The 2010 IEEE/RSJ International Conference on
Intelligent Robots and Systems
October 18-22, 2010, Taipei, Taiwan

Detecting Repeated Patterns using Partly Locality Sensitive Hashing

Koichi Ogawara and Yasufumi Tanabe and Ryo Kurazume and Tsutomu Hasegawa

***Abstract*— Repeated patterns are useful clues to learn previously unknown events in an unsupervised way. This paper presents a novel method that detects relatively long variable-length unknown repeated patterns in a motion sequence efficiently.**

The major contribution of the paper is two-fold: (1) Partly Locality Sensitive Hashing (PLSH) [1] is employed to find repeated patterns efficiently and (2) the problem of finding consecutive time frames that have a large number of repeated patterns is formulated as a combinatorial optimization problem which is solved via Dynamic Programming (DP) in polynomial time $O(N^{1+1/\alpha})$ thanks to PLSH where N is the total amount of data. The proposed method was evaluated by detecting repeated interactions between objects in everyday manipulation tasks and outperformed previous methods in terms of accuracy or computational time.

I. INTRODUCTION

Recognizing human activity is one of the fundamental techniques in order to assist humans in everyday environment using robotics technology. Many researchers have designed a variety of task-dependent recognizers that identify actions as well as capture the necessary parameters to describe them [2], [3], [4]. However, there are huge variety of human activities in everyday life, it is not practical to prepare recognizers for all of them. Thus, a mechanism to learn new knowledge, new recognizers, in an unsupervised way is required.

The goal of this study is to develop an efficient method that detects previously unknown repeated patterns in an observation without task-dependent knowledge as a basis to realize the above mentioned mechanism. The basic idea behind this is if a particular pattern appears many times in a long-term observation, this pattern must be meaningful to a user or to a task. These patterns are good candidates for important human actions and can be used to train recognizers, to learn personal habits, to predict a user's next action, etc.

The major contribution of the paper is two-fold. The first contribution is that Partly Locality Sensitive Hashing (PLSH) [1] is employed to find repeated patterns efficiently in a framework of approximate nearest neighbor search. The second contribution is that the problem of finding variable-length repeated patterns is formulated as a combinatorial optimization problem which is solved via a global optimization framework using Dynamic Programming (DP) in polynomial time $O(N^{1+1/\alpha})$ thanks to PLSH where N is the total amount of data.

K. Ogawara is with Institute for Advanced Study, Kyushu University, Fukuoka, JAPAN ogawara@ait.kyushu-u.ac.jp

Y. Tanabe is with Department of Electrical Engineering and Computer Science, Kyushu University, Fukuoka, JAPAN

R. Kurazume and T. Hasegawa are with Faculty of Information Science and Electrical Engineering, Kyushu University, Fukuoka, JAPAN

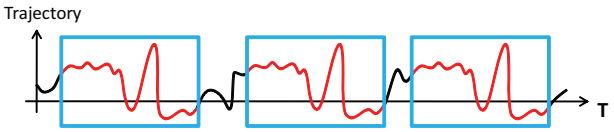


Fig. 1. Repeated patterns in a time series data.

The remainder of the paper is organized as follows. A survey of the related research is given in Section II and an overview of the algorithm is presented in Section III. PLSH is explained in Section IV and the global optimization framework is explained in Section V. The experimental results are presented in Section VI and this paper concludes in Section VII.

II. RELATED WORK

While there has been a large body of work regarding efficiently locating previously known patterns in a time series data [5], [6], we focus on locating previously unknown repeated patterns in this study as shown in Fig. 1. Finding unknown repeated patterns, or motifs, has been a well-known task in the bioinformatics community [7], in the data mining community [8], [9], [10] and in the motion analysis community [11], [12].

In contrast to finding known-length repeated patterns [7], [8], [10], several approaches based on Dynamic Programming (DP) have been proposed to deal with unknown-length repeated patterns in quadratic time [11].

In our previous work [13], we proposed a method that finds repeated patterns in $O(N \log N)$ time by counting the number of repeated patterns exactly using kD-tree, however the hidden constant becomes linear with respect to N in the worst case that increases the computational time.

