One-Shot Learning for Real-Time Action Recognition*

Sean Ryan Fanello[1,2], Ilaria Gori[1], Giorgio Metta[1], and Francesca Odone[2]

[1] iCub Facility,
Istituto Italiano di Tecnologia
{sean.fanello, ilaria.gori, giorgio.metta}@iit.it
[2] Dipartimento di Informatica, Bioingegneria, Robotica e Ingegneria dei Sistemi,
Università degli Studi di Genova
francesca.odone@unige.it

Abstract. The goal of the paper is to develop a one-shot real-time learning and recognition system for 3D actions. We use RGBD images, combine motion and appearance cues, and map them into a new overcomplete space. The proposed method relies on descriptors based on 3D Histogram of Flow (3DHOF) and on Global Histogram of Oriented Gradient (GHOG); adaptive sparse coding (SC) is further applied to capture high-level patterns. We add effective on-line video segmentation and finally the recognition of actions through linear SVMs. The main contribution of the paper is a real-time system for one-shot action modeling; moreover we highlight the effectiveness of sparse coding techniques to represent 3D actions. We obtain very good results on the ChaLearn Gesture Dataset and with a Kinect sensor.

Keywords: Real-Time Action Recognition, One-Shot Action Learning

1 Introduction

The recognition of human actions is a very fertile research theme in computer vision due to its strong applicability in several real world domains, ranging from video-surveillance to content-based video retrieval and video classification. This paper refers specifically to action recognition in the context of Human-Machine Interaction (HMI), and therefore it focuses on actions performed by a human who is standing at a short distance from the sensor. We propose a complete system based on RGBD video sequences, which models actions from a single example; subsequently it recognizes the learned actions accurately and in real-time, even though they are performed in different environments, at different speeds, and if they are combined in sequences of actions.
The literature is rich of algorithms for gesture, action, and activity recognition — we refer the reader to [1, 2] for a complete overview of the topic. In spite of this, depth sensors have seldom been used in the past; most methods focused on features derived from RGB images: many well-known approaches rely on descriptors based on temporal templates [3], Histograms of Flow (HOFs) [4], or a combination of multiple features [5]. The recognition phase is usually carried out through complex structures such as Hidden

* Research supported by the European FP7 ICT project No. 270490 (EFAA) and project No. 270273 (Xperience).

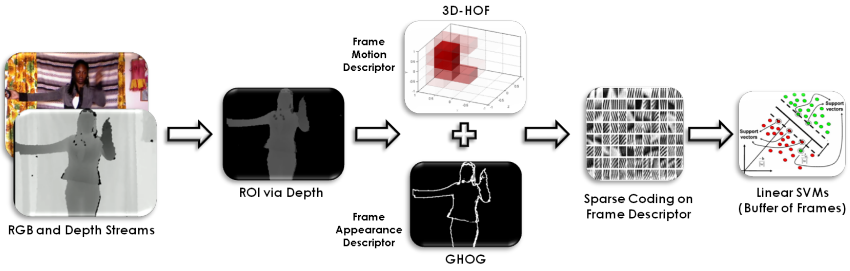

Fig. 1. Overview of the recognition system: we propose solutions that cope with real-time requirements. Each frame is represented by 3DHOF and GHOG descriptors. The learning stage implicitly model the time dependency of actions concatenating descriptors of buffer of frames.

Markov Models [6], Coupled Hidden Semi-Markov Models [7], or Action Graphs [8], but all these methods require an expensive offline learning phase. More recent works exploiting depth information and one-shot learning [9–11] show interesting results; unfortunately both methods use either batch learning or complex action models which are difficult to reconcile with real-time action recognition.

We combine motion and appearance to obtain an effective description of a wide variety of actions. In Laptev et al. [12, 13] descriptors are based on Spatial-Temporal Interest Points [14], whereas we use a single frame descriptor, and the time evolution modelling is delegated to the learning stage. The main advantages are the possibility of one-shot learning and real-time performances — in fact, spatial-temporal detectors run at most $1 - 5$fps [12], while our whole system runs at 25fps.

