HoP: Histogram of Patterns for Human Action Representation

Vito Monteleone(✉), Liliana Lo Presti, and Marco La Cascia

Universita' degli Studi di Palermo, Palermo, Italy
vito.monteleone@unipa.it

Abstract. This paper presents a novel method for representing actions in terms of multinomial distributions of frequent sequential patterns of different length. Frequent sequential patterns are series of data descriptors that occur many times in the data. This paper proposes to learn a codebook of frequent sequential patterns by means of an apriori-like algorithm, and to represent an action with a *Bag-of-Frequent-Sequential-Patterns* approach. Preliminary experiments of the proposed method have been conducted for action classification on skeletal data. The method achieves state-of-the-art accuracy value in cross-subject validation.

Keywords: Action classification · Apriori algorithm · Frequent pattern

1 Introduction

In this work we propose to represent time series of descriptors by means of distributions of frequent sequential patterns of different length for action classification. We define a sequential pattern as a series of data descriptors indexed in time order, and a frequent pattern is one that occurs many times in the data [10].

A classical approach to represent actions is *Bag Of Visual Words (BoVW)* [5, 8,13,14,16]. In BoVW, an action is represented as a distribution of image/video patches (visual words). The codebook of visual words is generally computed by clustering algorithms, i.e. k-means [9,12,15,17]. To consider the dynamics of visual information in a time series within BoVW, spatio-temporal descriptors extracted from fixed-length cuboids [13,14,16] or multi-scale time windows [4] have been used. Visual feature dynamics are especially useful for discriminating actions that share similar body poses but show different temporal evolution; as an example, *sit down* and *get up* are actions sharing similar body poses, but these poses appear in different time order.

In contrast to the classical BoVW approach, we describe an action by means of frequent sequences of visual descriptors, thus focusing more on the body motion dynamics rather than actual body poses. Figure 1 gives an overview of the proposed *Bag-of-Frequent-Sequential-Patterns* approach. In our approach, the codebook of frequent sequential patterns is computed by means of a modified apriori algorithm [1,6]. Our implementation of the apriori algorithm allows

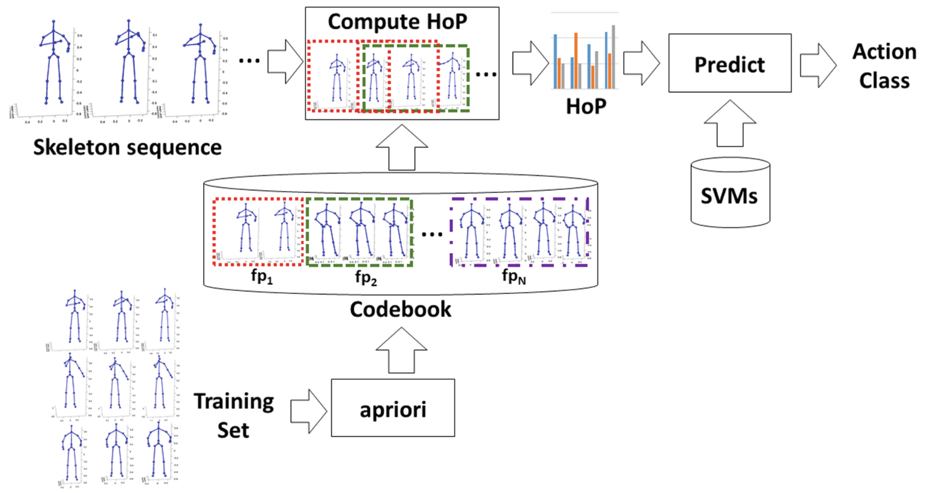

Fig. 1. *Bag-of-Frequent-Sequential-Patterns:* a test sequence is encoded in terms of frequent sequential patterns (fp_1, fp_2, ..., fp_N) by means of vector quantization; hence, a histogram of frequent sequential patterns is computed and used to predict the action class based on 1-vs-1 SVMs. In the proposed approach, the codebook is learned by a modified apriori algorithm on the training set.

us to calculate frequent patterns of different lengths, which represent different levels of body motion details. While in general clustering algorithms group elements based only on pairwise element similarities, our technique considers both similarity and frequency of the elements when learning a codebook of frequent sequential patterns. This allows us to ignore infrequent patterns that might be less informative or even confusing for classification purposes.

