2009 IEEE International Conference on Robotics and Automation
Kobe International Conference Center
Kobe, Japan, May 12-17, 2009

Detecting Repeated Motion Patterns via Dynamic Programming using Motion Density

Koichi Ogawara and Yasufumi Tanabe and Ryo Kurazume and Tsutomu Hasegawa

Abstract— **In this paper, we propose a method that detects repeated motion patterns in a long motion sequence efficiently. Repeated motion patterns are the structured information that can be obtained without knowledge of the context of motions. They can be used as a seed to find causal relationships between motions or to obtain contextual information of human activity, which is useful for intelligent systems that support human activity in everyday environment.**

The major contribution of the proposed method is two-fold: (1) motion density is proposed as a repeatability measure and (2) the problem of finding consecutive time frames with large motion density is formulated as a combinatorial optimization problem which is solved via Dynamic Programming (DP) in polynomial time $O(N \log N)$ 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 the previous method in terms of both detectability and computational time.

I. INTRODUCTION

Supporting daily life is one of the upcoming key applications of robotics technology and the capability to understand human activity is essential for this purpose. One of the common approaches to understand human activity is to design a set of necessary and sufficient task-dependent recognizers that detect significant actions as well as capture the necessary parameters to describe the action [1], [2], [3]. However, the variety of human activity in everyday environment is very diverse as opposed to those in a well-designed environment such as a factory, it is not practical to prepare recognizers to cover all of them.

For this reason, a desirable system should have a mechanism to learn new knowledge, new recognizers, from observation incrementally. As a bootstrap process to realize this mechanism, we are trying to detect repeated motion patterns as a structured information in observation data that can be extracted without knowledge of the context of a task. The basic idea is if a particular motion pattern appears many times in observation data, this pattern must be meaningful to the user or to the task. When a system detects repeated motion patterns while observing daily activity for a long period of time, e.g. several days, these patterns can be considered as meaningful actions and the causal relationships between these actions can be used to predict the next action of a user or to learn contextual information of human activity, which is useful for intelligent systems that support human activity in everyday environment.

In this paper, we propose a method that detects previously unknown repeated motion patterns in a long motion sequence efficiently. The contribution of the proposed method is two-fold. The first contribution is to introduce a notion of motion density as a repeatability measure which evaluates the number of similar motions for each time frame. The second contribution is that the problem of finding variable-length repeated motion patterns, that is consecutive time frames with large motion density, is formulated as a combinatorial optimization problem which is solved via Dynamic Programming (DP) in polynomial time $O(N \log N)$ in average where N is the total amount of data, while most of the previous methods assume known-length patterns [4] or take $O(N^2)$ time [5]. Among few exceptions is [6] which takes $O(N^{1+1/\alpha})$ time, however it solves the problem in three-steps. On the other hand, the proposed method finds repeated motion patterns under a single energy minimization framework.

The remainder of this paper is organized as follows. In Section II, we give an overview of the related research. Then we introduce a notion of motion density in Section III and extend the framework to cover interrelated multiple time series data for our particular application in Section IV. The details of the proposed method is explained in Section V. The experimental results are presented in Section VI and we conclude the paper in Section VII.

II. RELATED WORK

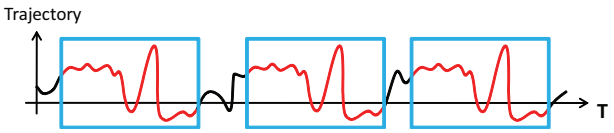


Fig. 1. Repeated Patterns in a Time Series Data

While there has been a large body of work regarding efficiently locating previously known patterns in time series data [7], [8], [9], 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 [10], in the data mining community [4], [11] and in the motion analysis community [12], [5], [6].

Finding repeated patterns has been studied extensively in analysis of nucleic acid sequences decades ago where the time series data is a sequence of discrete symbols from

K. Ogawara is with Faculty of Engineering, Kyushu University, Fukuoka, JAPAN ogawara@is.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

a set of 4 bases. Staden proposed a method that detects the most repeated known-length pattern by a voting scheme considering all the possible combination of patterns [10]. The computational time is linearly proportional to N, where N is the total amount of data.