In more detail, after a segmentation of the actor via depth, we resort to multidimensional histograms of the 3D flow (3DHOFs), which are functional to catch motion information in the 3D space. Our descriptor naturally enables the modeling and recognition of 3D actions (i.e. forward and backward movements). Afterwords, a global histogram of oriented gradient (GHOG), is employed to represent shapes with low computational cost. In order to gain robustness and generality on the representation phase, we rely on adaptive sparse coding (SC) [15], which produces a sparse description of each frame. The temporal evolution of an action is modeled by concatenating the frame representations in a frame buffer. Then we train an SVM on a set of data obtained by slicing the sole training video in buffers. Despite SVMs are not designed for one-shot learning, they show impressive classification results when trained on small training sets, thanks to the effectiveness of the learned representation. At run time we segment the video input online and consequently detect the action occurrence from the SVM output.

Importantly our architecture is based on relatively simple features, which do not involve complex structures or complicated temporal coding methods, since we aim at real-time performance as required by HMI; nonetheless, we achieve high-quality performance, while fulfilling real-time requirements. We believe this is a strong advantage, since both high accuracy and real-time performances are fundamental in real world applications.

We assessed our framework on different publicly available and in-house datasets: a first

Fig. 2. Region of Interest via depth. On the left the RGB video frame. On the centre the depth frame. On the right the ROI containing the actor.

evaluation has been conducted on the ChaLearn Gesture Dataset [16, 17]. Then, using a Kinect sensor, we demonstrate how to model more difficult gestures, in particular those involving only finger movements. In both the test cases we report extremely good results.

2 System Overview

The system is a sequence of 4 layers and it is depicted in Fig. 1. In this section we detail the contribution of each level.

2.1 Region Of Interest Segmentation

The first step of each action recognition system is to identify correctly where the action is being performed. Most of the algorithms that have been proposed in the literature involve background modeling techniques [18]. This task is greatly simplified in our architecture, since in human-machine interaction we can safely assume the human to stand in front of the camera sensors. Indeed, after we detect the region where motion is occurring, we exploit depth information to correctly isolate the subject of interest from the background (see Fig. 2).

2.2 Frame Representation

Although many complex features describing human actions have been proposed, some of whom implying long and expensive processing routines, our belief is that simple features already have enough discriminative power, especially when we can rely on an depth based ROI. We decided to use global descriptors of the whole ROI and we made our choices on the basis of computational criteria. We show that a combination of 3D Histograms of Flow (3DHOFs) and Global Histograms of Gradient (GHOGs) models satisfactorily human actions.

3D Histogram of Flow. For each frame F_t we compute the 2D flow vectors $U(x, y, t)$ and $V(x, y, t)$ for the x and y components, with respect to the previous frame F_{t-1} via the Fanerbäck's algorithm [19]. Since we are in a calibrated context, each pixel (x_{t-1}, y_{t-1}) belonging to the ROI of the frame F_{t-1} can be easily reprojected in the

3D space $(X_{t-1}, Y_{t-1}, Z_{t-1})$. Similarly we can reproject the final point (x_t, y_t) of the 2D vector representing the flow, obtaining another 3D vector $(X_t, Y_t, Z_t)^T$. For each pixel of the ROI, we can define the scene flow as the difference of the two 3D vectors in two successive frames F_{t-1} and F_t, i.e. $\mathbf{D} = (\dot{X}, \dot{Y}, \dot{Z})^T = (X_t - X_{t-1}, Y_t - Y_{t-1}, Z_t - Z_{t-1})^T$. Once the 3D flow for each pixel of the ROI at time t has been computed we normalize it with respect to the $L2$-norm, so that the resulting descriptors $\mathbf{D_1}, \ldots, \mathbf{D_n}$ (n pixels of the ROI) are invariant to the overall speed of the action. In order to extract a compact representation we build a 3D Histogram of Flow (3DHOF) $\mathbf{z}(t)$ of the 3D motion vectors, with $\mathbf{z}(t) \in \mathbb{R}^{n_1 \times n_1 \times n_1}$ where n_1 is the quantization parameter of the space (i.e. the bin size). In addition we normalize each 3DHOF $\mathbf{z}(t)$ (i.e. $\sum_j z_j(t) = 1$), guaranteeing that these descriptors are invariant to the subject of interest's scale.