To summarize, our contribution in this paper is twofold:

1. we represent actions by multinomial distributions of frequent sequential patterns;
2. we propose an apriori algorithm-based learning approach for codebook of frequent sequential patterns.

We demonstrate our approach in the context of 3D skeleton-based action classification [11]. The proposed framework can be easily extended to other kinds of visual descriptors such as histograms of STIP features [16] or HOG [3]. We present preliminary experimental results on the Microsoft Research Cambridge-12 (MSRC-12) gesture dataset [18] in cross-subject validation. Our technique achieves state-of-the-art accuracy values.

The paper is organized as follows. Section 2 discusses related work; Sect. 3 explains both our modified apriori algorithm for learning a codebook of frequent sequential patterns, and how to represent an action in terms of histogram of frequent patterns (HoP); Sect. 4 presents experimental results and, finally, Sect. 5 discusses conclusions and future work.

2 Related Work

Two of the most successful approaches for representing visual content in images or videos, dictionary-based representations and Bag of Visual Words (BoVW), are based on dictionary/codebook learning. In dictionary-based representation approaches, the signal is represented as a linear combination of elements of a dictionary [23]. In Bag-of-Visual-Words (BoVW) approaches [14], introduced first for visual categorization in [5], visual content of images/videos is represented in terms of distributions of elements (codewords) in a codebook. Whilst [5] adopts a Bayesian hierarchical model to learn such kind of distributions, in practice the most commonly used pipeline requires the following steps [14]: local feature extraction, learning of a codebook by means of clustering techniques (e.g., k-means), vector quantization (for discretization of the analyzed signal in terms of codewords), codewords-based histogram computation. Such kind of paradigm has been adopted for action representation in several former works [4,8,9,12,13,15–17] In particular, in [4], sequences are represented as a distribution of local temporal texture descriptors estimated at different time scales. A codebook of multi-scale local representations is learned via k-means, and classification is performed via SVM. In [22], a codebook of temporal windows is learned via spectral clustering of data subsequences. Similarly to [4,22], we represent an action as a distribution of temporal windows of different lengths, but we adopt a data mining technique rather than a clustering technique to learn a codebook.

In the context of 3D Action Representation from skeletal data [11], the work in [20] represents actions in terms of co-occurring spatial and/or temporal configurations (poses) of specific body parts. A bag-of-words approach is adopted to represent an action where the codebook comprises co-occurring body-parts and is learned by contrast mining technique. In this sense, the codebook represents emerging patterns, that is patterns whose supports change significantly from one class to another. The work in [21] applies the apriori algorithm to find discriminative actionlet. An actionlet is defined as a subset of joints in the skeleton, and an action is represented by a linear combination of actionlets whose weights are learned via a multiple kernel learning approach. In contrast to this approach, our method aims at mining frequent sequential patterns and representing actions with a Bag-of-Frequent-Patterns approach. Our modified apriori algorithm is inspired by the work in [6]. The work in [6] focuses on detecting reduplications in a video of American Sign Language (ASL). The method detects frequent sequential patterns of increasing length by combining smaller frequent sequential patterns, and relies on approximate matching of the discovered sequential patterns with data. In counting frequencies of patterns, a waiting mechanism is used to account for poor matching arising in presence of small misalignments between patterns and data sequence. In this sense, [6] finds gapped sequential patterns. The focus of our paper is action classification; we use a method similar in spirit to [6] for mining sequential patterns to be added to our codebook. We apply our technique to a set of data streams rather than a single stream and look for non-gapped sequential patterns. During the pattern discovery process, all frequent patterns that do not contribute to the generation of longer patterns are added to our codebook. In contrast to [6], we learn frequent pattern models by averaging over matched data windows. In practice, this strategy proved to account for noise in data.

3 Representation by Histogram of Frequent Patterns

As shown in Fig. 1, a time series is represented as a histogram of frequent patterns by matching subsequences with the patterns stored in the codebook.

