recognition based on depth motion maps

Chen Chen · Kui Liu · Nasser Kehtarnavaz

Received: 8 April 2013 / Accepted: 25 July 2013
© Springer-Verlag Berlin Heidelberg 2013

Abstract This paper presents a human action recognition method by using depth motion maps (DMMs). Each depth frame in a depth video sequence is projected onto three orthogonal Cartesian planes. Under each projection view, the absolute difference between two consecutive projected maps is accumulated through an entire depth video sequence forming a DMM. An l_2-regularized collaborative representation classifier with a distance-weighted Tikhonov matrix is then employed for action recognition. The developed method is shown to be computationally efficient allowing it to run in real-time. The recognition results applied to the Microsoft Research Action3D dataset indicate superior performance of our method over the existing methods.

Keywords Human action recognition · Depth motion map · RGBD camera · Collaborative representation classifier

1 Introduction

Human action recognition is an active research area in computer vision. Earlier attempts at action recognition have involved using video sequences captured by video cameras. Spatio-temporal features are widely used for recognizing human actions, e.g. [1–6]. As imaging technology advances, it has become possible to capture depth information in real-time. Compared with conventional images, depth maps are insensitive to changes in lighting conditions and can provide 3D information toward distinguishing actions that are difficult to characterize using conventional images. Figure 1 shows two examples consisting of nine depth maps of the action *Golf swing* and the action *Forward kick*. Since the release of low cost depth sensors, in particular Microsoft Kinect and ASUS Xtion, many research works have been carried out on human action recognition using depth imagery, e.g. [7–13]. As noted in [14], 3D joint positions of a person's skeleton estimated from depth images provide additional information to achieve action recognition.

In this paper, the problem of human action recognition from depth map sequences is examined from the perspective of computational efficiency. These images are captured by an RGBD camera. Specifically, the depth motion maps (DMMs) generated by accumulating motion energy of projected depth maps in three projective views (front view, side view, and top view) are used as feature descriptors. Compared with 3D depth maps, DMMs are 2D images that provide an encoding of motion characteristics of an action. Motivated by the success of sparse representation in face recognition [15–18] and image classification [18, 19], an l_2-regularized collaborative representation classifier is utilized which seeks a match of an unknown sample via a linear combination of training samples from all the classes. The class label is then derived according to the class which best approximates the unknown sample. Basically, our introduced method involves a spatio-temporal motion representation based on DMMs followed by an l_2-regularized collaborative representation classifier with a distance-weighted Tikhonov matrix to perform computationally efficient action recognition.

The rest of the paper is organized as follows. In Sect. 2, related works are presented. In Sect. 3, the details of generating DMMs feature descriptors are stated. In Sect. 4, the sparse representation classifier (SRC) is first introduced and then the l_2-regularized collaborative representation classifier is described for performing action recognition.

C. Chen (✉) · K. Liu · N. Kehtarnavaz
The University of Texas at Dallas, Richardson, TX, USA
e-mail: chenchen870713@gmail.com

Published online: 11 August 2013

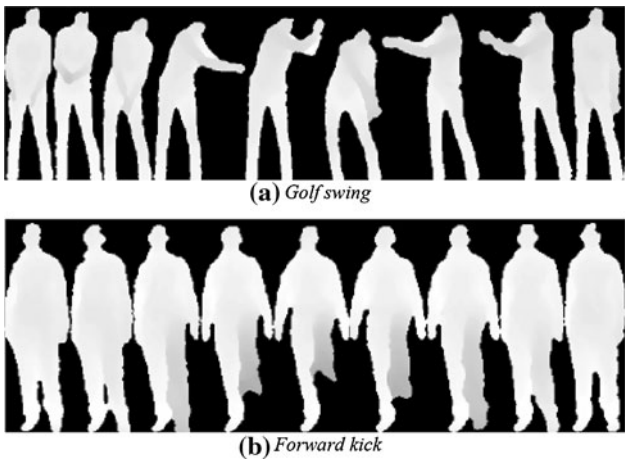

Fig. 1 Examples of depth map sequences for **a** *Golf swing* action, and **b** *Forward kick* action

The experimental results are reported in Sect. 5. Finally, in Sect. 6, concluding remarks are stated.

2 Related works

Space–time based methods such as space–time volumes, spatio-temporal features, and trajectories have been widely utilized for human action recognition from video sequences captured by traditional RGB cameras. In [1], spatio-temporal interest points coupled with an SVM classifier was used to achieve human action recognition. Cuboid descriptors were employed in [2] for action representation. In [3], SIFT-feature trajectories modeled in a hierarchy of three abstraction levels were used to recognize actions in video sequences. Various local motion features were gathered as spatio-temporal bag-of-features (BoF) in [4] to perform action classification. Motion-energy images (MEI) and motion-history images (MHI) were introduced in [5] as motion templates to model spatial and temporal characteristics of human actions in videos. In [6], a hierarchical extension for computing dense motion flow from MHI was presented. A major shortcoming associated with using these intensity-based or color-based methods is the sensitivity of recognition to illumination variations, limiting the recognition robustness.

With the release of RGBD sensors, research into action recognition based on depth information has grown. Skeleton-based approaches utilize locations of skeletal joints extracted from depth images. In [7], a view invariant posture representation was devised using histograms of 3D joint locations (HOJ3D) within a modified spherical coordinate system. HOJ3D were re-projected using LDA and clustered into k posture visual words. The temporal evolutions of these visual words were modeled by a discrete hidden Markov model. In [8], a Naive-Bayes-Nearest-Neighbor (NBNN) classifier was employed to recognize human actions based on Eigen Joints (i.e., position differences of joints) combining static posture, motion, and offset information. Such skeleton-based approaches have limitations due to inaccuracies in skeletal estimation. Moreover, the skeleton information is not always available in many applications.