The continuous counterpart has received much attention in the data mining community as well as in the motion analysis community. In the early stages, Liu et al. proposed a method in that a continuous time series data is first discretized into a sequence of symbols and the most repeated known-length pattern is found probabilistically by a voting scheme using a hash function in $O(N)$ time [4].

To deal with unknown-length repeated patterns, several approaches based on Dynamic Programming (DP) have been proposed. Ogawara et al. proposed a method that detects a set of motion patterns appeared in all M observation data in the same order using multi-dimensional DP matching in $O(N^M)$ time [13]. Uchida et al. proposed a method that detects pair-wise similar motion patterns from a single observation data using logical DP matching in $O(N^2)$ time [5].

To reduce computational time, Yankov et al. proposed a method that extends [4] to deal with uniform scaling of the known-length repeated motion patterns in $O(N^{1+1/\alpha})$ time [11]. To find totally unknown-length repeated motion patterns, Meng et al. proposed a three-steps method in $O(N^{1+1/\alpha})$ time in that it finds k-nearest-neighbor data points using Locality Sensitive Hashing (LSH), then it connects these detected data points along time axis to find repeated motion candidates and finally it performs spectral clustering to find the repeated motion patterns [6]. However, the output from each step cannot be guaranteed to be the global optimum.

In contrast to Meng's method, the proposed method finds repeated motion patterns from a continuous time series data under a single energy minimization framework in $O(N \log N)$ time, thus it finds the global optimum in terms of the given energy function. Another advantage is that it can detect consistent repeated motion patterns among interrelated multiple continuous time series data which is not possible with other methods.

Ogawara et al. previously proposed a method that finds consistent repeated motion patterns among interrelated multiple continuous time series data in $O(N \log N)$ time [14]. However, the problem is formulated as a combinatorial optimization problem with two correlated parameters. These two parameters are estimated alternately but independently, thus the global optimum is not guaranteed to be obtained. In the proposed method, the problem is re-formulated as a combinatorial optimization problem with a single parameter and the global optimum is analytically obtained via Dynamic Programming (DP).

Finding repeated patterns can be seen as a data compression problem where repeated patterns would be coded with a shorter word. Zhao et al. proposed a method that compresses a vector quantized motion data using Huffman coding which minimizes both the amount of information to code the entire data and the size of the dictionary under Minimum Description Length (MDL) principle [12]. It works well on ballet motion, however the result is severely affected by non-repeated patterns which dominates a time series data in general.

III. Motion Density

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

Here, we introduce a notion of motion density as a repeatability measure which evaluates the density of similar motions at around each time frame. With that, the problem is formulated as to find consecutive time frames with large motion density under an energy minimization framework.

Motion density $md(x,t)$ represents the density of similar motion segments in the neighborhood of data point $o_{x,t}$ at time t in time series data x. In this study, motion density is defined as the number of motion segments found within radius R around $o_{x,t}$ excluding the one to which $o_{x,t}$ belongs:

$$md(x,t) = \#\{\mathrm{ms}_j | o_{x,t} \notin \mathrm{ms}_j, \mathrm{ms}_j \subset \cup o_{x,k}, |o_{x,k} - o_{x,t}| < R\} \quad (1)$$

where motion segment ms_j is a continuous sequence of $o_{x,k}$ without a gap. If motion density is high, that means there are many repeated motion patterns similar to the one around $o_{x,t}$.