Global Histogram of Oriented Gradient. We extend the motion descriptor with a Global Histogram of Gradient (GHOG) that produces an overall description of the appearance of the ROI. Notably, we compute GHOGs on the depth image, since the shape of the body is not influenced by its texture. We compute the histogram of gradient orientations of the pixels in the ROI, obtaining another descriptor $\mathbf{h}(t) \in \mathbb{R}^{n_2}$, where n_2 is the number of bins. The scale invariance property is preserved normalizing the descriptor so that $\sum_j h_j(t) = 1$. The choice of a global descriptor rather than a local one as in [20], is mainly due to computational complexity.
At this stage each frame F_t is represented by two descriptors: $\mathbf{z}(t)$ for the motion component and $\mathbf{h}(t)$ for the pose/appearance component.

2.3 Sparse Coding Stage

Although image descriptors usually belong to a high dimensional space, it is likely that meaningful features in the data lie in the same low-dimensional space. Thus we employ sparse coding [15], that has been recently investigated in the Machine Learning community. Adaptive sparse coding approaches often need an early stage, called Dictionary Learning, which performs the learning of the atoms (i.e. the elements of the dictionary) directly from the data. Specifically, given the set of the previously computed 3DHOFs $\mathbf{Z} = [\mathbf{z}_1, \ldots, \mathbf{z}_N]$ where N is the number of all the frames in the training data (for clarity, we suppress the explicit dependence on t in the notation), our goal is to learn one dictionary $\mathbf{D_1}$ and a code $\mathbf{U_1}$ that minimize the reconstruction error:

$$\min_{\mathbf{D_1}, \mathbf{U_1}} \|\mathbf{Z} - \mathbf{D_1}\mathbf{U_1}\|_F^2 + \lambda \|\mathbf{U_1}\|_1 \tag{1}$$

where $\|\cdot\|_F$ is the Frobenius norm. Notice that fixing $\mathbf{U_1}$, the above optimization reduces to a least square problem, whilst, given $\mathbf{D_1}$, it is equivalent to a linear regression with the sparsifying norm $L1$. The latter problem is referred to as a feature selection problem with a known dictionary. We used the *feature-sign search* algorithm [15] for the solution of the above minimization problem. The same approach is applied for GHOG descriptors, therefore, after the Sparse Coding stage, we can describe a frame as a code $\mathbf{u_i}$, which is the concatenation of the motion and pose codes.

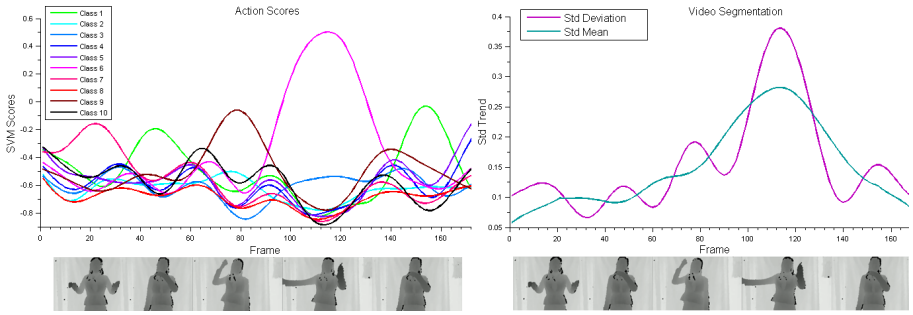

Fig. 3. The figure illustrates on the left the SVMs scores (Eq. 2) computed in real-time at each time step T over a sequence of 170 frames. On the right the standard deviation of the scores and its mean computed on a sliding window are depicted. The local minima of the standard deviation function are break points that define the end of an action and the beginning of another one. See Sec. 2.4 for details.

2.4 Learning and Recognition

The learning and recognition phase has been carried out with Support Vector Machines (SVMs) [21]. We employ linear SVMs, since they ensure constant complexity during the test phase fulfilling real-time requirements. During the learning stage we model the temporal dependency of actions: from the frame descriptor (described in Sec. 2.2 and 2.3) we move to a concatenation of frames for the classification. Therefore, a set of T frames is represented as a sequence $[\mathbf{u}_1, \ldots, \mathbf{u}_T]$ codes. Moreover we define an online method to segment the video sequence.

