Detecting Frequent Patterns in Video using Partly Locality Sensitive Hashing

Koichi Ogawara, Yasufumi Tanabe, Ryo Kurazume, and Tsutomu Hasegawa

Kyushu University

Abstract. Frequent patterns in video are useful clues to learn previously unknown events in an unsupervised way. This paper presents a novel method for detecting relatively long variable-length frequent patterns in video efficiently. The major contribution of the paper is that Partly Locality Sensitive Hashing (PLSH) is proposed as a sparse sampling method to detect frequent patterns faster than the conventional method with LSH. The proposed method was evaluated by detecting frequent everyday whole body motions in video.

1 Introduction

Detection of previously learned human actions in video [1–3] or detection of irregular, i.e. not learned, human actions in video [4, 5] have been an active research topic because they can be applied to various tasks including content-based retrieval from video, surveillance, etc.

For this, a human action database is usually built in advance in a supervised way by manually annotating a large collection of videos. To accelerate this process, several methods have been proposed to detect frequent human actions in video based on frequent data mining techniques [6–9]. This line of research can be grouped into 2 approaches. In the first approach, a large number of spatio-temporal patches are extracted from video and are classified into different actions [10, 11], however it lacks the ability to detect relatively long actions. In the second approach, dynamic programming is employed to detect relatively long actions [12], however the computational complexity becomes $O(N^2)$ where N is the length of a video, thus it is not appropriate to deal with a long video.

To decrease the computational time, Meng et al. proposed a method that finds data similar to the data at t from the entire input time series by Locality Sensitive Hashing (LSH) [13] for all t and connects them along the time axis so as to detect frequent actions from a motion capture data in $O(N^{1+1/\alpha})$ [14]. The problem is that there is a large overlap between the found data at successive times, thus the search is redundant.

In this paper, we propose a novel method for detecting frequent patterns in video efficiently where the redundancy in [14] is resolved. In the proposed method, detected nearby data in the first frame are maintained by a linked list and are updated along the time axis. However, dynamic change of nearby data is not handled correctly, thus the linked list is modified based on a small number of data sparsely sampled in each frame, whereas nearby data are densely sampled in every frame in [14] which results in a large computational time.

As a sparse sampling method, Partly Locality Sensitive Hashing (PLSH) is proposed which is an extension to LSH. Experimental results show that the proposed method can detect frequent patterns in video much faster than the conventional method with LSH.

2 Overview of the proposed method

The purpose of this study is to detect unknown frequent patterns appeared in a d-dimensional time series. Frequent patterns are a set of subsequences of a time series where minor variation of shape and length is allowed.

Fig. 1 shows a 2D slice of a d-dimensional time series. If a data point $\boldsymbol{o}(t)$ observed at t is on a frequent pattern, many similar shaped patterns exist at around $\boldsymbol{o}(t)$. So, if a sequence of data has many other data in its neighborhood, it can be considered as a good candidate for a frequent pattern.

Here, terminology is defined as follows. "Neighborhood" is defined as the inside of a hyper sphere of radius R in d-dimensional space. "Segment" is defined as a subsequence of a time series bounded by a hyper sphere. "Data density" is defined as the total length of segments in a hyper sphere. Then, the data density at t is calculated as

$$D(t) = \sum_{i \in S(t)} \|\boldsymbol{o}(i) - \boldsymbol{o}(i+1)\| \qquad (1)$$
$$\text{where } S(t) = \{i; \|o(i) - o(t)\| \leq R\}.$$

Data density is used to evaluate the existence of frequent patterns in each frame. However, it takes quadratic time in total to calculate data density exactly in all frames even if a tree-like data structure is used.

To overcome this problem, data density is calculated efficiently by an algorithm outlined in Table 1 using the proposed approximate nearest neighbor search scheme named PLSH.

At $t = 1$, a linked list is initialized by checking all N data so that each element in the linked list holds the endpoints, i.e. the start time (st) and the end time (ed), of all the segments as in Fig. 1 (a).

From $t = 2$ to N, the linked list is updated along the time axis. Firstly, the endpoints of the segments found in the previous frame are shifted so that they lie on the boundary of the current hyper sphere as in Fig. 1(b). The amount of shift between neighboring frames is usually very small, so the time taken to update the linked list can be considered as a constant when the total number of elements in the linked list is limited. When a segment goes out of the sphere, it is removed from the linked list. When 2 segments are connected together, the corresponding elements in the linked list are merged as in Fig. 1 (c).