To reduce computational time, Meng et al. proposed a method that finds K near neighbor data points using Locality Sensitive Hashing (LSH) at each time frame and connects them along time axis so as to find consistent repeated patterns in $O(N^{1+1/\alpha})$ time [12]. However, when the total amount of data becomes large, the number of data in each hash bucket also becomes large. Thus, the actual computational time also becomes large.

Tanabe et al. proposed Partly Locality Sensitive Hashing (PLSH), an extension to LSH, to find repeated patterns much efficiently[1]. The proposed method extends [1] to work with newly defined *pattern density* in order to deal with repeated patterns robustly.

III. Overview of the Algorithm

Fig. 1 shows an example of the problem of interest. Given a long time series data of d dimensions, the task is to find unknown repeated patterns efficiently where minor variation of shape and length is allowed.

Fig. 2 shows a 2D slice of a time series data. If a data point $\boldsymbol{o}(t)$ at time t belongs to a set of repeated patterns, many similar shaped patterns exist at around the data point. So, a sequence of data points that have many other data points in their neighborhood can be considered as a good candidate for a repeated pattern.

To find candidate repeated patterns, *neighborhood* is defined as the inside of a hyper sphere of radius R in d dimensional space and *pattern density*, the length of the segments in the sphere, is defined as

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

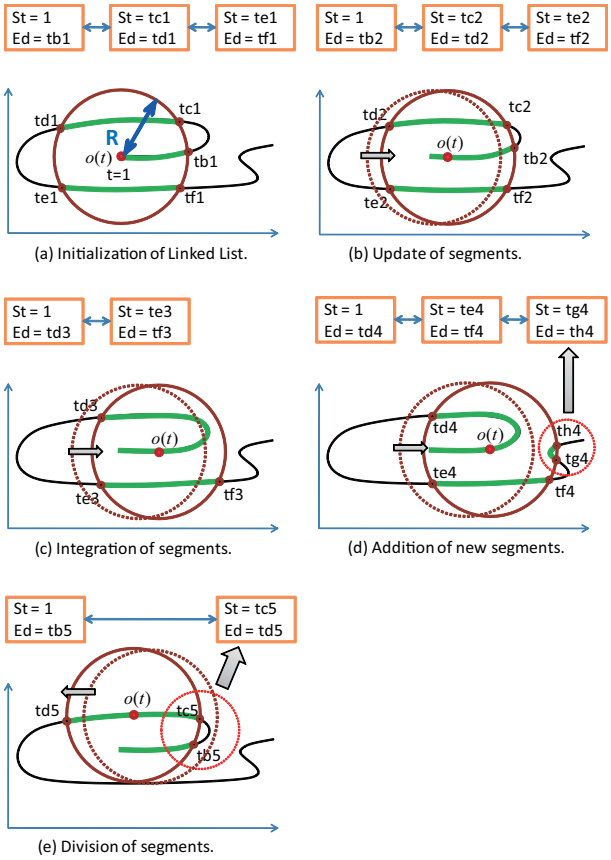


Fig. 2. Calculation of pattern density.

Then, the task to find sequences of data points that have large pattern density is solved efficiently as follows.

At time $t = 1$, the linked list is initialized by checking all N data points so that the elements in the list stores the endpoints, i.e. the start time (st) and the end time (ed), of all the segments inside the sphere as in Fig. 2 (a).

From time $t = 2$ to $t = N$, the linked list is updated by shifting the endpoints of each segment so that they lie exactly on the boundary of the sphere as in Fig. 2 (b). We can expect that the amount of shift between frames is very small, 0 or 1 in most cases, so the time to update the linked list can be considered as a constant. When a segment goes outside the sphere, it is removed from the linked list. When two segments are connected together, the two corresponding elements in the linked list are merged as in Fig. 2 (c).

When a new segment enters into the sphere, we have to detect it and add it to the linked list as in Fig. 2 (d). For this, if we check all N data points, it takes quadratic time in total. Instead, we randomly sample data points around the current data point, and add them to the linked list if they are inside the sphere and are not yet included in the linked list. Here, a new segment is not necessarily detected when it just enters into the sphere, because we can expect that it will be detected in the subsequent steps anyway if it is sufficiently similar to the current segment. When a segment is detected later, the pattern density in the past is modified at that time.