Learning Actions. Given a video V_s of t_s frames, containing only one action A_s, we compute a set of descriptors $[\mathbf{u_1}, \ldots, \mathbf{u_{t_s}}]$ as described in Sec. 2.2 and 2.3. At this stage the data are the descriptions of a frame buffer $\mathbf{B}_T(t)) = (\mathbf{u_{t-T}}, \ldots, \mathbf{u_{t-1}}, \mathbf{u_t})^T$, where T is its length. We use a one-versus-all strategy to train a binary linear SVM for each class A_s, so that at the end of the training phase we obtain a set of N linear SVM classifiers $f_1(\bar{\mathbf{B}}), \ldots, f_N(\bar{\mathbf{B}})$, where N is the number of actions. In particular, in this one-shot learning pipeline, the set of buffers $\mathbf{B_s} = [\mathbf{B}_T(t_0), \ldots, \mathbf{B}_T(t_s)]$ computed from the single video V_s of the class A_s are used as positive examples for the action A_s. All the buffers belonging to A_j with $j \neq s$ are the negative examples. Although we use only one example for each class, we benefit from the chosen representation: indeed, descriptors are computed per frame, therefore one single video of length t_s provides a number of examples equal to $t_s - T$ where T is the buffer size.

Online Recognition: Video Segmentation. Given a test video V, which may contain one or more known actions, the goal is to predict correctly the sequence of the performed actions. The video is analyzed through a sliding window $\mathbf{B}_T(t)$ of size T. We compute the output score of the learned SVM machines for each test buffer $\mathbf{B}_T(t)$

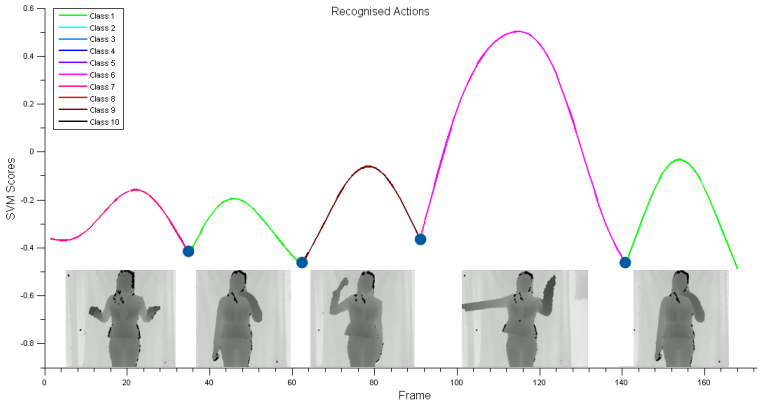

Fig. 4. The figure illustrates only the scores of the recognized actions via the method described in Sec. 2.4. Blue dots are the break points computed by the video segmentation algorithm that indicate the end of an action and the beginning of a new one.

and we filter these scores with a low-pass filter W that attenuates high frequencies and noise. Therefore the new score at time t becomes:

$$H_s(\mathbf{B}_T(t)) = W \star f_s(\mathbf{B}_T(t)) \tag{2}$$

where the $\star$ is the convolution operator. Fig. 3 depicts an example of these scores computed in real-time. As long as the scores evolve we need to predict online when an action ends and another one begins; this is achieved computing the standard deviation $\sigma(H)$ for a fixed t over all the scores H_i^t (Fig. 3, right chart). The local minima of the retrieved function suggest the end of an action and the beginning of a new one. Let n be the number of local minima computed from the standard deviation function; there will be $n+1$ actions, and in particular actions with the highest score before and after each break point will be recognized. We can easily find these minima in real-time: we calculate the mean value of the standard deviation over time using a sliding window. When the standard deviation trend is below the mean, all the SVMs scores predict similar values, hence it is likely that an action has just ended. In Fig. 4 the segmented and recognized actions are depicted with their scores.