However, newly appeared segments cannot be detected immediately by the update method above, so they have to be found and added to the linked list

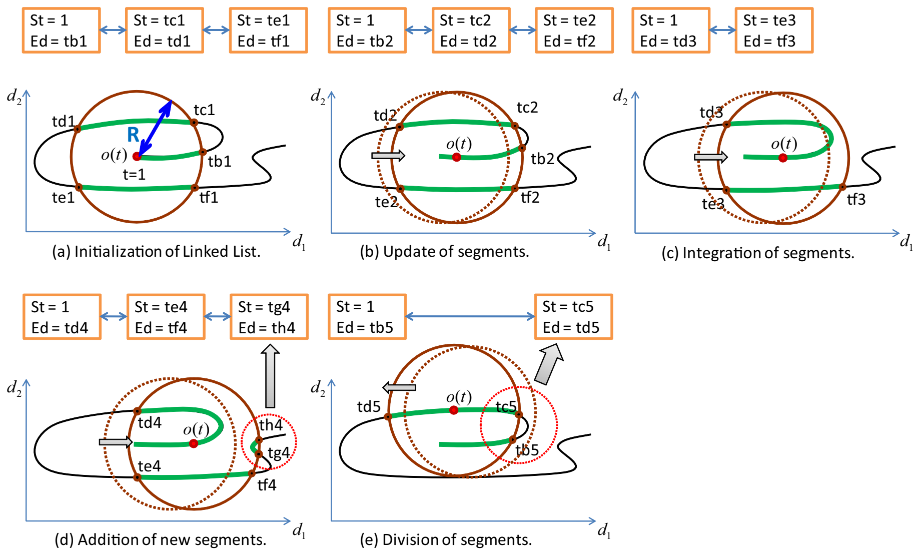


Fig. 1. Calculation of data density

separately as in Fig. 1 (d). For this, instead of checking all N data which results in quadratic time in total, nearby data are sparsely sampled in the proposed method. If a sample is inside the hyper sphere and is not included in the linked list yet, the segment including that sample are added to the linked list. A new segment is not necessarily detected at the time it enters into the hyper sphere, because it will be detected in the subsequent frames if it is sufficiently similar to the segment at t. When a new segment is detected later, the data density in the past can be modified at that time.

Then, it is important to select nearby data sparsely and efficiently. To achieve this, Partly Locality Sensitive Hashing (PLSH) is proposed in Section 3.

Another problem is that divided segments cannot be detected immediately by the update method above, so they also have to be found separately as in Fig. 1 (e). For this, nearby data at around $t - T_{\text{delay}}$ are sparsely sampled in the proposed method. If a sample is outside the current hyper sphere and is included in the linked list, the segment including that sample are divided. In this case, a divided segment is not necessarily detected at the time the hyper sphere divides it, because it will be detected in the subsequent frames if division is persistent. When a divided segment is detected later, the data density in the past can be modified at that time.

Finally, frequent patterns are detected by a global optimization method using Dynamic Programming as explained in Section 4.

Table 1. Algorithm to find frequent patterns

```
1. At $t = 1$:
    Initialize a linked list that maintains segments in the hyper sphere. (Fig. 1(a))
2. From $t = 2$ to $N$:
    Update the linked list. (Fig. 1(b),(c))
    Detect new segments by PLSH and modify the linked list. (Fig. 1(d))
    Detect divided segments by PLSH and modify the linked list. (Fig. 1(e))
3. From $t = 1$ to $N$:
    Detect frequent patterns by a global optimization method.
```

3 Partly Locality Sensitive Hashing

Partly Locality Sensitive Hashing (PLSH) is an extension to Locality Sensitive Hashing (LSH) [13] which is an approximate nearest neighbor search algorithm.

3.1 Combining locality sensitive and insensitive hash functions

In PLSH, a set of hash functions $g_l(\boldsymbol{p})$ $(1 \leq l \leq L)$ are defined as

$$g_l(\boldsymbol{p}) = < hs_{l,1}(\boldsymbol{p}), \ldots, hs_{l,K_s}(\boldsymbol{p}), hi_{l,1}(\boldsymbol{p}), \ldots, hi_{l,K_i}(\boldsymbol{p}) > .$$

where $hs_{l,k}(\boldsymbol{p})$ is an arbitrary locality sensitive hash function $hs_{l,k} : R^d \rightarrow U$, while $hi_{l,k}(\boldsymbol{p})$ is an arbitrary locality insensitive hash function $hi_{l,k} : R^d \rightarrow U$. In a projection-based scheme as in Fig. 2, these functions can be defined as