There are methods that involve extracting spatio-temporal features from the entire set of points in a depth map sequence to distinguish different actions. An action graph was employed in [9] to model the dynamics of actions and a collection of 3D points were used to characterize postures. However, the 3D points sampling scheme used generated a large amount of data leading to a computationally expensive training step. In [10], a DMM-based histogram of oriented gradients (HOG) was utilized to compactly represent the body shape and movement information toward distinguishing actions. In [11], random occupancy pattern (ROP) features were extracted from depth images using a weighted sampling scheme. A sparse coding approach was utilized to robustly encode ROP features for action recognition and the features were shown to be robust to occlusion. In [12], 4D space–time occupancy patterns were used as features which preserved spatial and temporal contextual information coping with intra-class variations. A simple classifier based on the cosine distance was then used for action recognition.

In [13], a hybrid solution combining skeleton and depth information was used for action recognition. 3D joint position and local occupancy patterns were used as features. An actionlet ensemble model was then learnt to represent each action and to capture intra-class variations.

In general, the above references do not elaborate on the computational complexity aspect of their solutions and do not provide actual real-time processing times. In contrast to the existing methods, in this work, both the computational complexity and the processing times associated with each component of our method are reported.

3 Depth motion maps as features

A depth map can be used to capture the 3D structure and shape information. Yang et al. [10] proposed to project depth frames onto three orthogonal Cartesian planes for the purpose of characterizing the motion of an action. Due to its computational simplicity, the same approach in [10] is adopted in this work while the procedure to obtain DMMs is modified. More specifically, each 3D depth frame is used to generate three 2D projected maps corresponding to front, side, and top views, denoted by map_v where $v \in \{f, s, t\}$. For a point (x, y, z) in a depth frame with z denoting the depth value in a right-handed coordinate system, the pixel value in three projected maps is indicated by z, x, and y,

respectively. Different from [10], for each projected map, the motion energy is calculated here as the absolute difference between two consecutive maps without thresholding. For a depth video sequence with N frames, DMM_v is obtained by stacking the motion energy across an entire depth video sequence as follows:

$$\mathrm{DMM}_v = \sum_{i=a}^{b} \left| \mathrm{map}_v^i - \mathrm{map}_v^{i-1} \right|, \tag{1}$$

where i represents the frame index; map_v^i is the projected map of ith frame under projection view v; $a \in \{2, \ldots, N\}$ and $b \in \{2, \ldots, N\}$ denote the starting frame and the end frame index. It should be noted that not all the frames in a depth video sequence are used to generate DMMs. This point is discussed further in the experimental setup section. A bounding box is then set to extract the non-zero region as the foreground in each DMM.

Let the foreground extracted DMM be denoted by DMM_v hereafter. Two examples of DMM_v generated from the *Tennis serve* and *Forward kick* video sequences are shown in Fig. 2. DMMs from the three projection views effectively capture the characteristics of the motion in a distinguishable way. That is the reason here for using DMMs as feature descriptors for action recognition. Since DMM_v of different action video sequences may have different sizes, bicubic interpolation is used to resize all DMM_v under the same projection view to a fixed size in order to reduce the intra-class variability, for example due to different subject heights. The size of DMM_f is $m_f \times n_f$, the size of DMM_s is $m_s \times n_s$, and the size of DMM_t is $m_t \times n_t$. Since pixel values are used as features, they are normalized between 0 and 1 to avoid large pixel values dominating the feature set. The resized and normalized DMM is denoted by $\overline{\mathrm{DMM}}_v$. For an action video sequence with three DMMs, a feature vector of size $\left(m_f \times n_f + m_s \times n_s + m_t \times n_t\right) \times 1$ is thus formed to be $\mathbf{h} = \left[\mathrm{vec}\left(\overline{\mathrm{DMM}}_f\right), \mathrm{vec}\left(\overline{\mathrm{DMM}}_s\right), \mathrm{vec}\left(\overline{\mathrm{DMM}}_t\right)\right]^T$ by concatenating the three vectorized DMMs; $\mathrm{vec}(\cdot)$ indicates the vectorization operator and T the matrix transpose. The feature vector encodes the 4D characteristics of an action video sequence. Note that the HOG descriptors of the DMMs are not computed here as done in [10] and image resizing is applied to DMMs but not to each projected depth map as done in [10]. As a result, the computational complexity of the feature extraction process is greatly reduced.

4 l_2-regularized collaborative representation classifier

Sparse representation (or sparse coding) has been an active research area in the machine learning community due to its success in face recognition [15–18] and image classification [18, 19]. The central idea of the SRC is to represent a test sample according to a small number of atoms sparsely chosen out of an over-complete dictionary formed by all the available training samples. Consider a dataset with C classes of training samples arranged column-wise $\mathbf{A} = [\mathbf{A}_1, \mathbf{A}_2, \ldots, \mathbf{A}_C] \in \mathbb{R}^{d \times n}$, where $\mathbf{A}_j (j = 1, \ldots, C)$ is the subset of the training samples associated with class j, d is the dimension of training samples and n is the total number of training samples from all the classes. A test sample $\mathbf{g} \in \mathbb{R}^d$ can be represented as a sparse linear combination of the training samples, which can be formulated as,