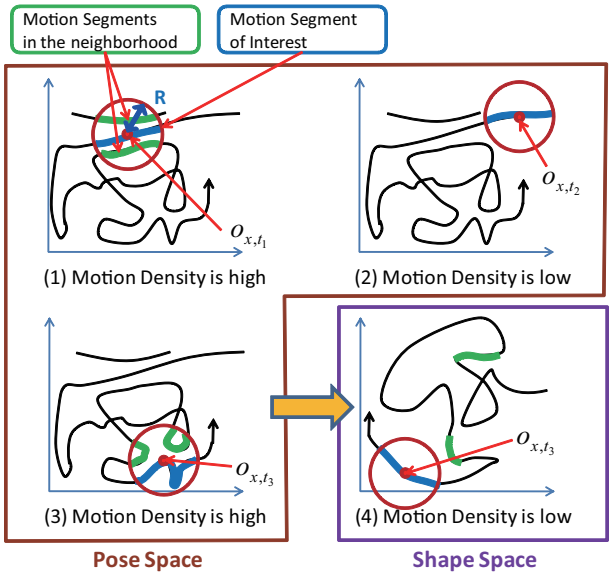


Fig. 2. Motion Density

Fig. 2 shows examples of motion density. The curve in Fig. 2 (1) to (3) represent the 2D slice of a time series data of arbitrary dimensions in Pose Space. Here, we assume that the time series data represents a trajectory, e.g. 6 D.O.F. position of a rigid object or joint angles of a human body, and Pose Space is the space where the time series data is directly projected without conversion.

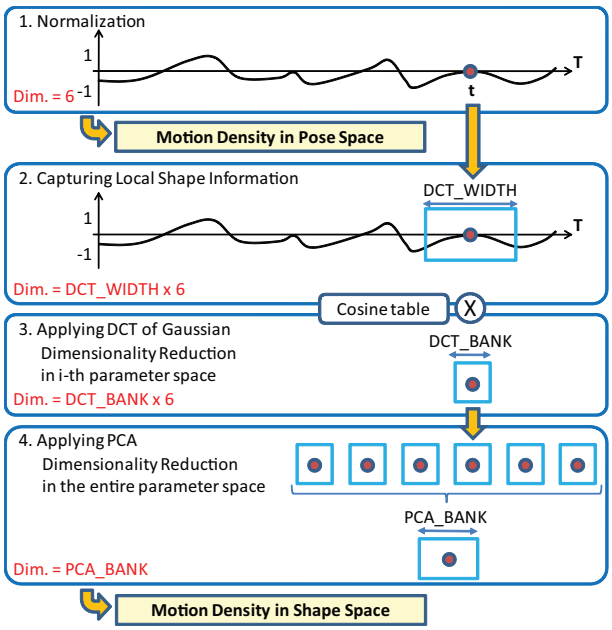


Fig. 3. Conversion from Pose Space to Shape Space

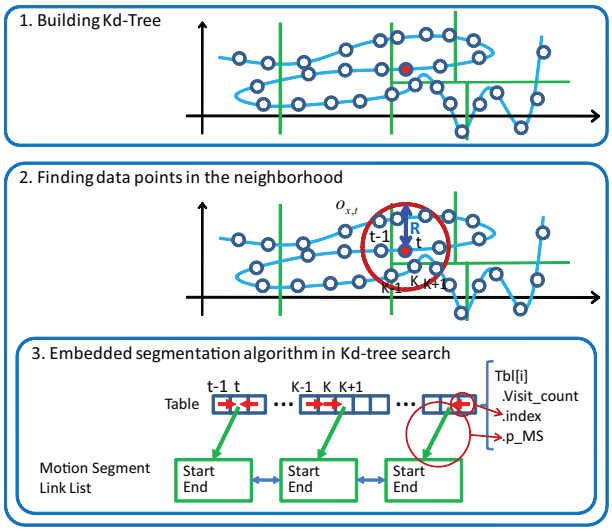


Fig. 4. Kd-tree with Embedded Segmentation Algorithm

In the case of Fig. 2 (1), there are several similar motion segments in the neighborhood of o_{x,t_1}, 2 in this case, thus the motion density $md_{\mathrm{p}}(x, t_1)$ is relatively high, On the other hand, in the case of Fig. 2 (2), there is no other motion segment in the neighborhood of o_{x,t_2}, thus the motion density $md_{\mathrm{p}}(x, t_2)$ is low.