$$\begin{aligned} hs_{l,k}(\boldsymbol{p}) &= \lfloor (\boldsymbol{a}_{sl,k} \cdot \boldsymbol{p} + b_{sl,k})/w_{sl,k} \rfloor, \\ hi_{l,k}(\boldsymbol{p}) &= \lfloor (\boldsymbol{a}_{il,k} \cdot \boldsymbol{p} + b_{il,k}) \rfloor \bmod w_{il,k}, \end{aligned}$$

where $\boldsymbol{a}, b$ are randomly chosen to satisfy $\boldsymbol{a} \in R^d, ||\boldsymbol{a}|| = 1, 0 \leq b < w$ for each hash function.

Approximate nearest neighbor search proceeds as follows. Firstly, L hash values are calculated for all N data by applying L hash functions and they are stored in the corresponding hash buckets in L hash spaces. Given an input query $\boldsymbol{p}$, L hash values are calculated in the same way and the data in the corresponding hash buckets in L hash spaces are examined.

3.2 Sparse sampling using PLSH

LSH is not appropriate to select nearby data inside a hyper sphere sparsely and efficiently because the data in a hash bucket are densely examined as in Fig. 3(a) [14].

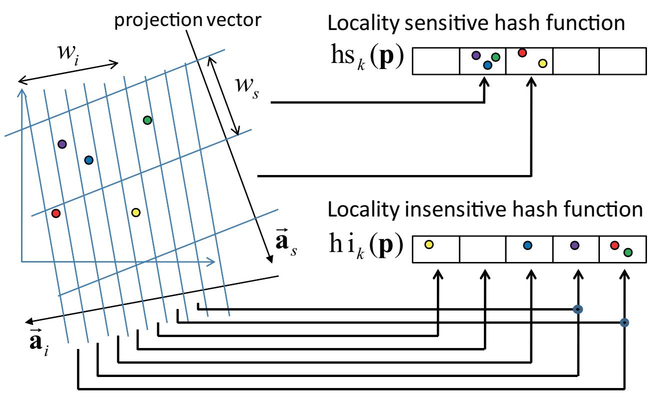


Fig. 2. Partly locality sensitive hash functions

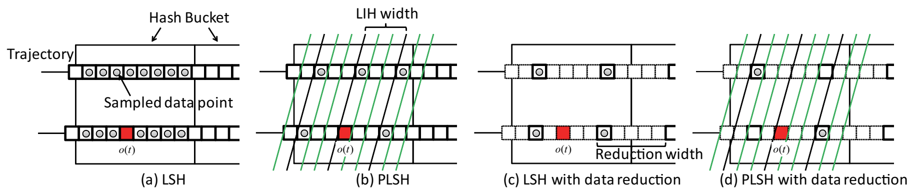


Fig. 3. Difference between LSH and PLSH

PLSH is useful in this case. In PLSH, an input data point is defined as $d+1$ dimensional vector $(p_1, \ldots, p_d, t)^T$ where the first d elements represent an original data point and the last 1 element represents the time when the data point is observed.

A hash function $g_l(\boldsymbol{p})$ is composed of K_s locality sensitive hash functions and a single locality insensitive hash function. As for the locality sensitive hash functions, the last value of $\boldsymbol{a}_s$ is always 0 and w_s is R. As for the locality insensitive hash function, the first d values of $\boldsymbol{a}_i$ is 0 and the last value is 1 so that data points in neighboring frames never collide in a same hash bucket. This means the data are scattered to w_i (= LIH width) hash spaces.

In this way, data are selected from different and independent hash spaces in neighboring frames as in Fig. 3(b). The number of data in a hash bucket is reduced by $\frac{1}{w_i}$ and the computational time is reduced similarly without affecting the probability of detecting new segments or divided segments because all data are examined anyway.

In LSH, similar reduction rate of $\frac{1}{w_r}$ can be achieved by sampling a time series in w_r intervals (= Reduction width) beforehand as in Fig. 3 (c). However, reduction rate of $\frac{1}{w_i \cdot w_r}$ is achieved in PLSH by combining data reduction ($\frac{1}{w_r}$) and data scattering ($\frac{1}{w_i}$) as in Fig. 3 (d).