Frequent patterns may be found by data mining techniques, such as the apriori algorithm proposed for transactional databases [1]. In such kind of applications, a pattern $C^{(k)}$ is a set of k items from an alphabet $\mathscr{A}$, and the problem is that of finding the longest frequent patterns in the database.

Since in transactional databases there is no need of considering the order of the items within the patterns, the method is not appropriate for sequential data, such as time series, and requires some modifications in order to calculate frequent ordered item-sets. Modified apriori-algorithm for sequential data have been proposed in [2,6,10]. In particular, the method in [6] deals with the discovery of reduplication of ASL within a single data stream. As we will detail next, we borrow some of the ideas in [6] and adapt them to the learning (rather than discovering) of sequential patterns from a set of time series.

3.1 Codebook of Frequent Patterns

The main idea behind apriori-like algorithms is that a pattern $C^{(k)}$ is frequent if and only if each pattern $C^{(k-1)} \subset C^{(k)}$ is frequent as well. Therefore a frequent pattern $C^{(k)}$ may be generated iteratively by extending a pattern $C^{(k-1)}$ with an item $i \in \mathscr{A}$, and ensuring that the generated pattern is composed of only frequent sub-patterns.

At the k-th iteration, apriori-like algorithms consist mainly of three steps:

– Generation of candidates of length k by frequent patterns of length $k-1$;
– Counting of candidate frequencies;
– Removal of infrequent patterns.

Infrequent patterns have a frequency count lower than a predefined threshold ψ.

We modified these steps to adapt them to the processing of sequential data. Algorithm 1 shows the work-flow required to discover frequent patterns from training data $\mathscr{D}$. The algorithm generates frequent sequential patterns $\underline{C}^{(K_M)}$ of maximal length K_M. At the k-th iteration, $\underline{C}^{(k)}$ is a set of patterns $C_i^{(k)}$ with $i \in [1, \ldots, N_k]$, where N_k represents the number of frequent sequential patterns of length k that have been found in data $\mathscr{D}$. Each $C_i^{(k)}$ is an ordered sequence of feature descriptors $c_{i,j}$, i.e. $C_i^{(k)} = [c_{i,1}, c_{i,2}, \ldots, c_{i,k}]$. The set *codebook* stores frequent sequential patterns of different-length. The set $\underline{fp}^{(k-1)}$ stores frequent sequential patterns of length $k-1$ that cannot be used to generate longer patterns.

Algorithm 1. Learning a codebook of frequent sequential patterns

```
1: function CODEBOOK = CODEBOOKLEARNING($\mathscr{D}$, $K_M$)
2:     $k \leftarrow \tau$
3:     codebook $\leftarrow \emptyset$
4:     $\underline{C}^{(k)} \leftarrow$ generateCandidatePatterns($\mathscr{D}$, k)
5:     while $k < K_M$ do
6:         $k \leftarrow k + 1$
7:         $[\underline{C}^{(k)}, \underline{fp}^{(k-1)}] \leftarrow$ newCandidatePatternGeneration($\underline{C}^{(k-1)}$)
8:         codebook $\leftarrow$ codebook $\cup\ \underline{fp}^{(k-1)}$
9:         $\underline{C}^{(k)} \leftarrow$ duplicatesRemoval($\underline{C}^{(k)}$)
10:        getFrequencies($\underline{C}^{(k)}, \mathscr{D}$)
11:        $\underline{C}^{(k)} \leftarrow$ infrequentPatternsRemoval($\underline{C}^{(k)}$)
12:    end while
13:    codebook $\leftarrow$ codebook $\cup\ \underline{C}^{(K_M)}$
14: end function
```

Candidate Pattern Generation: In the classical apriori algorithm [1], the initial set of items (alphabet $\mathscr{A}$) is known. In our application, this initial set is unknown and we start the algorithm with all possible windows of minimal length τ extracted from the data streams with a sliding window approach. We refine such initial set of candidate patterns $\underline{C}^{(\tau)}$ by pruning the duplicated and infrequent ones as detailed later.