In the case of Fig. 2 (3), there are several motion segments in the neighborhood of o_{x,t_3}, thus $md_{\mathrm{p}}(x, t_3)$ is high. However, as shown in the figure, the shape of these motion segments are completely different, thus these should not be considered as repeated motion patterns.

To take the similarity of shape into consideration, another space named Shape Space is defined. In this space, each data point at time t encodes the local shape of the original time series data at around $o_{x,t}$ so that data points similar in shape locally are projected to be in the neighborhood.

Motion density is also computed in Shape Space and both of the motion densities are used as a repeatability measure at time t. Fig. 2 (4) shows the converted trajectory of Fig. 2 (3) into Shape Space and motion density $md_{\mathrm{s}}(x, t_3)$ is low in this space.

A. Definition of Shape Space

In Shape Space, the metric between two data points is computed as the similarity in shape between two motion segments around the data points. The conversion from Pose Space to Shape Space is made of 4 steps as shown in Fig. 3.

Here, we assume that the original time series data in Pose Space represents a trajectory of an object and each data point is made of 6 parameters, 3 for translation and 3 for rotation (quaternion). However, the following conversion generalizes to any parameterization including joint angles.

1) **Normalization**
Because the range of values is totally different among the parameters, translation v.s. rotation, the values are normalized in each parameter space independently. Motion density in Pose Space is computed from the normalized data points.
2) **Capturing Shape Information**
To capture the local shape information, DCT_WIDTH normalized data points around time t are gathered to form a shape vector in i-th parameter space. The number of dimension is increased from 6 to DCT_WIDTH $\times$ 6.
3) **Applying DCT of Gaussian**
Discrete Cosine Transform (DCT) of Gaussian is applied to the shape vectors for dimensionality reduction in i-th parameter space. The number of dimension is reduced to DCT_BANK $\times$ 6.
4) **Applying PCA**
Principal Component Analysis (PCA) is applied to the set of reduced shape vectors for dimensionality reduction in the entire parameter space. The number of dimension is further reduced to PCA_BANK.

We can directly apply PCA to a shape vector without applying DCT of Gaussian. The reason why DCT of Gaussian is applied intermediately is that PCA in high dimensional space takes time. PCA after DCT of Gaussian dramatically reduces the computational time, while the quality of dimensionality reduction is comparable with PCA only approach. From the viewpoint of computational complexity, the time for applying PCA is considered to be a constant, but it makes the algorithm considerably faster in practice.

B. Finding Motion Segments using Kd-tree

To count the number of motion segments within the hypersphere of radius R around a data point, Kd-tree with an embedded segmentation algorithm is used. The algorithm works as shown in Fig. 4.

1) **Building Kd-tree**
Two Kd-trees are built separately for the set of data points both in Pose Space and in Shape Space. The data points are recursively subdivided by a hyper-plane perpendicular to the axis along which the variance of the remaining data points becomes maximum. The computational time for building Kd-tree is $O(N \log N)$.

2) **Finding Data Points in the Neighborhood**
For each data point at time t, all the data points within the hyper-sphere of radius R are searched. The average computational time is close to $O(\log N)$, however it becomes $O(N)$ in the worst case if all the data points lie within the hyper-sphere.

3) **Embedded Segmentation Algorithm in Kd-tree Search**
After finding all the data points within the hyper-sphere, these points have to be grouped into segments. However, if segmentation is performed separately after Kd-tree search finishes, additional time is required to sort the data points whose computational time is ether $O(N)$ or $O(K \log K)$ where K is the number of the data points found within the hyper-sphere.
Thus, a segmentation algorithm is embedded in the search algorithm of Kd-tree without increasing computational complexity. As shown in Fig. 4, it maintains a table of length N each of which holds 3 parameters: (1) $visit\_count$ is used to check if it is already visited of not, (2) $index$ holds the table-index of another data point that belongs to the same motion segment and (3) p_MS is a pointer to the corresponding Motion Segment structure. It also maintains the linked-list of Motion Segment structure each of which holds the start and end time of the motion segment.
The flow of the algorithm is presented in TABLE I. This algorithm is called with the time parameter t whenever a new data point within radius R is found during a single Kd-tree search. If the mean number of data points around a data point is $\bar{K}$, then this is called $\bar{K} \times N$ times in total. For each call, the set of Motion Segment structure is updated using one of the following functions: mergeMotionSegment() is to merge two consecutive segments together, enlargeMotionSegment() extends the range of the segment to include time t and addNewMotionSegment() creates a new segment at time t. visit_count is incremented once in each Kd-tree search, that is N times.
When all the data points within the hyper sphere are visited, the data points are already grouped into motion segments and motion density is calculated by just counting the number of segments.