4 Detection of frequent patterns

Given a time series $\boldsymbol{O} = (\boldsymbol{o}(1), \cdots, \boldsymbol{o}(N))$, frequent patterns are detected in 2 steps:

1. Detection of frequent patterns using data density
2. Classification of frequent patterns

4.1 Detection of frequent patterns using data density

Firstly, frequent patterns are detected from $\boldsymbol{O}$ irrespective of types of patterns. This problem is formulated as a combinatorial optimization problem in that each frame is assigned binary labels as $\boldsymbol{X} = (x_1, \cdots, x_N)$ where $x_t \in \{1 = \text{frequent pattern}, 0 = \text{not frequent pattern}\}$. This problem is re-formulated as to find $\boldsymbol{X}$ that minimizes the energy function defined as

$$E(\boldsymbol{O}, \boldsymbol{X}) = E_v(\boldsymbol{O}, \boldsymbol{X}) + E_d(\boldsymbol{O}, \boldsymbol{X}) + E_s(\boldsymbol{X}). \quad (2)$$

The energy function is composed of 3 terms: velocity term, data density term and smoothing term.

Velocity term $E_v(\boldsymbol{O}, \boldsymbol{X})$ penalizes data points with small velocity and is defined as

$$E_v(\boldsymbol{O}, \boldsymbol{X}) = \sum_t -\log(1 - \exp(-\frac{|\dot{\boldsymbol{o}}_{x_t}(t)|}{< |\dot{\boldsymbol{o}}_{x_t}(t)| >}))$$

where $< |\dot{\boldsymbol{o}}_{x_t}(t)| >$ is the mean value of $|\dot{\boldsymbol{o}}_{x_t}(t)|$.

Data density term $E_d(\boldsymbol{O}, \boldsymbol{X})$ penalizes data points with small data density and is defined as

$$E_d(\boldsymbol{O}, \boldsymbol{X}) = \sum_t -\log(1 - \exp(-\frac{D_{x_t}(t)}{< D_{x_t}(t) >}))$$

where $< D_{x_t}(t) >$ is the mean value of $D_{x_t}(t)$.

Smoothing term $E_s(\boldsymbol{X})$ penalizes different neighboring labels to reject short patterns and is defined as

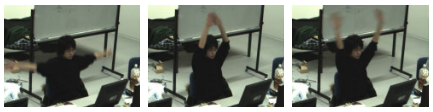
$$E_s(\boldsymbol{X}) = \sum_t T(x_t \neq x_{t+1}) \cdot C_{\text{smooth}}$$

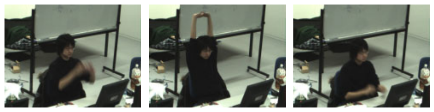
where C_{smooth} is a constant and $T(s)$ is defined as $T(\text{true}) = 1, T(\text{false}) = 0$.

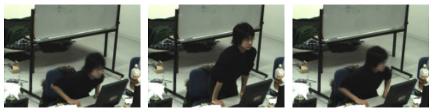
Because all the terms in Eq. (2) satisfies the first order Markovian property, the energy function can be minimized analytically by Dynamic Programming.

4.2 Classification of frequent patterns

Because different types of actions are mixed in the detected frequent patterns, an agglomerative clustering technique is applied to classify them by iteratively grouping 2 patterns together whose average distance is smaller than the radius R of the hyper sphere.


(a) Bye


(b) Stretch


(c) Stand-up

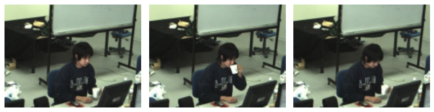
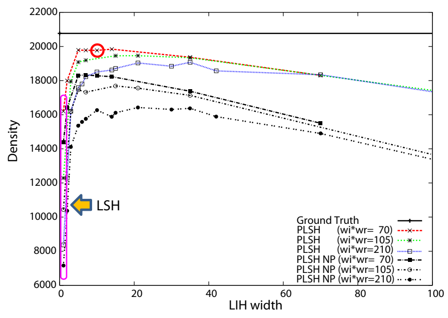
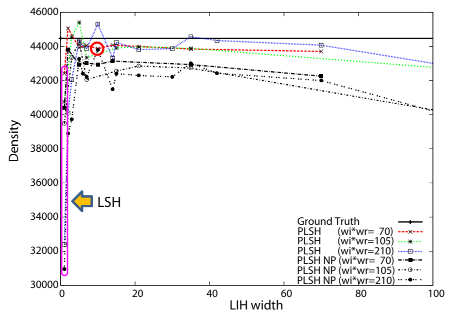

(d) Drink

Fig. 4. 4 whole body motions



(a) Video 1



(b) Video 2

Fig. 5. Estimation of data density

5 Experimental results

The proposed method was evaluated by detecting frequent whole body motions in video. To evaluate the sparse sampling methods for detecting new segments and divided segments, PLSH was compared with LSH. L is fixed to 8 and K_s is fixed to 3 in both PLSH and LSH for all the experiments. All the experiments were performed on a Xeon 3.0GHz PC.