3 Experiments

The first evaluation has been conducted on a publicly available dataset, **ChaLearn Gesture Dataset**, released by ChaLearn [16]. We compare our method with other techniques, mentioned in [16], whose methods have been published. In the second setting we show how to improve the recognition rate using the whole functionality of a real **Kinect Sensor**. For the computation of the accuracy between a sequence of actions returned by our system and the ground truth sequence we use the normalized Levenshtein

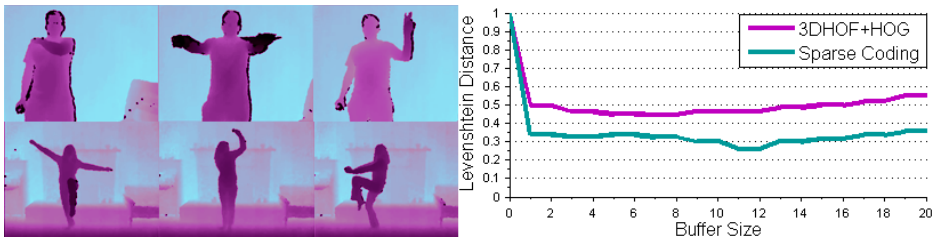

Fig. 5. On the left examples of 2 different batches from the ChaLearn Dataset [16]. On the right the overall Levenshtein Distance computed in 20 batches with respect to the buffer size parameter is depicted for both 3DHOF+GHOG features and descriptors processed with sparse coding.

Distance [22] defined as:

$$TeLev = \frac{S + D + I}{N} \tag{3}$$

where S is the number of substitutions (misclassifications), D the number of deletions (false negatives), I the number of insertions (false positives) and N the length of the ground truth sequence. We empirically chose a quantization parameter for the 3DHOF equal to 5, 64 bins for the GHOG descriptor and dictionary size equal to 256 for both motion and appearance components. This led to a frame descriptor of size 189 for simple descriptors, which increases to 512 after the sparse coding processing. The whole system runs at 25fps on 2.4Ghz Core 2 Duo Processor.

Table 1. Levenshtein Distance on the ChaLearn Gesture Dataset. For SVM classification we chose the best buffer size for each batch of the dataset. TeLev is the Levenshtein Distance, TeLen is the average error made on the number of gestures.

Method	TeLev	TeLen	Frame Descriptor Size
Sparse Coding **(ours)**	**25.11%**	**5.02%**	512
3DHOF + GHOG **(ours)**	43.32%	9.03%	189
Template Matching	62.56%	15.12%	320×240
Dynamic Time Warping	49.41%	Manual Split	189
Manifold LSR [10]	28.73%	6.24%	NA
MHI [9]	30.01%	NA	NA
Extended-MHI [9]	26.00%	NA	NA
BoVW [9]	72.32%	NA	NA

3.1 ChaLearn Gesture Dataset

The dataset [16, 17] (see Fig. 5) is organized in batches, where each batch includes 100 recorded gestures grouped in sequences of 1 to 5 gestures arbitrarily performed with

different speed. The gestures are drawn from a small vocabulary of 8 to 15 unique gestures called **lexicon**, which is defined within a batch. For each video both RGB and Depth streams are provided, but only one example is provided for the training phase. We did not use information on the body pose of the human. For this test we considered the batches from *devel_01* to *devel_20*; each batch has 47 videos, where L are training videos (L is the lexicon size) and the others are used for test. The main parameter of the system is the buffer size T, however in Fig. 5 it is possible to notice that the parameter offers stable performances with a buffer range of $1-20$, so it does not represent a critical variable of our method. We compute the Levenshtein Distance as the average over all the batches, which is 25.11% for features processed with sparse coding, whereas simple 3DHOF+GHOG descriptors lead to a performance of 43.32%. Notably, each batch has its own lexicon and some of them are composed of only gestures performed by hand or fingers; in these cases, if the GHOG is computed on the entire ROI, the greatest contribution of the histogram comes from the body shape, whilst finger actions (see Fig. 2) represent a poor percentage of the final descriptor. If we consider batches where the lexicon is not composed of only hand/finger gestures, the Levenshtein Distance reduces to 15%.