Candidate Pattern Frequencies: Given a set of candidate patterns $\underline{C}^{(k)}$ and data $\mathscr{D}$, we need to count how many times each candidate pattern occurs in the data. In contrast to the classical apriori algorithm, our method entails the processing of non categorical data; therefore we need a strategy to establish approximate matches between candidate patterns and data. In particular, each candidate pattern $C_i^{(k)}$ has to be compared against temporal windows extracted from data and of the same length as the considered pattern. Let us assume for a moment that $\mathscr{D}$ contains only one sequence, i.e. $\mathscr{D} = [d_1, d_2, \dots d_N]$, and consider a pattern $C_i^{(k)} = [c_{i,1}, c_{i,2}, \dots, c_{i,k}]$. We consider a sliding window $W_t = [d_t, d_{t+1}, \dots, d_{t+k-1}]$. The similarity between the candidate pattern and the temporal window W_t is measured by the following similarity score:

$$s(C_i^{(k)}, W_t) = \frac{1}{k} \cdot \sum_{j=1}^{k} e^{-\lambda \cdot ||c_{i,j} - d_{t+j-1}||_2} \tag{1}$$

where λ is a scaling parameter that multiplies the per-item squared Euclidean distance. When this score is greater than a threshold ϵ, it is possible to establish a match between the pattern and the window, and increment the candidate pattern frequency. For each pattern, we keep track of the matched temporal windows by considering the list $\underline{W}^{C_i} = \{W_j\}_{j \in J}$.

New Candidate Pattern Generation and Codebook Learning: Let us consider two frequent patterns $C_1^{(k-1)} = [c_{1,1}, c_{1,2}, \dots, c_{1,k-1}]$ and $C_2^{(k-1)} = [c_{2,1}, c_{2,2}, \dots, c_{2,k-1}]$ such that $c_{1,j} = c_{2,j-1}$ $\forall j \in [2, k-1]$. Following [6], a candidate frequent pattern of k items can be defined as $C^{(k)} = [C_1^{(k-1)}, c_{2,k-1}]$. Figure 2 sketches the new candidate pattern generation procedure.

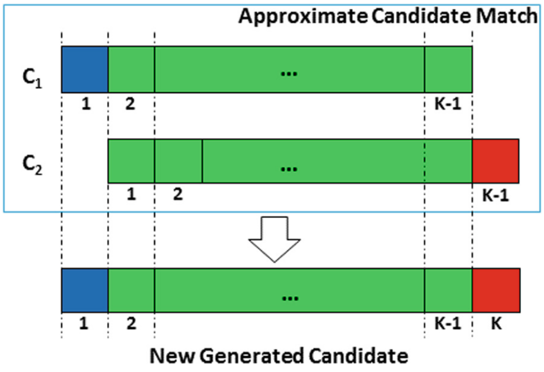

Fig. 2. The figure illustrates the idea behind the candidate pattern generation process. The new generated candidate is formed by concatenating the first item of C_1, the items shared by both C_1 and C_2, and the last item of C_2.

This candidate generation procedure would work in case of exact match of the items. In our implementation, we establish approximate matches between candidate patterns $C_1^{(k-1)}$ and $C_2^{(k-1)}$ when all corresponding items score a similarity greater than ϵ. By defining the following binary variable:

$$m(C_1^{(k-1)}, C_2^{(k-1)}) = \prod_{j=2}^{k-1} (e^{-\lambda \cdot ||c_{1,j} - c_{2,j-1}||} \geq \epsilon), \qquad (2)$$

if $m(C_1^{(k-1)}, C_2^{(k-1)})$ is equal to 1 then an approximate match between the two candidate patterns can be established.

In contrast to [6], where the items of each frequent pattern comes from the data stream, we learn a pattern model by means of the lists of matched windows of the two candidate patterns, respectively $\underline{W}^{C_1}$ and $\underline{W}^{C_2}$. The new generated pattern will have the form $C^{(k)} = [\mu_1, \mu_{2:k-1}, \mu_k]$ where μ_1 is the expected value of the first item of $C^{(k)}$ and is computed by averaging the first elements of the windows in $\underline{W}^{C_1}$; $\mu_{2:k-1}$ are expected values of subsequent items in the pattern $C^{(k)}$ and are calculated by considering both the items of windows in $\underline{W}^{C_1}$ and windows in $\underline{W}^{C_2}$; finally, μ_k is the expected value of the last item in $C^{(k)}$ and is computed by averaging the last elements of the windows in $\underline{W}^{C_2}$.