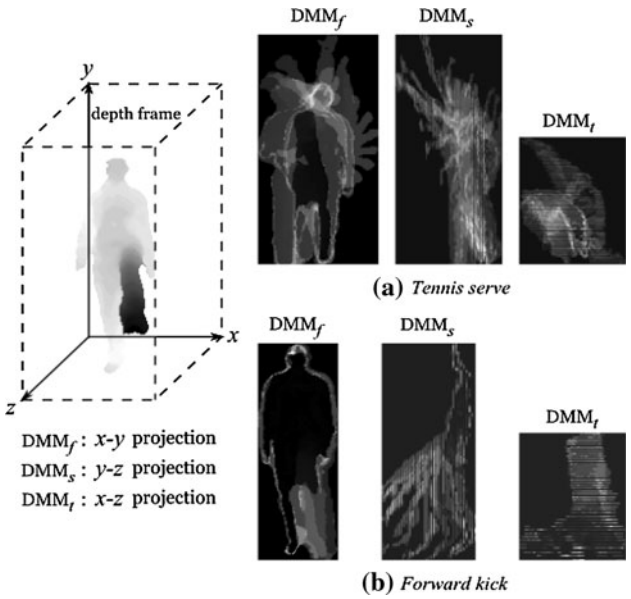

Fig. 2 DMM_v generated from **a** *Tennis serve*, and **b** *Forward kick* depth action video sequences

$$\mathbf{g} = \mathbf{A}\boldsymbol{\alpha}, \tag{2}$$

where $\boldsymbol{\alpha} = [\boldsymbol{\alpha}_1, \boldsymbol{\alpha}_2, \ldots, \boldsymbol{\alpha}_C]$ is an $n \times 1$ vector of coefficients corresponding to all the training samples and $\boldsymbol{\alpha}_j (j = 1, \ldots, C)$ denotes the subset of the coefficients associated with the training samples from the jth class, i.e. $\mathbf{A}_j$. From a practical standpoint, one cannot directly solve for $\boldsymbol{\alpha}$ since (2) is typically under-determined [17]. To reach a solution, one can solve the following l_1 norm minimization problem,

$$\hat{\boldsymbol{\alpha}} = \arg\min_{\boldsymbol{\alpha}} \left\{ \|\mathbf{g} - \mathbf{A}\boldsymbol{\alpha}\|_2^2 + \theta \|\boldsymbol{\alpha}\|_1 \right\}, \tag{3}$$

where θ is a scalar regularization parameter which balances the influence of the residual and the sparsity term. The class label of $\mathbf{g}$ is then obtained via,

$$\mathrm{class}(\mathbf{g}) = \arg\min_j \{e_j\} \tag{4}$$

where $e_j = \left\|\mathbf{g} - \mathbf{A}_j \hat{\boldsymbol{\alpha}}_j\right\|_2$. The reader is referred to [15] for more details.

As described in [20], it is the collaborative representation, i.e. use of all the training samples as a dictionary, but not the

l_1-norm sparsity constraint, that improves the classification accuracy. The l_2-regularized approach generates comparable results but with significantly lower computational complexity [20, 21]. Therefore, here the l_2-regularized approach is used for action recognition. As mentioned in Sect. 3, each depth video sequence generates a feature vector $\mathbf{h} \in \mathbb{R}^{m_f \times n_f + m_s \times n_s + m_t \times n_t}$, therefore the dictionary is $\mathbf{A} = [\mathbf{h}_1, \mathbf{h}_2, \ldots, \mathbf{h}_K]$ with K being the total number of available training samples from all the action classes. Let $\mathbf{y}_q \in \mathbb{R}^{m_f \times n_f + m_s \times n_s + m_t \times n_t}$ denote the feature vector of an unknown action sample. Tikhonov regularization [22] is employed here to calculate the coefficient vector according to,

$$\hat{\boldsymbol{\alpha}} = \arg\min_{\boldsymbol{\alpha}} \left\{ \|\mathbf{y}_q - \mathbf{A}\boldsymbol{\alpha}\|_2^2 + \lambda \|\mathbf{L}\boldsymbol{\alpha}\|_2^2 \right\}, \tag{5}$$

where $\mathbf{L}$ is the Tikhonov regularization matrix and λ is the regularization parameter. The term $\mathbf{L}$ allows the imposition of prior knowledge on the solution. Normally, $\mathbf{L}$ is chosen to be the identity matrix. The approach proposed in [23] is adopted here by giving less weight to the situations which are dissimilar from the unknown sample than those which are similar. Specifically, a diagonal matrix $\mathbf{L}$ in the following form is considered.

$$\mathbf{L} = \begin{bmatrix} \|\mathbf{y}_q - \mathbf{h}_1\|_2 & & 0 \\ & \ddots & \\ 0 & & \|\mathbf{y}_q - \mathbf{h}_K\|_2 \end{bmatrix}. \tag{6}$$

The coefficient vector $\hat{\boldsymbol{\alpha}}$ is calculated as follows [24]:

$$\hat{\boldsymbol{\alpha}} = \left(\mathbf{A}^T\mathbf{A} + \lambda \mathbf{L}^T\mathbf{L}\right)^{-1}\mathbf{A}^T\mathbf{y}_q. \tag{7}$$

The class label for each unknown sample is then found from (4). Algorithm 1 provides more details of the l_2-regularized collaborative representation classifier utilized

Algorithm 1 l_2 **− Regularized Collaborative Representation Classifier for Action Recognition**

Input: Training feature set $\mathbf{A} = \{\mathbf{h}_l\}_{l=1}^{K}$, class label ω_l (used for class partitioning), test sample $\mathbf{y}_q$, λ, C (number of action classes)

Calculate $\hat{\boldsymbol{\alpha}}$ according to (7)

for all $j \in \{1, \ldots, C\}$ **do**

Partition $\mathbf{A}_j$, $\hat{\boldsymbol{\alpha}}_j$

Calculate $e_j = \|\mathbf{y}_q - \mathbf{A}_j\hat{\boldsymbol{\alpha}}_j\|_2$

end for

Decide $class(\mathbf{y}_q)$ via (4)

Output: $class(\mathbf{y}_q)$

5 Experimental setup

In this section, it is explained how our method was applied to the public domain Microsoft Research (MSR) Action3D dataset [9] with the depth map sequences captured by an RGBD camera. Our method is then compared with the existing methods.