To sample data points inside the sphere efficiently, Partly Locality Sensitive Hashing (PLSH) is proposed as explained in the next section.

A segment in the sphere can be divided into two segments as in Fig. 2 (e). Since only the endpoints of segments are stored in the linked list, division cannot be detected immediately. To solve this problem, we randomly sample data points around the past data point $\boldsymbol{o}(t - T_{\text{delay}})$ by PLSH. If a sampled data point is outside the sphere but is included in the linked list, that means the corresponding segment is divided and the linked list should be modified. A divided segment is not necessarily detected when it just begins to leave the sphere, because we can expect that it will be detected in the subsequent steps anyway if it leaves the sphere. When a divided segment is detected later, the pattern density in the past is modified at that time.

The difference from [13], [1] is that they calculate the number of nearby segments which is sensitive to outlier segments found at the boundary of the sphere, while we calculate the density of segments which is much stable, Furthermore, they only consider addition of a new segment, while we also consider a divided segment, thus the accumulated error in calculating pattern density becomes small.

Finally, repeated patterns are estimated via global optimization framework using Dynamic Programming (DP) as explained in Section V.

IV. Partly Locality Sensitive Hashing

Partly Locality Sensitive Hashing (PLSH) is an extension to Locality Sensitive Hashing (LSH) scheme [14] which is an approximate nearest neighbor search algorithm.

A. Locality Sensitive Hashing

In LSH framework, L hash functions $g_l(\boldsymbol{p})$ $(1 \leq l \leq L)$ that calculate K dimensional hash values are defined as

$$g_l(\boldsymbol{p}) = < h_{l1}(\boldsymbol{p}), h_{l2}(\boldsymbol{p}), \ldots, h_{lK}(\boldsymbol{p}) > \quad (2)$$

where $\boldsymbol{p}$ is a d dimensional input value.

$h(\boldsymbol{p})$ can be any hash function $h : R^d \rightarrow U$ that preserves locality of input values, i.e. the probability of collision of two closer input values in a hash bucket is high. In a projection-based LSH, the following function can be used:

$$h(\boldsymbol{p}) = \lfloor (\boldsymbol{a} \cdot \boldsymbol{p} + b)/w \rfloor \tag{3}$$

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 $h(\boldsymbol{p})$. w controls the size of a hash bucket.

Given an input query $\boldsymbol{p}$, L hash buckets are determined by $g_l(\boldsymbol{p})$. Then, the data points stored in these buckets are examined whether they are sufficiently close to the query. LSH solves an approximate nearest neighbor search problem in $O(N^{1/\alpha})$ where $\alpha(> 1)$ is an approximation factor determined by the parameters of LSH.

B. Partly Locality Sensitive Hashing

In PLSH framework, L hash functions $g_l(\boldsymbol{p})$ $(1 \leq l \leq L)$ that calculate K dimensional hash values are similarly defined by combining locality sensitive hash functions and locality insensitive hash functions as

$$g_l(\boldsymbol{p}) = < hs_{l,1}(\boldsymbol{p}), \ldots, hs_{l,K_s}(\boldsymbol{p}), hi_{l,1}(\boldsymbol{p}), \ldots, hi_{l,K_i}(\boldsymbol{p}) > \tag{4}$$

where $hs(\boldsymbol{p})$ is a locality sensitive hash function and $hi(\boldsymbol{p})$ is a locality insensitive hash function.

$hs(\boldsymbol{p})$, or $hi(\boldsymbol{p})$, can be any hash function $h : R^d \rightarrow U$ that preserves, or does not preserve, locality of input values. In the projection-based scheme, these functions can be defined as

$$\begin{aligned} hs(\boldsymbol{p}) &= \lfloor (\boldsymbol{a} \cdot \boldsymbol{p} + b)/w_s \rfloor, \\ hi(\boldsymbol{p}) &= \lfloor (\boldsymbol{a} \cdot \boldsymbol{p} + b) \rfloor \bmod w_i \end{aligned} \tag{5}$$

where $hs(\boldsymbol{p})$ is exactly the same as in Eq.(3), while $hi(\boldsymbol{p})$ calculates discretized residual. $\boldsymbol{a}, b$ are randomly chosen to satisfy $\boldsymbol{a} \in R^d, ||\boldsymbol{a}|| = 1, 0 \leq b < w$ for each hash function.