C. Consistency between Motion Densities

Since repeated motion patterns must appear in both motion densities, a motion segment that is not shared by both of the motion densities is removed. Using remaining motion segments, motion density is calculated as eq.(1).

TABLE I
EMBEDDED SEGMENTATION ALGORITHM

```
1   Input: t;
2   idxl = idxr = t;
3   if tbl[t-1].visit_count == visit_count
4     idxl = t-1;
5     do (idxl = tbl[idxl].index)
6       while (idxl != tbl[idxl].index);
7     tbl[t-1].index = idxl;
8   end;
9   if tbl[t+1].visit_count == visit_count
10    idxr = t+1;
11    do (idxr = tbl[idxr].index)
12      while (idxr != tbl[idxr].index);
13    tbl[t+1].index = idxr;
14  end;
15  if idxl != t && idxr != t
16    mergeMotionSegment(tbl[idxl].p_MS, tbl[idxr].p_MS);
17    tbl[t].idx = idxl;
18  else if idxl != t
19    enlargeMotionSegment(tbl[idxl].p_MS, t);
20    tbl[t].idx = idxl;
21  else if idxr != t
22    enlargeMotionSegment(tbl[idxr].p_MS, t);
23    tbl[t].idx = idxr;
24  else
25    addNewMotionSegment(t);
26    tbl[t].idx = t;
27  end;
28  tbl[t].visit_count = visit_count;
```

IV. EXTENSION TO INTERRELATED MULTIPLE CONTINUOUS TIME SERIES DATA

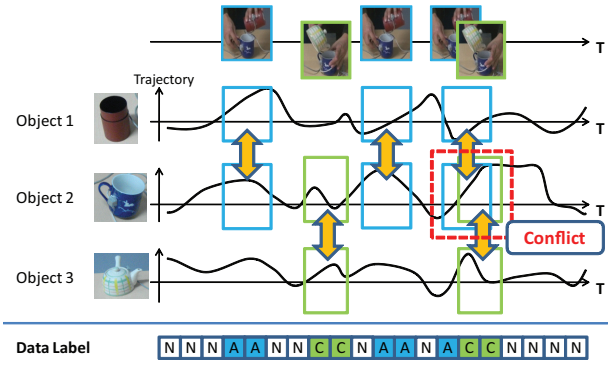


Fig. 5. Finding Consistent Repeated Motion Patterns

So far, we implicitly assume a single time series data. Here, we extend our framework to cover interrelated multiple continuous time series data. Fig. 5 shows an example of the problem. The input is multiple trajectories of objects, 3 in this case, and the algorithm is required to output the motion 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 motion patterns among the trajectories.

This is the case when repeated interactions between objects are of interest. In Fig. 5, there are 3 possible interactions

between: (A) Obj.1 and Obj.2, (B) Obj.1 and Obj.3, (C) 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 is 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. 5. $''A''$, "B'', "C'' mean there occurs a repeated motion between the respective object pairs, while $''Non''$ means there is non-repeated motion at that time.

Please note that this formulation is a natural extension from the single time series data case in that we have only two labels: repeated and not-repeated.

V. Algorithm Details

A. Problem Formulation

The problem is formulated as a combinatorial optimization problem regarding data labels $X = \{x_t | x_t \in \{A, B, \cdots, Non\}, 1 \leq t \leq N\}$ as shown in Fig. 5. There are two types of data labels: $''A''$, "B'', $\cdots$ mean the current data point is a repeated motion belonging to the time series data of that label, while $''Non''$ means the current data point is non-repeated motion.