The MSR-Action3D dataset includes 20 actions performed by 10 subjects. Each subject performed each action 2 or 3 times. Each subject performed the same action differently. As a result, the dataset incorporated the intra-class variation. For example, the speed of performing an action varied with different subjects. The resolution of each depth map was 320 × 240. To facilitate a fair comparison, the same experimental settings as done in [7–10, 12] were considered. The actions were divided into three subsets as listed in Table 1. For each action subset, three different tests were performed. In Test One, 1/3 of the samples were used as training samples and the rest as test samples; in Test Two, 2/3 of the samples were used as training samples and the rest as test samples; in Cross Subject Test (or Test Three), half of the subjects were used as training and the rest as test subjects. In the experimental setup reported in [9], in Test One (or Two), for each action and each subject, the first (or first two) action sequences were used for training; while in Cross Subject Test, subjects 1, 3, 5, 7, 9 (if existed) were used for training. Noting that the samples or subjects used for training and testing were fixed, they are referred to as Fixed Tests here.

Another experiment was conducted by randomly choosing training samples or training subjects corresponding to the three tests. In other words, the action sequences of each subject for each action were randomly chosen to serve as training samples in Test One and Test Two. For Cross Subject Test, half of the subjects were randomly chosen for training and the rest used for testing. These tests are referred to as Random Tests here.

For each depth video sequence, the first five frames and the last five frames were removed and the remaining frames were used to generate DMM_v. The purpose of this frame removal was two-fold. First, at the beginning and the end, the subjects were mostly at a stand-still position with only small body movements, which did not contribute to the motion characteristics of the video sequences. Second, in our process of generating DMMs, small movements at the beginning and the end resulted in a stand-still body shape with large pixel values along the edges which contributed to a large amount of reconstruction error. Therefore, the initial and end frame removal was done to remove no motion condition. Other frame selection methods may be used here to achieve the same.

To have three fixed sizes for DMM_v, the sizes of the DMMs of all the samples (training and test samples) were

Table 1 Three subsets of actions used for msr-action3D dataset

Action set 1 (AS1)	Action set 2 (AS2)	Action set 3 (AS3)
Horizontal wave (2)	High wave (1)	High throw (6)
Hammer (3)	Hand catch (4)	Forward kick (14)
Forward punch (5)	Draw x (7)	Side kick (15)
High throw (6)	Draw tick (8)	Jogging (16)
Hand clap (10)	Draw circle (9)	Tennis swing (17)
Bend (13)	Two hand wave (11)	Tennis serve (18)
Tennis serve (18)	Forward kick (14)	Golf swing (19)
Pickup throw (20)	Side boxing (12)	Pickup throw (20)

found under each projection view. The fixed size of each DMM was simply set to half of the mean value of all the sizes. For the training feature set and the test feature set, principal component analysis (PCA) was applied to reduce the dimensionality. The PCA transform matrix was calculated using the training feature set and then applied to the test feature set. This dimensionality reduction step provided computational efficiency for the classification. In our experiments, the largest 85 % of the eigenvalues were kept.

5.1 Parameter selection

In the l_2-regularized collaborative representation classifier, a key parameter is λ which controls the relative effect of the Tikhonov regularization term in the optimization stated in (5). Many approaches have been presented in the literature—such as the L-curve [25], discrepancy principle, and generalized cross-validation (GCV)—for finding an optimal value for this regularization parameter. To find an optimal λ, a set of values were examined. Figure 3 shows the recognition rates with different values of λ for Fixed Cross Subject Test. Random Cross Subject Test was also performed with the same set of values. For each value of λ, the testing was repeated 50 times. The average recognition rates are shown in Fig. 4. From Figs. 3 and 4, one can see that the recognition accuracy was quite stable for a large range of λ values. As a result, in all the experiments reported here, the value of $\lambda = 0.001$ was thus chosen.

5.2 Rejection option

An option was added to reject an action which did not belong to an action set. For example, since action *Jump* was not included in the MSR-Action3D dataset, it was rejected. This was done by setting a rejection threshold for the minimum reconstruction error calculated from (4). This threshold was set according to the degree of similarity of the action not included in the recognition set. Let $e_{\min}$ indicate the minimum reconstruction error, the