Whenever a candidate pattern of length $k-1$ does not contribute to generate candidate patterns of length k, and its frequency is greater than a threshold ψ, then the pattern is stored into the codebook.

Removal of Duplicated and Infrequent Candidate Patterns: After the generation step, a pairwise comparison of candidate patterns is carried on. Each pair of candidates with a similarity score greater than ϵ is replaced by a new candidate generated averaging the lists of matched windows. Such kind of pruning is necessary to deal with approximate matches between data and patterns. To focus on frequent patterns, candidate patterns with a frequency count smaller than a threshold ψ are considered infrequent and, hence, pruned.

3.2 Histogram of Frequent Patterns

Provided with a codebook of N frequent sequential patterns $\{C_i\}_{i\in[1,N]}$ of different length, we aim at representing a time series $V = \{y_1, y_2, \ldots, y_v\}$ as a histogram of frequent patterns (HoP) by performing vector quantization (VQ) [14]. For each frame in V and for each pattern C_i in the codebook, we consider a subsequence of V that starts from the current frame, and of length equal to that of the considered pattern C_i. We compare each window to the patterns by the score in Eq. (1) and only increment the bin of the histogram that corresponds to the pattern achieving the highest similarity (i.e. we apply hard coding).

At the top of Fig. 3, a sample of the action class *Push-Right* is shown. The bar under the sequence indicates which patterns in the codebook have been detected in the sequence (each color corresponds to a different pattern); the patterns are represented under the bar while, at the bottom of the figure, the histogram of patterns is plotted.

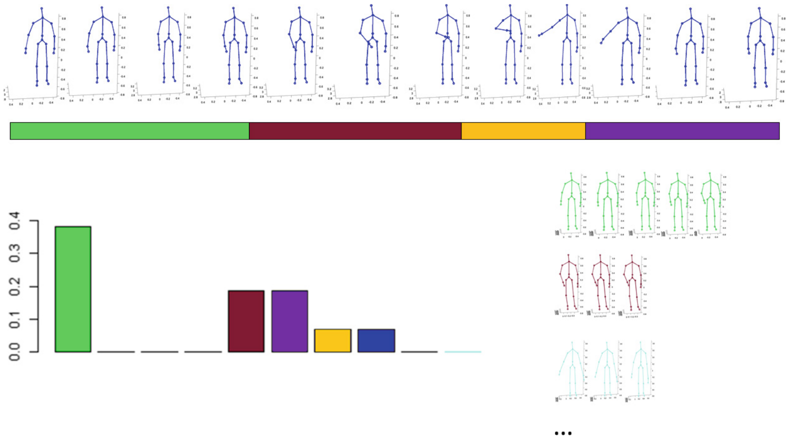

Fig. 3. The figure illustrates the HoP of a sample of the *Push* − *Right* class in terms of frequent patterns learned by our apriori algorithm. In the figure, only few distinctive skeletons of the sequence and of the patterns are shown.

4 Experimental Results

We validated our technique on the Microsoft Research Cambridge-12 (MSRC-12) gesture dataset [18]. The dataset consists of sequences of skeletons described by means of the coordinates of 20 3D body joints. Skeletons were estimated by using the Kinect Pose Estimation pipeline [19]. The dataset includes 594 sequences representing the performances of 12 actions (*Start system* (SS), *Duck* (D), *Push Right* (PR), *Goggles* (G), *Wind it up* (W), *Shoot* (S), *Bow* (B), *Throw* (T), *Had enough* (H), *Change weapon* (C), *Beat both* (BB), *Kick* (K)) from 30 different subjects. Each sequence is a recording of one subject performing one gesture several times. Considering that the MSRC-12 dataset has been proposed for action detection, no temporal segmentation of the single performance is provided with the dataset but only the time when the action is considered recognizable. In order to test our method in action classification, we adopted the annotation made publicly available by [7]. Such annotation specifies the initial/final frame when each performance starts/ends. This annotation has produced 6243 different action sequences. In order to account for biometric differences, we preprocessed each action sequence by removing its average skeleton. In general, mining algorithms are used over a single sequence to discover repetitive patterns. In contrast, our algorithm learns frequent patterns over the entire training dataset, which includes segmented action sequences from different classes and performed by different subjects. Thus, our training approach allows us to learn more general frequent patterns. We repeated the experiment 10 times in cross-validation with a 50% subject split experimental protocol, that is we randomly select half of the subjects to build the training set, while the sequences of the remaining subjects are used for test.