Given an input query $\boldsymbol{p}$, L hash buckets are determined from $g_l(\boldsymbol{p})$. Then, the data points stored in these buckets are examined as in the case of LSH.

Fig. 3 shows how PLSH works.

C. Sparse sampling using PLSH

To sample data points from the entire sphere centered at $\boldsymbol{o}(t)$, the width of a hash bucket is fixed to R. If we use LSH to sample data points as in [12], it takes time to sample sufficient number of data points when N increases because the number of data points in a hash bucket also increases.

PLSH is effective in this case. We represent each data point by $d+1$ dimensional vector $(p_1, \ldots, p_d, t)^T$ where the first d elements represent a data point and the last 1 element represents the time frame.

To design $g_l(\boldsymbol{p})$, we choose K_s locality sensitive hash functions $hs_{l,k}(\boldsymbol{p})$ where the last value of $\boldsymbol{a}$ is always 0 and w_s is R. We also choose one locality insensitive hash function $hi_l(\boldsymbol{p})$ where the first d values of $\boldsymbol{a}$ is always 0. Because time frame is a discretized value, we design $\boldsymbol{a}$ and w_i in $hi_l(\boldsymbol{p})$ so that the width of a hash bucket is exactly 1 time frame and consecutive data points never collide in a same hash bucket.

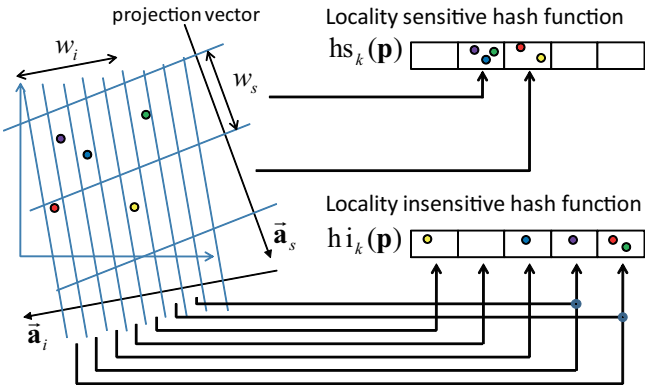


Fig. 3. Partly Locality Sensitive Hashing.

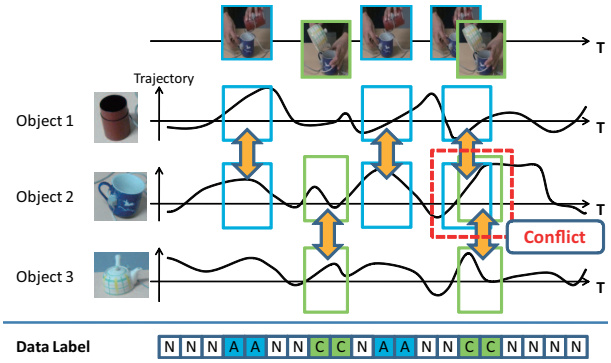


Fig. 4. Finding consistent repeated patterns.

In so doing, nearby data points along time axis accesses independent hash spaces in which no data point is shared when calculating pattern density. Thus, the computational time to examine the data points in a hash bucket decreases because the number of data points in a hash bucket decreases while the possibility of detecting a new data point, or a dividing data point, is the same as that of LSH.

V. Detection of Repeated Patterns

A. Target problem

In this study, interrelated multiple continuous time series data is dealt with. Fig. 4 shows an example of the problem. The input is trajectories of multiple objects, 3 in this case, and the algorithm is required to output the patterns that appear many times in the input trajectories. If the multiple trajectories are independent of each other, then the algorithm has only to be applied to each trajectory separately. However, if the trajectories are interrelated with each other, the algorithm has to find the consistent repeated patterns among the trajectories.