2 videos were prepared that contain several of 4 types of whole body motions as in Fig. 4. Video 1 contains 5 Byes, 6 Stretches and 5 Stand-ups. Video 2 contains 7 Byes, 7 Stretches, 8 Drinks and 4 noisy motions. Cubic Higher-order Local Auto Correlation (CHLAC) [15] is used as an image descriptor to represent patterns in video in a position invariant way.

5.1 Evaluation of sampling methods in data density estimation

As described in Section 3.2, the sparse sampling method using PLSH achieves reduction rate of $\frac{1}{w_i \cdot w_r}$ by combining data reduction ($\frac{1}{w_r}$) and data scattering ($\frac{1}{w_i}$).

Table 2. Evaluation of video 1 [2700 frames]

Action	Bye	Stretch	Stand-up	False	False	Precision	Recall	Time
Presented #	5	6	5	Positive	Negative			[msec]
(1)LSH (w_r=1)	5.00	5.00	5.00	0.00	1.00	1.00	0.94	5807
(2)LSH (w_r=15)	5.00	5.00	5.00	0.00	1.00	1.00	0.94	512
(3)LSH (w_r=70)	5.00	4.90	0.00	0.00	6.10	1.00	0.62	198
(4)PLSH	5.00	5.00	5.00	0.00	1.00	1.00	0.94	224

Table 3. Evaluation of video 2 [3600 frames]

Action	Bye	Stretch	Drink	False	False	Precision	Recall	Time
Presented #	7	7	8	Positive	Negative			[msec]
(1)LSH (w_r=1)	7.00	6.00	8.00	1.00	1.00	0.95	0.95	13499
(2)LSH (w_r=15)	7.00	4.00	8.00	1.00	3.00	0.95	0.86	1134
(3)LSH (w_r=70)	4.00	0.00	7.20	0.30	10.80	0.97	0.51	338
(4)PLSH	7.00	3.00	8.00	1.00	4.00	0.95	0.82	482

Because the computational time is roughly proportional to $\frac{1}{w_i \cdot w_r}$, the optimal w_i, w_r that does not degrade the detection rate of frequent patterns, i.e. accuracy in data density estimation, should be determined.

Fig. 5 shows the accumulated data density for different pairs of w_i, w_r. The vertical axis represents the accumulated data density and the horizontal axis represents w_i. The polygonal lines in the figure connect the pairs where $\frac{1}{w_i \cdot w_r}$ is the same. NP (Non Propagation) means data density in the past is not modified when new segments or divided segments are detected.

In Fig. 5 (b), data density exceeds the ground truth for some pairs. This is because divided segments were not detected well in those parameters.

The left most point in each polygonal line, i.e. w_i is 1 which means data are not scattered, represents the case where LSH is applied. From the figure, we can see that data density is estimated better when w_i increases from 1 when $\frac{1}{w_i \cdot w_r}$ is a constant, however data density moves away from the ground truth when w_i increases further. This is because data are not sampled sufficiently when one of w_i and w_r takes a large value that decreases detection rate of frequent patterns.

The red circles in Fig. 5 (a) and (b), i.e. w_i is 10 and w_r is 7, were chosen as the best parameters and were used in the following experiments.

5.2 Detection rate of frequent patterns

4 different sampling methods were evaluated: (1) LSH ($w_r = 1$, no data reduction), (2) LSH ($w_r = 15$, experimentally chosen to achieve the same accumulated data density as PLSH), (3) LSH ($w_r = 70$, same reduction rate as PLSH), (4) PLSH ($w_i \cdot w_r = 70$). The detection rate of frequent patterns were summarized in

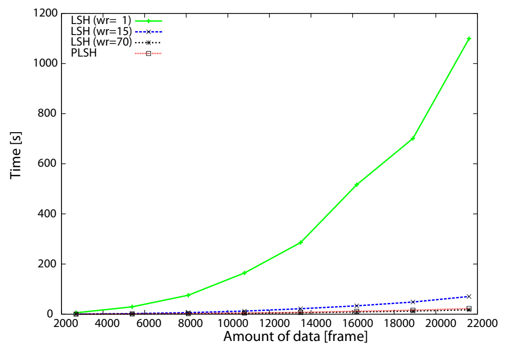


Fig. 6. Computational time v.s. amount of data

Table 2 and Table 3. To evaluate the results, Precision and Recall were calculated as

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}, \quad \text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

where TP means True Positive, FP means False Positive and FN means False Negative. Each experiment was evaluated 10 times and the average is shown because the parameters of hash functions were determined randomly,

Similar detection rate to PLSH was achieved with (1) and (2), but the computational time became longer because the reduction rate was worse. On the other hand, the detection rate with (3) was considerably bad while the computational time was slightly better. From these results, we see that PLSH can be better in both computational time and detection rate than LSH.