The training set has been used to learn a codebook of frequent sequential patterns, and to train one vs one χ^2 kernel SVMs with C equals to 10. In our modified apriori algorithm we set minimal and maximal pattern length respectively to $\tau = 3$ and $K_M = 30$. The similarity threshold ϵ used to establish a match between pattern candidates and time windows was set to 0.9, while the threshold ψ was set to 75.

4.1 Results

We performed experiments to test the quality of the codebook of frequent sequential patterns generated via Algorithm 1. On average, our codebook has a size of 120 ± 14.72 patterns. The average accuracy value over 10 runs is approximately of about 88.32%. This result is very encouraging considering that the action representation is very compact.

As detailed in Sect. 3.1, the codebook stores all patterns with a frequency count greater than ψ that do not contribute to the generation of longer patterns. However, since we adopt an approximate matching strategy, the frequency count of the generated patterns is not a very reliable measure of the importance of the learned patterns. Hence, it is reasonable to wonder if patterns that are considered infrequent during the codebook learning procedure might actually improve

T vs P	SS	D	PR	G	W	S	B	T	H	C	BB	K
SS	**80.23**	0	0	0	0.43	4.18	0.04	0.90	v4.59	1.07	7.63	0.93
D	0	**99.96**	0	0	0	0	0.04	0	0	0	0	0
PR	0.04	0	**96.35**	0	0.73	1.42	0	0.24	0.12	1.09	0	0
G	0.12	0	0	**93.14**	1.00	1.66	0	0	3.12	0.48	0.48	0
W	0.42	0	1.24	0.09	**92.43**	1.18	0	0.10	0	2.00	2.54	0
S	0.59	0	0.07	0.11	0.30	**93.76**	0.04	0.04	0.12	2.28	2.67	0
B	0	4.38	0	0	0	0.20	**95.15**	0.04	0	0.03	0	0.19
T	0.04	0	0.08	0	0.04	0.81	0.44	**93.10**	0	1.42	0.04	4.03
H	2.74	0	0.04	5.35	0.12	1.28	0	0	**89.00**	0.04	1.42	0
C	0.08	0	0.04	0.08	0.28	3.31	0	0	0.04	**95.77**	0.40	0
BB	3.19	0.62	0.08	0.41	3.89	6.22	0	0.15	1.61	2.47	**81.33**	0.04
K	0.20	0.07	0	0	0.04	0.24	0.30	0.35	0	0.49	0	**98.30**

action classification. To validate our hypothesis, we also included in the codebook sequential patterns that are pruned in line 11 of Algorithm 1 and having a frequency count greater than a threshold ϕ. Then, we study how frequent a frequent pattern should be for being included in the codebook by studying how the average recall changes when varying ϕ in the range $[0, 100]$.

Figure 4(a) shows the trend of the average per-class recall over 10 runs when varying ϕ. Vertical bars represent standard deviations of recall values. Figure 4(b) shows the number of patterns in the codebook with a frequency greater than ϕ. As shown in the latter plot, the codebook size decreases exponentially; on average, the codebook size ranges between 50583 (when $\phi = 0$, i.e. all infrequent patterns are included in the codebook) and 44 (when $\phi = 100$).