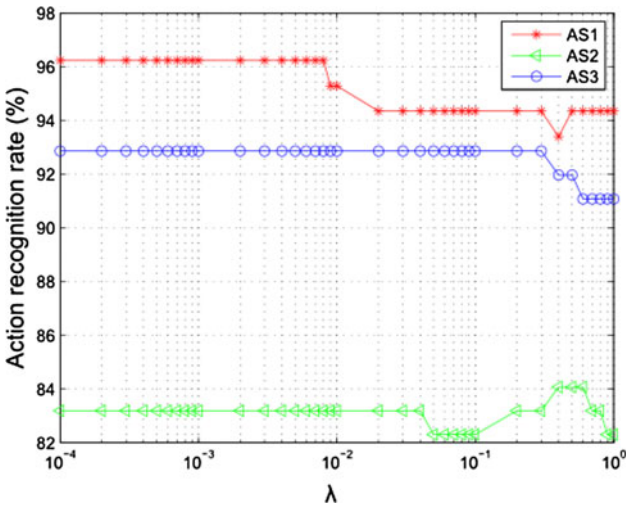

Fig. 3 Recognition rates of Fixed Cross Subject Test for various values of λ

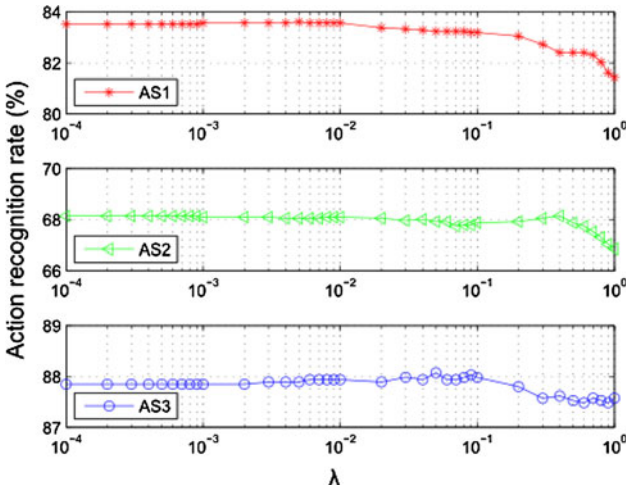

Fig. 4 Average recognition rates of Random Cross Subject Test for various values of λ

decision of rejecting or accepting an unknown action sample was made as follows:

$$\text{Decision (action)} = \begin{cases} \text{Reject}, & \text{if } e_{\min} > \text{threshold} \\ \text{Accept}, & \text{otherwise} \end{cases} \tag{8}$$

To find an appropriate rejection threshold, Random Tests on the MSR-Action3D dataset were done by repeating each test for each subset 200 times. Noting K_T to be the total number of test samples in a subset for a random test, 200 test trials generated $200 \times K_T$ minimum reconstruction errors thus forming a vector $\mathbf{E} = \left[e^1_{\min}, e^2_{\min}, \ldots, e^{200\times K_T}_{\min}\right]$. The mean of $\mathbf{E}$ was then calculated and used as the rejection threshold.

5.3 Results and discussion

5.3.1 Recognition results

Our method was compared with the existing methods using the MSR-Action3D dataset. The comparison results are reported in Table 2. The best recognition rate achieved is highlighted in bold. From Table 2, it can be seen that our method outperformed the method reported in [9] in all the test cases. For the challenging Cross Subject Test, our method produced 90.5 % recognition rate which was slightly lower than the method reported in [10]. However, it should be noted that our method did not require the calculation of HOG descriptors and thus it was computationally much more efficient. The confusion matrix of our method for Fixed Cross Subject Test is shown in Fig. 5. For a compact representation, numbers are used to indicate the actions listed in Table 1. There are three possible reasons for the misclassifications in the Cross Subject Test. First, large intra-class variations existed due to considerable variations in the same action performed by different subjects. Although the DMM_v of all the samples were normalized to have the same sizes, the normalization could not eliminate the intra-class variations entirely. Second, the feature formed by DMM_v did not exhibit enough discriminatory power to distinguish similar motions. For example, *Hammer* was confused with *Forward punch* and *High throw* was confused with *Tennis serve* since they had similar motion characteristics. In other words, the DMM_v generated by these actions were similar. Finally, since our classification decision was based on the reconstruction errors of different training classes in (4), the class with the smallest reconstruction error was favored. Hence, a misclassification occurred when two actions were similar and the wrong class had a smaller reconstruction error.

To verify that our method did not depend on specific training data, another experiment was done by randomly choosing training samples or training subjects for the three tests. Each test was run for each subset 200 times and the mean performance (mean accuracy $\pm$ standard deviation) was computed, see Table 3. For Test One and Test Two, the average recognition rate over the subsets was found comparable with the outcome shown in Table 2. In the Cross Subject Test, the average recognition rate dropped by about 10 % mainly due to a large intra-class variation. However, our method still achieved 80 % recognition rate overall which was higher than the rates reported in [7] and [9] using the Fixed Cross Subject Test.

Furthermore, l_1- regularized SRC (denoted by L1) and SVM [27] were considered in order to compare the recognition performance with our l_2-regularized collaborative representation classifier (denoted by L2). These three classifiers were tested on the same training and test samples of the Random Cross Subject Test with 200 trials. The SPAMS toolbox [26] was employed to solve the optimization problem in (3) due to its fast implementation. Radial basis function (RBF) kernel was used for the SVM and its two parameters (penalty parameter and kernel width) were tuned for optimal recognition rates. The average recognition rates using the three classifiers are shown in Fig. 6. As exhibited in this figure, our l_2-regularized collaborative representation classifier was on par with the SCR and consistently outperformed the SVM classifier in all the three subsets. A disadvantage of SVM was also the requirement to tune its two parameters.