In the case of finding consistent interactions between objects, there are $M = m(m-1)/2$ data labels where m is the number of objects in the scene and data point $o_{i,t}$ represents relative position between i-th object pairs at time t.

Given multiple time series data $TS = \{o_{i,t} | 1 \leq i \leq M, 1 \leq t \leq N\}$, we find X that minimizes an energy function defined as

$$E(TS, X) = E_v(TS, X) + E_m(TS, X) + E_s(X) \quad (2)$$

$E_v(TS, X)$ is a term regarding velocity$_{i,t}$, that is the velocity of the relative position between i-th object pairs at time t, and it penalizes stationary or non-interacted objects. In our particular application, objects stay still in most of the period in a time series data, thus we have to avoid the case where huge number of motionless regions are detected as repeated motion patterns. Otherwise, this term is not necessary. It is decomposed into

$$E_v(TS, X) = \sum_t e_v(TS, x_t, t)$$

$$e_v(TS, x_t, t) = -\log(1 - \exp(-\frac{\text{velocity}_{x_t,t}}{< \text{velocity}_{x_t,t} >}))$$

where $<$ velocity$_{x_t,t}$ $>$ is the mean value of velocity$_{x_t,t}$.

$E_m(TS, X)$ is a term regarding motion density and it penalizes data points with small number of similar motion segments in the neighborhood. It is decomposed into

$$E_m(TS, X) = \sum_t e_m(TS, x_t, t)$$

$$e_m(TS, x_t, t) = -\log(1 - \exp(-\frac{md_{\mathrm{p}}(x_t, t)}{< md_{\mathrm{p}}(x_t, t) >})) - \log(1 - \exp(-\frac{md_{\mathrm{s}}(x_t, t)}{< md_{\mathrm{s}}(x_t, t) >}))$$

where $md_{\mathrm{p}}(x_t, t)$ and $md_{\mathrm{s}}(x_t, t)$ are the motion densities defined in Pose Space and Shape Space. $< md_{\mathrm{p}}(x_t, t) >$ and $< md_{\mathrm{s}}(x_t, t) >$ are the mean values of $md_{\mathrm{p}}(x_t, t)$ and $md_{\mathrm{s}}(x_t, t)$ respectively.

$E_s(X)$ is a term regarding the prior distribution of X. It penalizes the difference between consecutive data labels which leads to reject short motion segments. It is decomposed into

$$E_s(X) = \sum_t e_s(x_t, x_{t+1})$$

$$e(x_t, x_{t+1}) = T(x_t \neq x_{t+1}) \cdot K,$$

where K is a constant and $T(s) = 1$ iff $s =$ true, otherwise $T(s) = 0$.

B. Energy Minimization via Dynamic Programming

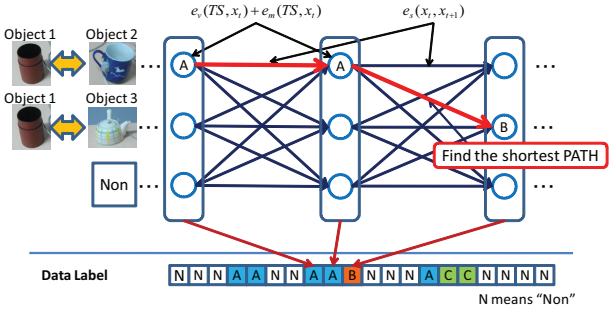


Fig. 6. Estimation of Data Labels via Dynamic Programming

All the terms in eq.(2) satisfies first order Markov property, the energy minimization problem can be analytically solved by Dynamic Programming as shown in Fig. 6.

The trouble is we know nothing about the energy terms E_v and E_m regarding data label $''Non''$. In general, the lower these energy terms are, the fewer the number of detected repeated motion patterns be. However, it is not clear how these terms are determined, since the appropriate value is context dependent. So we decided to adaptively change the values of these terms so as to output the desired number of motion patterns which is provided by a user.