On the other hand, as shown in Fig. 4(a), there is an increase of the recall values for growing values of ϕ. For value of ϕ in [20–70] there is a very limited variation of the average recall; what it really changes is the codebook size that affects the complexity of the vector quantization step. The best average per-class recall is obtained for $\phi = 40$ and is of about 92.38% $\pm$ 0.97. The corresponding

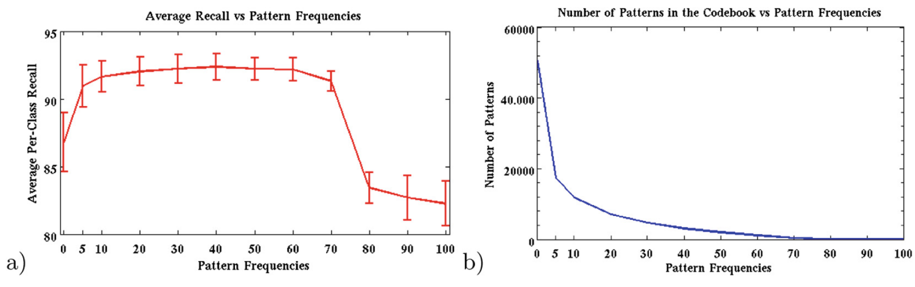

Fig. 4. Plots in (a) and (b) show how the average per-class recall and the number of patterns in the codebook, respectively, change by varying the minimal pattern frequency. Values are averaged over 10 runs, and vertical bars show standard deviations.

codebook size is of about 3086. For $\phi = 70$, the average recall is of about 91.31% and the codebook size is on average 400. For $\phi > 70$, the recall value decreases, however the information embedded in very frequent patterns is still very high considering that, with only 44 codewords (on average) with $\phi = 100$, the method achieves an average recall of about 82.26%.

Experimental results shows the confusion matrix obtained with our technique averaged over 10 runs when $\phi = 40$. Columns of the table represents predicted class labels while rows represent true class labels. As shown in the table, most of the confusion is between the action classes *Start System (SS)* and *Beat both (BB)*, *Had enough (H)* and *Goggles(G)*, *Beat both (BB)* and *Shoot (S)*. We stress here that our technique has been tested directly on the 3D joints coordinates and the only preprocessing of the sequences consists of making them zero mean. Since the method is very general, we believe that the use of more complex features extracted from skeletal data might result in higher value of the average recall.

We compare our method against the work in [7] on equal terms of experimental protocol. In [7], a pyramid of covariance matrices of 3D joints coordinates is used to represent a sequence of skeletons: the root node encode information about the entire sequence; at lower levels, sequences of covariance matrices calculated by a sliding window approach are considered. Action classification is performed by linear SVM. The work only reports the average correct classification rate or accuracy value averaged over 10 runs in different configuration, and achieves the best accuracy value of about 91.7%. Our accuracy value is of about 92.31% at $\phi = 40$, which is slightly superior to the one of [7].

5 Conclusions and Future Work

In this paper we demonstrate the idea of representing sequences of skeletons by means of distributions of frequent patterns. In our framework, frequent sequential patterns are computed by means of a modified version of the apriori algorithm. At each iteration, all frequent patterns that cannot be used for generating longer patterns are stored and used as codewords. This approach yields to a codebook of patterns of different length.

To encode the data, at each frame, we use a temporal window whose length adapts to the length of the pattern. Then, the most similar pattern is found and the histogram is updated accordingly.

One question we have tried to answer in our experiments is how frequent our frequent patterns have to be. Our experiments show that the method benefits from ignoring infrequent patterns both in terms of recall and computational complexity, since a more compact sequence description can be obtained with a smaller codebook. However, considering only the most frequent patterns may result in a lost of details of the action representation and, hence, might have a negative impact on the performance of the method.

We presented preliminary results by validating our method on skeletal data. On the MSRC-12 dataset our method achieves state-of-the-art accuracy values. In future work, we will extensively study the effect of varying some parameters, such as ϵ and ψ, on the performance of the method. The main limitation of our method is that it might not be able to cope with varying execution velocity of the action, which also depends on the subject. Therefore, we also plan to extend our formulation by accounting for the misalignments between patterns and matched temporal window in order to improve the learning of sequential patterns.