Table 2 Recognition rates (%) comparison of Fixed Tests for msr-action3D dataset

	Li et al. [9]	Lu et al. [7]	Yang et al. [8]	Yang et al. [10]	Vieira et al. [12]	Our method
Test one						
AS1	89.5	**98.5**	94.7	97.3	98.2	97.3
AS2	89.0	**96.7**	95.4	92.2	94.8	96.1
AS3	96.3	93.5	97.3	98.0	97.4	**98.7**
Average	91.6	96.2	95.8	95.8	96.8	**97.4**
Test two						
AS1	93.4	98.6	97.3	98.7	**99.1**	98.6
AS2	92.9	97.2	**98.7**	94.7	97.0	**98.7**
AS3	96.3	94.9	97.3	98.7	98.7	**100**
Average	94.2	97.2	97.8	97.4	98.3	**99.1**
Cross subject test						
AS1	72.9	88.0	74.5	**96.2**	84.7	**96.2**
AS2	71.9	**85.5**	76.1	84.1	81.3	83.2
AS3	79.2	63.6	**96.4**	94.6	88.4	92.0
Average	74.7	79.0	82.3	**91.6**	84.8	90.5

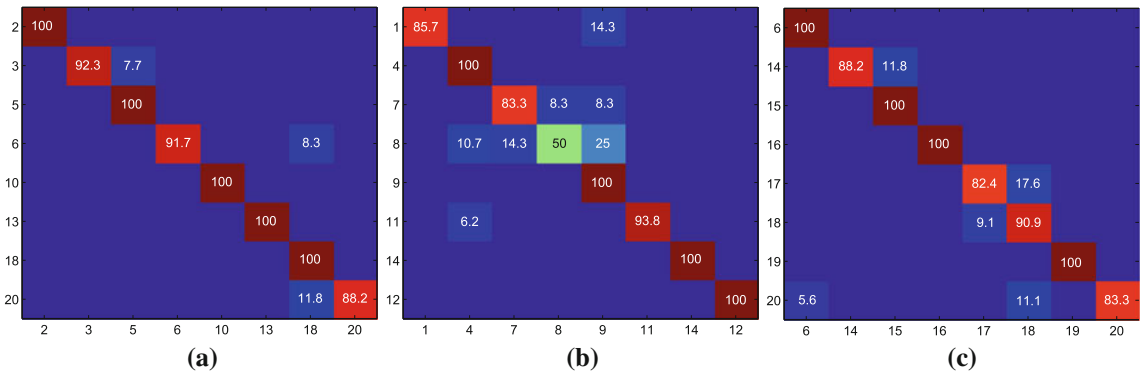

Fig. 5 Confusion matrix of our method for Fixed Cross Subject Test. **a** Subset AS1. **b** Subset AS2. **c** Subset AS3

Table 3 Average and standard deviation recognition rates (%) of our method for msr-action3D dataset in Random Tests

	Test one	Test two	Cross subject test
AS1	97.4 ± 0.9	98.5 ± 1.1	84.8 ± 4.4
AS2	96.1 ± 1.5	97.8 ± 1.4	67.8 ± 4.3
AS3	97.7 ± 1.2	98.9 ± 1.1	87.1 ± 3.7
Average	97.1 ± 1.2	98.4 ± 1.2	79.9 ± 4.1

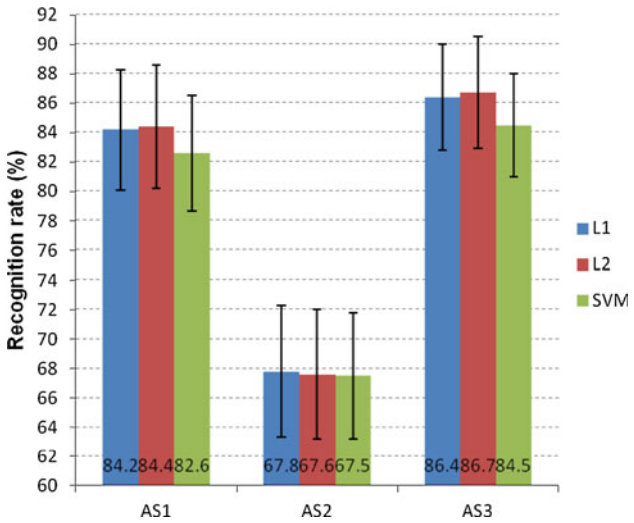

Fig. 6 Comparison of recognition rates (%) using different classifiers in Random Cross Subject Test

5.3.2 *Real-time operation*

There are four main components in our method: projected depth map generation (three views) for each depth frame, DMMs feature generation, dimensionality reduction (PCA), and action recognition (l_2-regularized collaborative representation classifier). Our real-time action recognition timeline is displayed in Fig. 7. The numbers in Fig. 7 indicate the main components in our method. The generation of the projected map and DMMs are executed right after each depth frame is captured while the dimensionality reduction and action recognition are performed after an action sequence gets completed. Since the PCA transform matrix is calculated using the training feature set, it can be directly applied to the feature vector of a test sample. Our code is written in Matlab and the processing time reported is for a PC with 2.67 GHz Intel Core I7 CPU with 4 GB RAM. The average processing time of each component is listed in Table 4. Note that the average number of depth frames in an action video sequence (after frame removal) is about 30.