This is the case when repeated interactions between objects are of interest. In Fig. 4, there are 3 possible interactions between: (A) Obj.1 and Obj.2, (B) Obj.1 and Obj.3, (C)

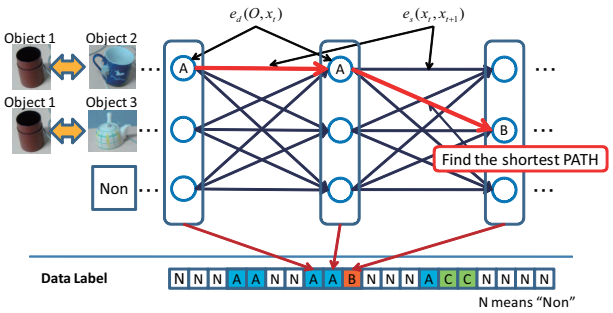


Fig. 5. Estimation of data labels via dynamic programming.

Obj.2 and Obj.3. If two of them are detected at the same time, this means a conflicting situation.

In the proposed method, we assume that there occurs a single significant interaction at most at a certain moment. This assumption is not necessarily true when there are 4 or more objects, or a motion is composed of more than two objects, but it will be satisfied in most cases.

Then, the problem is formulated as a combinatorial optimization problem in that a data label $x \in \{A, B, C, Non\}$ is assigned to each time frame as shown in the bottom of Fig. 4. A, B, C mean there occurs a repeated interaction between the respective object pairs, while Non means there is no repeated pattern at that time.

Please note that this formulation includes the case where repeated patterns are to be detected from a single time-series data in that we have only two labels: repeated and not-repeated.

B. Formulation

The problem is formulated as a combinatorial optimization problem regarding the data labels $\boldsymbol{X} = (x_1, \cdots x_N)$, $x_t \in \{A, B, \cdots, Non\}$ as shown in Fig. 4. There are $M = m(m-1)/2$ data labels where m is the number of objects in the scene and $\boldsymbol{o}_i(t)$ represents relative position between i-th object pairs at time t.

Given multiple time-series data $\boldsymbol{O} = (\boldsymbol{o}_1(1), \cdots, \boldsymbol{o}_1(N), \cdots, \boldsymbol{o}_M(1), \cdots, \boldsymbol{o}_M(N))$, we want to find $\boldsymbol{X}$ that minimizes the energy function

$$E(\boldsymbol{O}, \boldsymbol{X}) = E_d(\boldsymbol{O}, \boldsymbol{X}) + E_s(\boldsymbol{X}). \tag{6}$$

Note that the formulation is slightly different from our previous work [13], [1] in that the energy function has a third term that penalizes small velocity.

$E_d(\boldsymbol{O}, \boldsymbol{X})$ is a term regarding pattern density (Eq.(1)) and it penalizes data points with small density. It 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 pattern density at time t on the relative trajectory corresponding to x_t, and $< D_{x_t}(t) >$ is the mean value of $D_{x_t}(t)$.

$E_s(\boldsymbol{X})$ is a term regarding the prior distribution of $\boldsymbol{X}$. It penalizes different consecutive data labels to reject short patterns. It 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 $T(s) = 1$ iff $s =$ true, otherwise $T(s) = 0$.

Because all the terms in Eq.(6) satisfies the first order Markov property, the energy minimization problem can be analytically solved by Dynamic Programming as shown in Fig. 5.

C. Clustering of patterns

An agglomerative clustering technique is applied to the detected patterns of the same label to divide them into different types of patterns. In this study, for each pair of detected patterns, we calculate the ratio of the number of correspondences whose distance is below the radius R of the hyper sphere. If the ratio is above the threshold C_{mutual}, the pair of patterns are regarded as to belong to the same type of pattern.