We compared our method with several approaches. First of all a Template Matching technique, where we used as descriptor the average of all depth frames for each action. The test video is split in slices estimated using the average size of actions. In recognition phase we classify each slice of the video comparing it with all the templates. The overall Levenshtein Distance becomes 62.56%. We then compare our approach with Dynamic Time Warping (DTW) [23] with 3DHOF + GHOG features. We manually divided test videos in order to facilitate the recognition for DTW; nevertheless the global Levenshtein Distance is 49.41%. Finally we report the results obtained by the recent works in the field, which exploit techniques based on manifolds [10], Motion History Image (MHI) [9] and Bag of Visual Words (BoVW) [9].

Table 1 shows that all the compared approaches are outperformed by our method. Furthermore our online video segmentation algorithm shows excellent results with respect to the temporal segmentation used in the compared frameworks; in fact it is worth noting that the proposed video segmentation algorithm leads to an action detection error rate $TeLen = \frac{FP+FN}{N}$ equal to 5.02%, where FP and FN are the false positive and the false negative actions respectively and N is the number of all test gestures. The presented results show that SVMs seem to be an effective way to model actions even in one-shot learning settings.

3.2 Kinect Data

In this subsection we use a Kinect for Xbox 360 sensor to test our approach against more complex gestures. In Sec. 3.1, we remarked that the proposed appearance descriptor lacks when actions differ by small details, therefore a local description of the ROI rather then a global one would be more effective. The simplest way to add this local information is to resort to a body part tracker. An excellent candidate able to provide a reliable body tracker is Microsoft Kinect SDK [24], which implements the method in [25]. This tool retrieves the 20 principal body joints position and pose of the user's current posture. Given these positions we assign each 3D point of the ROI to its nearest

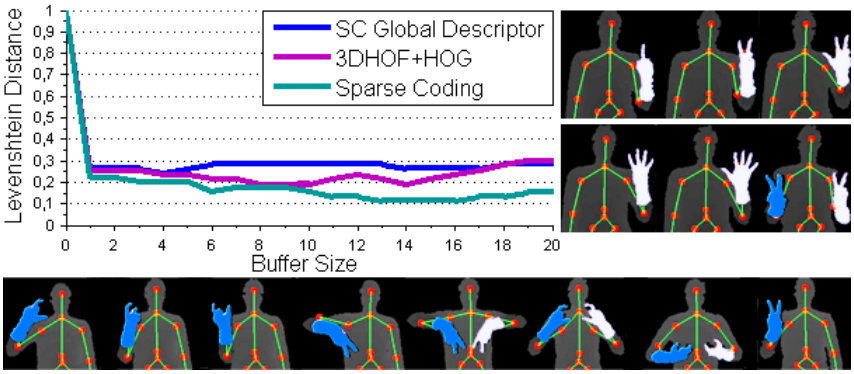

Fig. 6. On the right and bottom the two vocabularies used in Sec. 3.2; these gestures are difficult to model without a proper body tracker, indeed the most contribution for the GHOG comes from the body shape rather than the hand. On the left the Levenshtein Distance.

joint, so that it is possible to correctly isolate the two hands and the body from the rest of the scene (see Fig. 6). Then, we slightly modify the approach, computing 3DHOF and GHOG descriptors on the three different body parts (left/right hand and body shape); the final frame representation becomes the concatenation of all the part descriptors. We have defined two different datasets: in the first one the lexicon is composed of numbers performed with fingers, in the second one we tried to replicate one of the lexicons of the ChaLearn Gesture Dataset where we got the worst results, namely the *devel_3*. In Fig. 6 on the left the overall accuracy is depicted; using sparse coding descriptors computed on the whole body shape we obtain a Levenshtein Distance around 30%, instead computing per part descriptors the system achieves 10% for sparse coding and 20% for normal descriptors.