The computational complexity aspect of the major components involved in different methods are provided in

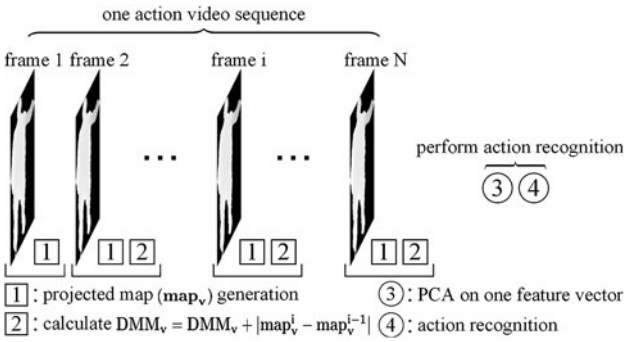

Fig. 7 Real-time action recognition timeline

Table 4 Average and standard deviation of processing time of the components of our method

Components	Processing time (ms)
1	2.0 ± 0.4/frame
2	3.3 ± 0.6/frame
3	2.5 ± 1.2/action sequence
4	1.8 ± 0.5/action sequence

Table 5 Computational complexity and speed-up performance

Method	Computational complexity of major components	Approximate speedup for a typical set of parameters
Li et al. [9]	$O(J \times K_h D^2)$	1
Lu et al. [7]	$O(K_h MP + P^3) + O(N_h H^2)$	3
Yang et al. [8]	$O(m^3 + m^2 r) + O(r \times n_c \times n_d \times \log(n_c \times n_d))$	16
Yang et al. [10]	$O(r^3)$	9
Vieira et al. [12]	$O(m^3 + m^2 r) + O(n_c \times r^2)$	20
Our method	$O(m^3 + m^2 r) + O(n_c \times r)$	24

Table 5. In [9], the bi-gram maximum likelihood decoding (BMLD) for Gaussian Mixture Model (GMM) was adopted to mitigate the computational complexity with the complexity of $O(J \times K_h D^2)$ [28], where J denotes the number of iterations, K_h the number of samples in the dataset and D the dimensionality of the state. As was reported in [7], the complexity is mostly due to the Fisher's linear discriminant analysis (LDA) and HMM. The computation of voting of joints into the bins is relatively trivial. The computational complexity for LDA is $O(K_h MP + P^3)$, where M is the number of extracted features and $P = \min(K_h, M)$. The computational complexity for HMM is $O(N_h H^2)$ [29], where N_h denotes the total number of states and H the length of the observation sequence. In [8], the computational complexity for PCA [30] and Naive-Bayes-Nearest-Neighbor (NBNN) classifier is stated as $O(m^3 + m^2 r)$ and $O[r \times n_c \times n_d \times \log(n_c \times n_d)]$, respectively, where m denotes the dimension of a sample vector, r denotes the number of training samples, n_c represents the number of classes, and n_d represents the number of descriptors. In [10], the computational complexity of SVM is stated as $O(r^3)$ [31]. In [12], the computational complexity of PCA and the classifier are stated as $O(m^3 + m^2 r)$ and $O(n_c \times r^2)$, respectively. Table 5 provides the speedup for a typical set of parameters: $J = 50$, $K_h = 200$, $D = 30$, $N_h = 6$, $H = 27$, $M = 125$, $r = 100$, $m = 50$, $n_c = 8$, $n_d = 40$. As can be seen from this table, our method is the most computationally efficient one.

6 Conclusion

In this paper, a computationally efficient DMM-based human action recognition method using l_2-regularized collaborative representation classifier was introduced. The DMMs generated from three projection views were used to capture the motion characteristics of an action sequence. An average recognition rate of 90.5 % on the MSR-Action3D dataset was achieved, outperforming the existing methods. In addition, the utilization of l_2-regularized collaborative representation classifier was shown to be computationally efficient leading to a real-time implementation.

References

1. Schuldt, C., Laptev, I., Caputo, B.: Recognition human actions: a local SVM approach. Proceedings of IEEE International Conference on Pattern Recognition, vol. 3, pp. 32–36, Cambridge, UK (2004)
2. Dollar, P., Rabaud, V., Cottrell, G., Belongie, S.: Behavior recognition via sparse spatio-temporal features. IEEE International Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance, pp. 65–72, Beijing, China (2005)
3. Sun, J., Wu, X., Yan, S., Cheong, L.F., Chua, T., Li, J.: Hierarchical spatio-temporal context modeling for action recognition. Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, pp. 2004–2011, Miami, FL (2009)
4. Laptev, I., Marszalek, M., Schmid, C., Rozenfeld, B.: Learning realistic human actions from movies. Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, pp. 1–8, Anchorage, AK (2008)
5. Bobick, A., Davis, J.: The recognition of human movement using temporal templates. IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 23, no. 3, pp. 257–267 (2001)
6. Davis, J.: Hierarchical motion history images for recognizing human motion. Proceedings of IEEE Workshop on Detection and Recognition of Events in Video, pp. 39–46, Vancouver, BC (2001)
7. Xia, L., Chen, C., Aggarwal, J.-K.: View invariant human action recognition using histograms of 3D joints. Proceedings of IEEE Conference on Computer Vision and Pattern Recognition Workshops, pp. 20–27, Providence, RI (2012)
8. Yang X., Tian, Y.: Eigen joints-based action recognition using Naïve-Bayes-Nearest-Neighbor. Proceedings of IEEE Conference on Computer Vision and Pattern Recognition Workshops, pp. 14–19, Province, RI (2012)
9. Li, W., Zhang, Z., Liu, Z.: Action recognition based on a bag of 3D points. Proceedings of IEEE Conference on Computer Vision and Pattern Recognition Workshops, pp. 9–14, San Francisco, CA (2010)
10. Yang, X., Zhang, C., Tian, Y.: Recognizing actions using depth motion maps-based histograms of oriented gradients. Proceedings of ACM International Conference on Multimedia, pp. 1057–1060, Nara, Japan (2012)
11. Wang, J., Liu, Z., Chorowski, J., Chen, Z., Wu, Y.: Robust 3D action recognition with random occupancy patterns. Proceedings of IEEE European Conference on Computer Vision, pp. 872–885, Florence, Italy (2012)
12. Vieira, A., Nascimento, E., Oliveira, G., Liu, Z., Campos, M.: Stop: Space-time occupancy patterns for 3D action recognition from depth map sequences. Iberoamerican Congress on Pattern Recognition, pp. 252–259, Buenos Aires, Argentina (2012)
13. Jiang, W., Liu, Z., Wu, Y., Yuan, J.: Mining actionlet ensemble for action recognition with depth cameras. Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, pp. 1290–1297, Province, RI (2012)