In order to calculate the ratio efficiently, we use the middle point t_j of each segment in the linked list as the correspondence to time t and merge two patterns together if Eq.(7) is satisfied. If no segment corresponding to the current pattern is found at time t, the numerator of Eq.(7) becomes 0 at this particular time.

$$\frac{\sum_{t=t_s}^{t_e} 1 - |\frac{t-t_s}{t_e-t_s+1} - \frac{t_j-t_{sj}}{t_{ej}-t_{sj}+1}|}{t_e - t_s + 1} \geq C_{\text{mutual}} \tag{7}$$

where t_s, t_e and t_{sj}, t_{ej} are the start and the end time of two patterns. The absolute difference between the normalized time t and t_j is used to realize nonlinear elastic matching.

D. Computational complexity

It requires $O(N^{1+1/\alpha})$ for building PLSH tables and calculating pattern density for all the time frames, and $O(M^2 N)$ for DP. Assuming $M << N$, the total average computational time is $O(N^{1+1/\alpha})$.

VI. Experimental Results

A. Experimental setup

4 different manipulation tasks were performed by a subject and were used to evaluate the proposed method. There were 4 objects in the scene and an electromagnetic motion tracking system (Polhemus FASTRAK) was used to observe the trajectory of each object at 30Hz during demonstrations.

As shown in Fig. 6, 6 actions are defined. A subject was instructed to perform a task following a scenario made by combining 6 actions. To emulate the change in the environment during a long-term observation, the subject was instructed to relocate the objects on the table several times so that the relative relationship between stationary objects was changed.

TABLE I

EVALUATION OF DATASET 1 [896 FRAMES].

Action	A1	False	False	Precision	Recall
Presented #	4	positive	negative	ratio	ratio
kD-tree	4.00	0.00	0.00	1.00	1.00
LSH	3.60	1.00	0.40	0.78	0.90
PLSH	3.90	0.00	0.10	1.00	0.98

TABLE II

EVALUATION OF DATASET 2 [1428 FRAMES].

Action	A1	A2	False	False	Precision	Recall
Presented #	3	3	Positive	Negative	ratio	ratio
kD-tree	3.00	3.00	0.00	0.00	1.00	1.00
LSH	3.00	1.80	0.30	1.20	0.94	0.80
PLSH	3.00	3.00	0.00	0.00	1.00	1.00

TABLE III

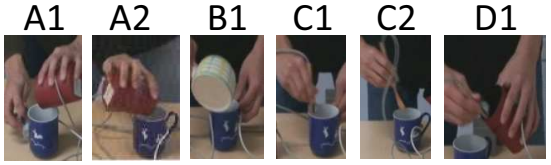
EVALUATION OF DATASET 3 [3737 FRAMES].

Action	A1	B1	C1	False	False	Precision	Recall
Presented #	5	7	6	Positive	Negative	ratio	ratio
kD-tree	5.00	7.00	6.00	0.00	1.00	1.00	0.95
LSH	4.70	5.60	5.50	0.00	3.20	1.00	0.83
PLSH	5.00	7.00	6.00	0.00	1.00	1.00	0.95

TABLE IV

EVALUATION OF DATASET 4 [5177 FRAMES].

Action	B1	C1	C2	D1	False	False	Precision	Recall
Presented #	7	5	10	10	Positive	Negative	ratio	ratio
kD-tree	7.00	3.00	6.00	0.00	0.00	16.00	1.00	0.50
LSH	6.40	3.60	3.60	0.00	0.60	19.00	0.96	0.41
PLSH	7.00	3.00	5.30	0.00	0.00	16.70	1.00	0.48



A1: Pour from a container to a cup from the left
A2: Pour from a container to a cup from the right
B1: Pour from a teapot to a cup
C1: Mix inside a cup with a spoon
C2: Put into a cup with a spoon
D1: Put from a container with a spoon

Fig. 6. 6 manipulation actions.

B. Evaluation

1) Different tasks: Because calculation of pattern density requires the highest computational complexity, 3 different methods of calculating pattern density, including exact search method by kD-tree, approximate search method by LSH and approximate search method by PLSH, were evaluated in terms of accuracy in detecting frequent patterns as well as in terms of computational time. Please note that the exact search method by kD-tree should give the best result in terms of accuracy. While the accuracy of the approximate search by LSH and PLSH depends on the randomly determined parameters of the hash functions, thus each experiment was evaluated 10 times and the average is shown.