First, the range of the value $e_v(TS, x_t, t) + e_m(TS, x_t, t)$ is calculated by checking all the combination of t and x_t. Then, assuming $e_{Non,t} = e_v(TS, Non, t) + e_m(TS, Non, t)$ is a fixed value independent of t, the optimal $e_{Non,t}$ within the range is obtained in binary-search fashion so as to best outputs the desired number of repeated motion patterns via Dynamic Programming.

C. Separation of Motion Patterns on a Same Time Series Data

So far, different motion patterns on a same time series data are labelled as the same. To separate these motion patterns, the following algorithm is applied.

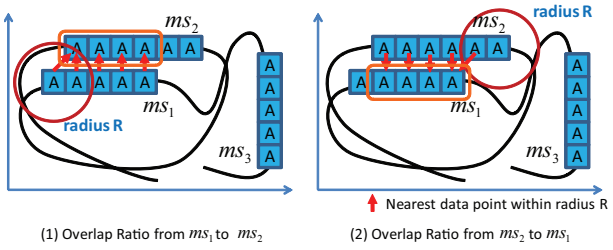


Fig. 7. Separation of motion patterns using mutual overlap ratio

Suppose one of the trajectory is labelled as in Fig. 7 and 3 motion segments are detected via DP. Then, for each pair of motion segments ms_i and ms_j, the overlap ratio $OR_{i,j}$ is calculated as the number of nearest data points within radius R in ms_j divided by the length of ms_i. If mutual overlap ratio, both $OR_{i,j}$ and $OR_{j,i}$, is greater than 0.5, two motion segments are grouped as the same motion pattern.

In the case of Fig. 7, $OR_{1,2} = \frac{4}{5}$ and $OR_{2,1} = \frac{4}{6}$, thus ms_1 and ms_2 are grouped as the same.

The nearest data point on each motion segment within radius R is computed during Kd-tree search, thus there is no increase in computational complexity.

D. Computational Complexity

The required computational time is $O(N \log N)$ for building Kd-tree, $O(N \log N)$ in average for motion density estimation and $O(M^2 N)$ for DP analysis. By assuming $M << N$, the total average computational time is $O(N \log N)$.

VI. Experimental Result

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. Fig. 8 shows the objects used in this experiment.

As shown in Fig. 9, 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 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.

B. Evaluation

In all the experiments, we use 33 for DCT_WIDTH, 4 for DCT_BANK and 6 for PCA_BANK so as to equalize the dimensions of Pose Space and Shape Space. Both of the radius of the hyper-sphere in Pose Space and in Shape Space

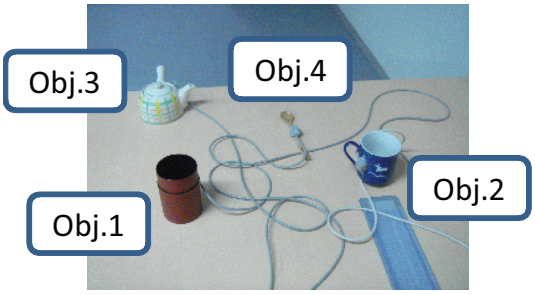
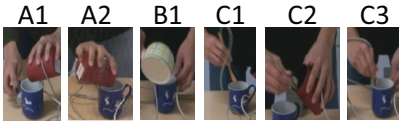


Fig. 8. Objects used in the Experiment



A1: Pour from Obj.1 to Obj.2 from the left
A2: Pour from Obj.1 to Obj.2 from the right
B1: Pour from Obj.3 to Obj.2
C1: Mix inside Obj.2 with Obj.4
C2: Spoon up from Obj.1 with Obj.4
C3: Put into Obj.2 with Obj.4

Fig. 9. 6 Repeated Actions in the Scenario

is 0.4 and K in E_s term is 15. These values were determined empirically.