14. Shotton, J., Fitzgibbon, A., Cook, M., Sharp, T., Finocchio, M., Moore, R., Kipman, A., Blake, A.: Real-time human pose recognition in parts from single depth images. Proceeding of IEEE Conference on Computer Vision and Pattern Recognition, pp. 1297–1304, Colorado springs, CO (2011)
15. Wright, J., Yang, A., Ganesh, A., Sastry, S., Ma, Y.: Robust face recognition via sparse representation. IEEE Transaction on Pattern Analysis and Machine Intelligence, vol. 31, no. 2, pp. 210–227 (2009)
16. Wright, J., Ma, Y.: Dense error correction via *l*1 minimization. Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing, pp. 3033–3036, Taipei, Taiwan (2009)
17. Wright, J., Ma, Y., Mairal, J., Sapiro, G., Huang, T., Yan, S.: Sparse representation for computer vision and pattern recognition. Proceedings of the IEEE, vol. 98, no. 6, pp. 1031–1044 (2010)
18. Gao, S., Tsang, I.W.-H., Chia, L.: Kernel sparse representation for image classification and face recognition. Proceedings of IEEE European Conference on Computer Vision, pp. 1–14, Crete, Greece (2010)
19. Yang, J., Yu, K., Gong, Y., Huang, T.: Linear spatial pyramid matching using sparse coding for image classification. Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, pp. 1794–1801, Miami, FL (2009)
20. Zhang, L., Yang, M., Feng, X.: Sparse representation or collaborative representation: which helps face recognition? Proceedings of IEEE International Conference on Computer Vision, pp. 471–478, Barcelona, Spain (2011)
21. Shi, Q., Eriksson, A., Hengel, A., Shen, C.: Is face recognition really a compressive sensing problem? Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, pp. 553–560, Colorado springs, CO (2011)
22. Tikhonov. A., Arsenin, V.: Solutions of Ill-Posed Problems. V. H. Winston & Sons, Washington, DC (1977)
23. Chen, C., Tramel, E., Fowler, J.: Compressed-sensing recovery of images and video using multi hypothesis predictions. Proceedings of Asilomar Conference on Signals, Systems, and Computer, pp. 1193–1198, Pacific Grove, CA (2011)
24. Golub, G., Hansen, P.C., O'Leary, D.: Tikhonov regularization and total least squares. SIAM J Matrix Anal. Appl. **21**(1), 185–194 (1999)
25. Hansen, P., O'Leary, D.: The use of the L-curve in the regularization of discrete ill-posed problems. SIAM J Sci. Comput. **14**(6), 1487–1503 (1993)
26. Mairal, J.: (SPArse Modeling Software), spams-devel.gforge.inria.fr
27. Chang, C., Lin, C.: LIBSVM: a library for support vector machines. ACM Transactions on Intelligent Systems and Technology, vol. 2, no. 3, pp. 27:1–27:27 (2011). Software available at http://www.csie.ntu.edu.tw/∼cjlin/libsvm/
28. Li, W., Zhang, Z., Liu, Z.: Expandable data-driven graphical modeling of human actions based on salient postures. IEEE Transactions on Circuits and Systems for Video Technology, vol. 18, no. 11, pp. 1499–1510 (2008)
29. Fine, S., Singer, Y., Tishby, N.: The hierarchical hidden Markov model: analysis and applications. Mach. Learn. **32**(1), 41–62 (1998)
30. Liu, K., Ma, B., Du, Q., Chen, G.: Fast Motion Detection from Airborne Videos Using Graphics computing units. J.Appl. Remote Sens. vol. 6, no. 1 (2012)
31. Tsang, I., Kwok, J., Cheung, P.-M.: Core vector machines: Fast SVM training on very large data sets. J. Mach. Learn. Res. vol. 6, no. 1, pp. 363–392 (2005)

Author Biographies

Chen Chen received the B.E. degree in automation from Beijing Forestry University, Beijing, China, in 2009 and the M.S. degree in electrical engineering from Mississippi State University, Starkville, MS, in 2012. He is currently working toward the Ph.D. degree in the Department of Electrical Engineering, University of Texas at Dallas, Richardson, TX. His research interests include compressed sensing, signal and image processing, pattern recognition, and computer vision.

Kui Liu received the B.S. degree in electrical engineering from Nanchang University, Nanchang, China in 2005, and received the M.S. degree in electrical engineering from Mississippi State University, Starkville, M.S. in 2011. He is currently a graduate research assistant in the Department of Electrical Engineering at the University of Texas at Dallas as a member of Signal and Image processing Laboratory. His research interests include real-time image processing, 3-D computer vision and machine learning.

Nasser Kehtarnavaz received the Ph.D. degree in electrical and computer engineering from Rice University in 1987. He is a Professor of Electrical Engineering and Director of the Signal and Image Processing Laboratory at the University of Texas at Dallas. His research areas include signal and image processing, real-time signal and image processing, biomedical image analysis, and pattern recognition. He has authored or co-authored 8 books and more than 200 papers in these areas. He is currently Chair of the Dallas Chapter of the IEEE Signal Processing Society, Cochair of the SPIE Conference on Real-Time Image and Video Processing, and Coeditor-in-Chief of Journal of Real-Time Image Processing. He is a fellow of IEEE and SPIE.