The following parameters are used in all the experiments. The number of hash functions $L = 8$ and the number of locality sensitive hash functions $K_s = 3$. $w_s = 0.25$ which is equal to the radius R of the hyper sphere. w_i was chosen randomly from 50 to 100. We terminate the sampling process after C_{search} data points are sampled in LSH and PLSH. As suggested in [14], $C_{\text{search}} = 3L$ for both LSH and PLSH.

TABLE I, II, III, IV show the results of detecting repeated patterns using 3 different methods. All the experiments were performed on a Xeon 3.0GHz PC with 2G memory.

As an error measure, Precision and Recall ratio are calculated from True Positive (TP), False Positive (FP) and False Negative (FN) as:

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

The precision and recall ratio of the proposed method was almost identical to the ones of kD-tree, while that of LSH is much worse.

The reason why action D1 was never detected in TABLE IV was that C2 always followed just after D1 and these two consecutive actions were detected as a single action which were labelled as C2.

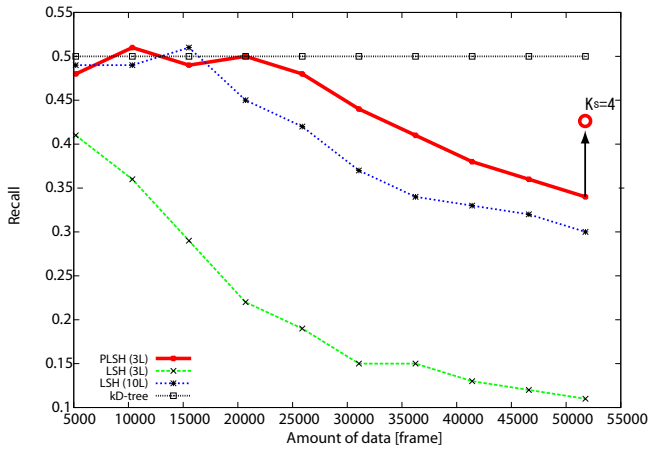


Fig. 7. Evaluation of recall ratio v.s. amount of data.

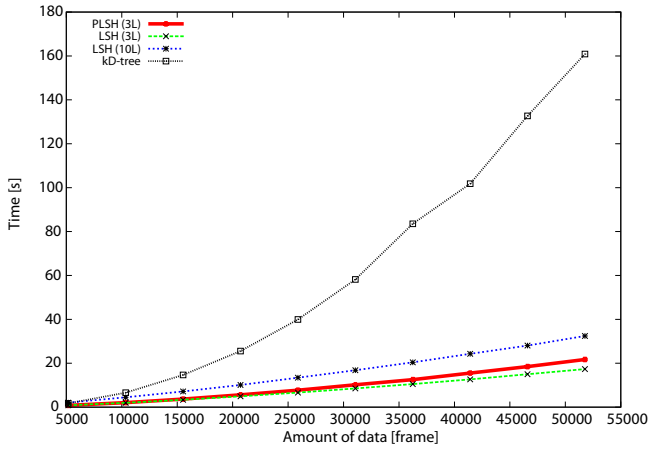


Fig. 8. Evaluation of computational time v.s. amount of data.

2) Different amount of data: In the second experiment. the 3 methods were evaluated against different amount of data. 10 datasets of different length were generated by simply concatenating the dataset in TABLE IV with Gaussian noise added.

Fig. 7 shows the Recall ratio obtained from the different methods. As the amount of data increases, recall ratio decreases. This is because the number of data points in a hash bucket increases as well. The result of kD-tree is the best. PLSH is far better than LSH with the same parameters. We also tested the case where $C_{\text{search}} = 10L$ for LSH to increase the number of buckets and to decrease the number of data points in a bucket, but PLSH is still better.

If we increase K_s, the number of locality sensitive hash functions, recall ratio is dramatically improved with a slight increase of computational time. If we set $K_s = 4$, the computational time is increased by 10 %, while recall ratio of the 10-th dataset becomes 0.43 as shown in Fig. 7.