The proposed method was compared with the method proposed in [14]. There are two noticeable differences between them. The first difference is that the desired number of motion patterns is provided by a user in the proposed method, while it is automatically selected in the previous method, though it is highly sensitive to the parameters. The second difference is that motion patterns in a same time series data can be discriminated in the proposed method, while it is not possible in the previous method.

TableII, III, IV, V shows the results of detecting repeated motion patterns using two different methods. All the computations were done on Xeon 3.0GHz PC.

As an error measure, Precision and Recall 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 proposed method outperformed the previous method in all the scores. However, there is a noticeable failure in TABLE V where the number of detected action C3 is zero. This is because, in this scenario, C3 always follows just after C2 and these two consecutive actions were detected as a single action which were labelled as C2.

VII. Conclusion

This paper presents a method that detects repeated motion patterns from a continuous time series data under a single energy minimization framework in $O(N \log N)$ time where

TABLE II

EVALUATION OF DATA SET 1 [896 FRAMES]

Action	A1						
Presented Number	4	TP	FN	FP	Precision	Recall	Time[msec]
Proposed Method	4	4	0	0	1.00	1.00	1164
Previous Method [14]	4	4	0	0	1.00	1.00	15873

TABLE III

EVALUATION OF DATA SET 2 [1428 FRAMES]

Action	A1	A2						
Presented Number	3	3	TP	FN	FP	Precision	Recall	Time[msec]
Proposed Method	3	3	6	0	0	1.00	1.00	2401
Previous Method [14]	5		5	1	2	0.71	0.83	80624

TABLE IV

EVALUATION OF DATA SET 3 [3737 FRAMES]

Action	A1	B1	C1						
Presented Number	6	7	6	TP	FN	FP	Precision	Recall	Time[msec]
Proposed Method	5	7	6	18	1	0	1.00	0.95	14800
Previous Method [14]	0	6	2	8	11	0	1.00	0.42	1769685

TABLE V

EVALUATION OF DATA SET 4 [5177 FRAMES]

Action	B1	C1	C2	C3						
Presented Number	7	5	10	10	TP	FN	FP	Precision	Recall	Time[msec]
Proposed Method	7	3	6	0	16	16	5	0.76	0.50	26703
Previous Method [14]	5	5			10	22	4	0.71	0.31	8388985

N is the total amount of data. The algorithm outputs top L repeated motion patterns, where the desired number L is provided by a user.

The problem is formulated as a combinatorial optimization problem in that the data label with large motion density is assigned to each time frame, which then be solved via Dynamic Programming (DP) analytically.

The proposed method was evaluated by detecting repeated interactions between objects in everyday manipulation tasks and outperformed the previous method[14] in terms of both detectability and computational time.

ACKNOWLEDGMENTS

This study was supported by Program for Improvement of Research Environment for Young Researchers from Special Coordination Funds for Promoting Science and Technology (SCF) commissioned by the Ministry of Education, Culture, Sports, Science and Technology (MEXT) of Japan.

REFERENCES

[1] 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.

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

[3] 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.

[4] 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.

[5] 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.

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

[7] 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.

[8] 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.

[9] T. Mori, Y. Nejigane, M. Shimosaka, Y. Segawa, T. Harada, and T. Sato, "Online recognition and segmentation for time-series motion with hmm and conceptual relation of actions," in *Int. conf. on Intelligent Robots and Systems*, 2005, pp. 2569–2574.

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

[11] 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.

[12] T. Zhao, T. Wang, and H. Shum, "Learning a highly structured motion model for 3d human tracking," in *Proc. of Asian Conference of Computer Vision*, 2002.

[13] K. Ogawara, J. Takamatsu, H. Kimura, and K. Ikeuchi, "Extraction of essential interactinos through multiple observations of human demonstrations," *IEEE Transactions on Industrial Electronics*, vol. 50, no. 4, pp. 667–675, 2003.

[14] K. Ogawara, Y. Tanabe, R. Kurazume, and T. Hasegawa, "Learning meaningful interactions from repetitious motion patterns," in *IEEE/RSJ 2008 Int. Conf. on Intelligent Robots and Systems*, 2008, pp. 3350–3355.