5.3 Computational time v.s. amount of data

10 videos of different length were generated by simply concatenating the video in Table 2 with Gaussian noise added. The same 4 methods were applied to these videos and the results are shown in Fig. 6.

The figure shows the computational time is drastically reduced by data reduction and data scattering of PLSH because the number of data in a hash bucket is reduced.

6 Conclusions

This paper presents a method for detecting relatively long variable-length frequent patterns efficiently from video.

Partly Locality Sensitive Hashing (PLSH) is proposed by combining locality sensitive hash functions and locality insensitive hash functions. Data reduction and data scattering of PLSH enables faster and much accurate detection of frequent patterns than the conventional method with LSH.

Acknowledgement.

This study was supported in part by Program for Improvement of Research Environment for Young Researchers from Special Coordination Funds for Promoting Science and Technology (SCF) commissioned by the MEXT of Japan, and in part by Grant-in-Aid for Young Scientists (B)(21700224).

References

1. Schuldt, C., Laptev, I., Caputo, B.: Recognizing human actions: a local svm approach. In: Proc. of ICPR. (2004)
2. Ke, Y., Sukthankar, R., Hebert, M.: Efficient visual event detection using volumetric features. In: Proc. of ICCV. (2005)
3. Niebles, J.C., null Li Fei-Fei: A hierarchical model of shape and appearance for human action classification. In: Proc. of IEEE Conference on Computer Vision and Pattern Recognition: CVPR. (2007) 1–8
4. Zhong, H., Shi, J., Visontai, M.: Detecting unusual activity in video. In: Proc. of IEEE Conference on Computer Vision and Pattern Recognition: CVPR. (2004) 819–826
5. Boiman, O., Irani, M.: Detecting irregularities in images and in video. International Journal of Computer Vision: IJCV **74** (2007) 17–31
6. Staden, R.: Methods for discovering novel motifs in nucleic acid sequences. Computer Applications in the Biosciences **5** (1989) 293–298
7. Lin, J., Keogh, E., Lonardi, S., Patel, P.: Finding motifs in time series. In: Proc. of the 2nd Workshop on Temporal Data Mining. (2002) 53–68
8. Yankov, D., Keogh, E., Medina, J., Chiu, B., Zordan, V.: Detecting time series motifs under uniform scaling. In: Proc. of the 13th ACM KDD Intl. Conf. on Knowledge Discovery and Data Mining. (2007) 844–853
9. Mueen, A., Keogh, E., Zhu, Q., Cash, S., Westover, B.: Exact discovery of time series motifs. In: Proc. of 2009 SIAM International Conference on Data Mining: SDM. (2009) 1–12
10. Niebles, J.C., Wang, H., Fei-Fei, L.: Unsupervised learning of human action categories using spatial-temporal words. In: Proc. of BMVC. (2006)
11. Kitani, K.M., Okabe, T., Sato, Y., Sugimoto, A.: Discovering primitive action categories by leveraging relevant visual context. In: Proc. Int. Workshop on Visual Surveillance (in conjunction with ECCV2008). (2008)
12. Uchida, S., Mori, A., Kurazume, R., Taniguchi, R., Hasegawa, T.: Logical dp matching for detecting similar subsequence. In: Proc. of Asian Conference of Computer Vision. (2007)
13. Datar, M., Immorlica, N., Indyk, P., Mirrokni, V.: Locality-sensitive hashing scheme based on p-stable distributions. In: Proc. of the twentieth annual Symposium on Computational Geometry. (2004) 253–262
14. Meng, J., Yuan, J., Hans, M., Wu, Y.: Mining motifs from human motion. In: Proc. of EUROGRAPHICS'08. (2008)
15. Kobayashi, T., Otsu, N.: Action and simultaneous multiple-person identification using cubic higher-order local auto-correlation. In: Proc. Int. Conference on Pattern Recognition: ICPR. (2004) 741–744