Precision ratio achieves almost always 1.00 for all the methods, thus we omitted the evaluation of precision ratio.

Fig. 8 shows the computational time obtained from the different methods. The time of kD-tree increases quadratically, while the time of LSH and PLSH increases almost linearly. This is the strong advantage of the proposed method when dealing with a huge amount of observations.

VII. Conclusions

This paper presents a method that detects consistent repeated patterns from multiple observations in $O(N^{1+1/\alpha})$ time where N is the total amount of data.

Partly Locality Sensitive Hashing (PLSH) is proposed to find candidate repeated patterns efficiently by sparsely sampling nearby patterns in subquadratic computational time.

The proposed method was evaluated by detecting repeated interactions between objects in everyday manipulation tasks. The proposed method outperformed the method based on kD-tree and LSH in terms of accuracy or computational time.

Acknowledgments

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] Y. Tanabe, K. Ogawara, R. Kurazume, and T. Hasegawa, "Detecting frequent actions using partly locality sensitive hashing," in *Proc. The Fourth Joint Workshop on Machine Perception and Robotics (MPR)*, 2009.

[2] K. Ikeuchi and T. Suehiro, "Toward an assembly plan from observation part i: Task recognition with polyhedral objects," *IEEE Trans. Robotics and Automation*, vol. 10, no. 3, pp. 368–384, 1994.

[3] Y. Kuniyoshi, M. Inaba, and H. Inoue, "Learning by watching," *IEEE Trans. Robotics and Automation*, vol. 10, no. 6, pp. 799–822, 1994.

[4] K. Bernardin, K. Ogawara, K. Ikeuchi, and R. Dillmann, "A sensor fusion approach for recognizing continuous human grasping sequences using hidden markov models," *IEEE Transactions on Robotics*, vol. 21, no. 1, pp. 47–57, 2005.

[5] R. Agrawal, C. Faloutsos, and A. Swami, "Efficient similarity search in sequence databases," in *Proc. of 4th International Conference on Foundations of Data Organization and Algorithms*, 1993, pp. 69–84.

[6] C.-S. Perng, H. Wang, S. R. Zhang, and D. S. Parker, "Landmarks: A new model for similarity-based pattern querying in time series databases," in *16th International Conference on Data Engineering (ICDE'00)*, 2000, pp. 33–42.

[7] R. Staden, "Methods for discovering novel motifs in nucleic acid sequences," *Computer Applications in the Biosciences*, vol. 5, no. 5, pp. 293–298, 1989.

[8] J. Lin, E. Keogh, S. Lonardi, and P. Patel, "Finding motifs in time series," in *Proc. of the 2nd Workshop on Temporal Data Mining*, 2002, pp. 53–68.

[9] D. Yankov, E. Keogh, J. Medina, B. Chiu, and V. Zordan, "Detecting time series motifs under uniform scaling," in *Proc. of the 13th ACM KDD Intl. Conf. on Knowledge Discovery and Data Mining*, 2007, pp. 844–853.

[10] A. Mueen, E. Keogh, Q. Zhu, S. Cash, and B. Westover, "Exact discovery of time series motifs," in *Proc. of 2009 SIAM International Conference on Data Mining: SDM*, 2009, pp. 1–12.

[11] S. Uchida, A. Mori, R. Kurazume, R. Taniguchi, and T. Hasegawa, "Logical dp matching for detecting similar subsequence," in *Proc. of Asian Conference of Computer Vision*, 2007.

[12] J. Meng, J. Yuan, M. Hans, and Y. Wu, "Mining motifs from human motion," in *Proc. of EUROGRAPHICS'08*, 2008.

[13] K. Ogawara, Y. Tanabe, R. Kurazume, and T. Hasegawa, "Detecting repeated motion patterns via dynamic programming using motion density," in *Proc. 2009 IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2009, pp. 1743–1749.

[14] M. Datar, N. Immorlica, P. Indyk, and V. Mirrokni, "Locality-sensitive hashing scheme based on p-stable distributions," in *Proc. of the twentieth annual Symposium on Computational Geometry*, 2004, pp. 253–262.