4 Discussion

We presented a complete system for learning and recognition of 3D human actions. We provide the possibility of **One-Shot Learning**: only one example needs to be provided to teach an action to the system. An **Effective Frame Representation** is proposed: starting from a computationally inexpensive description that combines global motion (3DHOF) and appearance (GHOG) information over the ROI, subsequently processed with a sparse coding stage, we obtain a sparse representation of each frame, which can handle a wide variety of actions. We developed an effective and reliable **Online Video Segmentation** method that leads to a 5% of action detection error rate on 2000 actions grouped in sequences of 1 to 5 gestures. Finally we achieved **Real-time Performances**: the proposed system can be used in real-time application, as it does not require neither a complex features processing nor a computationally expensive testing phase. We stress here the simplicity of this framework: each stage is easily implementable and fast to compute, without entailing elaborate and expensive data processing. Nevertheless, it

seems adequate to face the problem of gesture recognition, indeed we obtained high-quality results while fulfilling real-time requirements.

References

1. Aggarwal, J., Ryoo, M.: Human activity analysis: A review. ACM Computing Surveys (2011)
2. Poppe, R.: A survey on vision-based human action recognition. Image and Vision Computing (2010)
3. Bobick, A.F., Davis, J.W.: The recognition of human movement using temporal templates. TPAMI (2001)
4. Fanello, S.R., Gori, I., Pirri, F.: Arm-hand behaviours modelling: from attention to imitation. In: 6th international conference on Advances in visual computing. (2010)
5. Laptev, I., Marszalek, M., Schmid, C., Rozenfeld, B.: Learning realistic human actions from movies. In: CVPR. (2008)
6. Yamato, J., Ohya, J., Ishii, K.: Recognizing human action in time-sequential images using hidden markov model. In: CVPR. (1992)
7. Natarajan, P., Nevatia, R.: Coupled hidden semi markov models for activity recognition. In: Workshop Motion and Video Computing. (2007)
8. Li, W., Zhang, Z., Liu, Z.: Action recognition based on a bag of 3d points. In: CVPRW. (2010)
9. Wu, D., Zhu, F., Shao, L.: One shot learning gesture recognition from rgbd images. In: CVPR Workshop on Gesture Recognition. (2012)
10. Lui, Y.M.: A least squares regression framework on manifolds and its application to gesture recognition. In: CVPR Workshop on Gesture Recognition. (2012)
11. Seo, H., Milanfar, P.: Action recognition from one example. PAMI (2011)
12. Wang, H., Ullah, M.M., Klser, A., Laptev, I., Schmid, C.: Evaluation of local spatio-temporal features for action recognition. In: BMVC. (2009)
13. Schuldt, C., Laptev, I., Caputo, B.: Recognizing human actions: a local svm approach. In: ICPR. (2004)
14. Lv, F., Nevatia, R.: Single view human action recognition using key pose matching and viterbi path searching. In: CVPR. (2007)
15. Lee, H., Battle, A., Raina, R., Ng, A.Y.: Efficient sparse coding algorithms. In: NIPS. (2007)
16. ChaLearn: Chalearn gesture dataset (cgd2011). http://gesture.chalearn.org/data (2011)
17. Guyon, I., Athitsos, V., Jangyodsuk, P., Hammer, B., Balderas, H.J.E.: Chalearn gesture challenge: Design and first results. In: In CVPR Workshop on Gesture Recognition. (2012)
18. Stauffer, C., Grimson, W.E.L.: Adaptive background mixture models for real-time tracking. CVPR (1999)
19. Farnebäck, G.: Two-frame motion estimation based on polynomial expansion. In: 13th Scandinavian Conference on Image Analysis. (2003)
20. Dalal, N., Triggs, B.: Histograms of oriented gradients for human detection. CVPR (2005)
21. Vapnik, V.: Statistical Learning Theory. John Wiley and Sons, Inc. (1998)
22. Levenshtein, V.: Binary codes capable of correcting deletions, insertions, and reversals. Soviet Physics - Doklady (1966)
23. Sakoe, H., Chiba, S.: Dynamic programming algorithm optimization for spoken word recognition. Acoustics, Speech and Signal Processing (1978)
24. Microsoft: Kinect for windows. http://kinectforwindows.org/ (2011)
25. Shotton, J., Fitzgibbon, A., Cook, M., Sharp, T., Finocchio, M., Moore, R., Kipman, A., Blake, A.: Real-time human pose recognition in parts from a single depth image. In: CVPR. (2011)