Acknowledgement. We are grateful to Mr. Giovanni Caruana for making available his implementation of the classic apriori algorithm, which he implemented in his Master thesis work at University of Palermo.

References

1. Agrawal, R., Imieliński, T., Swami, A.: Mining association rules between sets of items in large databases. In: ACM SIGMOD record, vol. 22. no. 2. ACM (1993)
2. Agrawal, R., Srikant, R.: Mining sequential patterns. In: Proceedings of the Eleventh International Conference on Data Engineering. IEEE (1995)
3. Dalal, N., Triggs, B.: Histograms of oriented gradients for human detection. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), vol. 1. IEEE (2005)
4. Demirdjian, D., Wang, S.: Recognition of temporal events using multiscale bags of features. In: IEEE Workshop on Computational Intelligence for Visual Intelligence (CIVI). IEEE (2009)
5. Fei-Fei, L., Perona, P.: A bayesian hierarchical model for learning natural scene categories. In: IEEE Proceedings of Conference on Computer Vision and Pattern Recognition (CVPR), vol. 2. IEEE (2005)
6. Gavrilov, Z., Sclaroff, S., Neidle, C., Dickinson, S.: Detecting reduplication in videos of american sign language. In: Proceedings of Eighth International Conference on Language Resources and Evaluation (LREC), Instanbul, Turkey, May 2012
7. Hussein, M.E., et al.: Human action recognition using a temporal hierarchy of covariance descriptors on 3D joint locations. In: IJCAI, vol. 13 (2013)
8. Karaman, S., et al.: L1-regularized logistic regression stacking and transductive CRF smoothing for action recognition in video. In: ICCV workshop on action recognition with a large number of classes, vol. 13 (2013)
9. Laptev, I., et al.: Learning realistic human actions from movies. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR). IEEE (2008)
10. Laxman, S., Sastry, P.S.: A survey of temporal data mining. Sadhana **31**(2), 173–198 (2006)
11. Presti, L.L., La Cascia, M.: 3D skeleton-based human action classification: a survey. Pattern Recogn. **53**, 130–147 (2016)
12. Murthy, O.V., Goecke, R.: Ordered trajectories for large scale human action recognition. In: Proceedings of the IEEE International Conference on Computer Vision Workshops (2013)
13. Niebles, J.C., Wang, H., Fei-Fei, L.: Unsupervised learning of human action categories using spatial-temporal words. Int. J. Comput. Vis. **79**(3), 299–318 (2008)
14. Peng, X., et al.: Bag of visual words and fusion methods for action recognition: comprehensive study and good practice. Comput. Vis. Image Underst. **150**, 109–125 (2016)
15. Peng, X., et al.: Exploring motion boundary based sampling and spatial-temporal context descriptors for action recognition. In: British Machine Vision Conference (BMVC) (2013)

16. Schuldt, C., Laptev, I., Caputo, B.: Recognizing human actions: a local SVM approach. In: Proceedings of the 17th International Conference on Pattern Recognition (ICPR), vol. 3. IEEE (2004)
17. Wang, H., et al.: Dense trajectories and motion boundary descriptors for action recognition. Int. J. Comput. Vis. **103**(1), 60–79 (2013)
18. Fothergill, S., et al.: Instructing people for training gestural interactive systems. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems. ACM (2012)
19. Shotton, J., et al.: Real-time human pose recognition in parts from single depth images. Commun. ACM **56**(1), 116–124 (2013)
20. Wang, C., Wang, Y., Yuille, A.L.: An approach to pose-based action recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (2013)
21. Wang, J., et al.: Mining actionlet ensemble for action recognition with depth cameras. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR). IEEE (2012)
22. Zhao, X., et al.: Online human gesture recognition from motion data streams. In: Proceedings of the 21st ACM international conference on Multimedia. ACM (2013)
23. Zhu, Y., Zhao, X., Fu, Y., Liu, Y.: Sparse coding on local spatial-temporal volumes for human action recognition. In: Kimmel, R., Klette, R., Sugimoto, A. (eds.) ACCV 2010. LNCS, vol. 6493, pp. 660–671. Springer, Heidelberg (2011). doi:10.1007/978-3-642-19309-5_51