IEICE TRANS. FUNDAMENTALS, VOL.E88–A, NO.11 NOVEMBER 2005

PAPER Special Section on Concurrent/Hybrid Systems: Theory and Applications

Multiphase Learning for an Interval-Based Hybrid Dynamical System

Hiroaki KAWASHIMA[†a)] ***and*** **Takashi MATSUYAMA**[†]**,** ***Members***

SUMMARY This paper addresses the parameter estimation problem of an interval-based hybrid dynamical system (interval system). The interval system has a two-layer architecture that comprises a finite state automaton and multiple linear dynamical systems. The automaton controls the activation timing of the dynamical systems based on a stochastic transition model between intervals. Thus, the interval system can generate and analyze complex multivariate sequences that consist of temporal regimes of dynamic primitives. Although the interval system is a powerful model to represent human behaviors such as gestures and facial expressions, the learning process has a paradoxical nature: temporal segmentation of primitives and identification of constituent dynamical systems need to be solved simultaneously. To overcome this problem, we propose a multiphase parameter estimation method that consists of a bottom-up clustering phase of linear dynamical systems and a refinement phase of all the system parameters. Experimental results show the method can organize hidden dynamical systems behind the training data and refine the system parameters successfully.

key words: *hybrid dynamical system, interval transition, system identification, clustering of dynamical systems, expectation-maximization algorithm*

1. Introduction

Understanding the meaning of human commands and presenting suitable information to users is one of the primary objectives of man-machine interaction systems. Most of the previously proposed approaches, therefore, set the goal to classify verbal categories of dynamical events such as spoken words and gestures. Although this goal is still important, users sometimes feel frustration when the system gets out of human protocols. Many of the systems utilize only temporal order of events to estimate user intentions rather than to use temporal features such as acceleration patterns, tempo, and rhythms, which convey rich information of user's internal states.

To recognize detailed meaning of human activities and to generate actions with appropriate timing, we concentrate on modeling temporal structures in verbal and nonverbal events based on the relation among *intervals*. Firstly, we assume that a complex action is consist of dynamic primitives, which are often referred to as motion primitives [1], movemes [2], visemes [3], motion textons [4], and so on. For example, a cyclic lip motion can be described by simple lip motions such as "open," "close," and "remain closed." Once the set of dynamic primitives is determined, a complex action can be partitioned by temporal intervals with the labels of the primitives. Secondly, we assume that not only temporal orders but the duration of the intervals has negligible information in human communication. For instance, Kamachi et al. [5] suggested that duration of facial actions play an important role for human judgments of basic facial expression categories.

In this paper, we introduce an interval-based hybrid dynamical system (interval system) to describe complex events based on the structure of intervals. The system has a two-layer architecture consists of a finite state automaton and a set of multiple linear dynamical systems. Each linear dynamical system represents a dynamic primitive that corresponds to a discrete state of the automaton; meanwhile the automaton controls the activation timing of the dynamical systems. Thus, the interval system can generate and analyze complex multivariate sequences that consist of temporal regimes of dynamic primitives (see Fig. 1 for the example).

In spite of the flexibility of the interval system, especially for modeling human behaviors such as gestures and facial expressions, few applications have exploited the system to handle real-world signals. The reason is largely due to the paradoxical nature of the learning process: temporal segmentation and system identification problems need to be solved simultaneously.

The main contribution of this paper is that we propose a multiphase parameter estimation method that consists of (1) an agglomerative clustering process of dynamics based on a distance definition among dynamical systems, and (2) a refinement process of all the system parameters including

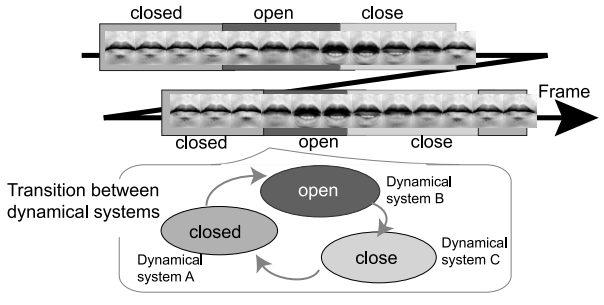


Fig. 1 Example of a generated lip image sequence from the interval system. Each of three linear dynamical systems models a simple lip motion, and an automaton generates a temporal order and a duration length of each motion.

Manuscript received April 4, 2005.
Manuscript revised June 10, 2005.
Final manuscript received July 13, 2005.

[†]The authors are with the Graduate School of Informatics, Kyoto University, Kyoto-shi, 606-8501 Japan.
a) E-mail: hiroaki@vision.kuee.kyoto-u.ac.jp

DOI: 10.1093/ietfec/e88–a.11.3022

dynamical systems and finite state automaton based on the approximated expectation-maximization (EM) algorithm.

In Sect. 2, we discuss the characteristics of our proposed method compared to related works. Section 3 describes a detailed model structure and an inference algorithm that searches the optimal interval sequence that provides the highest probability for the given observation. In Sect. 4 we verify the inference algorithm using simulated data. Section 5, proposes a multiphased learning method to identify the interval system. We evaluate the method using simulated and real data in Sect. 6. Finally, Sect. 7 concludes with a discussion of open problems.

2. Related Works

In this section, we introduce related works to explain the characteristics of our proposed method in the two aspects: the system definition and its identification.

2.1 Clock-Based and Event-Based State Transition

Previously proposed state transition models can be categorized into *clock-based* and *event-based* models. Clock-based models define the state transition in the physical-time domain by the formulation of differential or difference equations. For instance, dynamical systems in control theory [6] and recurrent neural networks are clock-based models. In contrast, an event-based model has a sustainable state, and it does not change the state before an input event occurred; for example, finite state automata and Petri nets are event-based models.

Clock-based models have the advantage of modeling continuously changing events such as human motion and utterance; however, they have the disadvantages of being unable to model duration of the state and to describe temporal structures of concurrent events such as lip motion and speech. Event-based models have the advantage of being able to model logical structures such as relations of temporal order, overlaps, and inclusion among events [7]. However, event-based models have typical signal-to-symbol problems; that is, it requires us to define an event set in advance.

2.2 Hybrid Dynamical Systems

Hybrid dynamical systems (hybrid systems) that integrate clock-based and event-based models are introduced to overcome the disadvantages of the two models in a complementary manner. In a hybrid system, an event-based model decides the activation timing of multiple clock-based models. The event-based model provides a solution to represent a logical structure of dynamic primitives; meanwhile, the clock-based models represent detailed dynamics in each primitive and also provide a distance definition among the primitives. Because of the high capability of modeling nonlinear and complicated events, hybrid systems are currently attracting great attention in various fields including controls, computer graphics, computer science, and neural networks.

A switching linear dynamical system (SLDS), which switches linear dynamical systems based on the state transition of the hidden Markov model (HMM) [8], becomes a common approach in modeling complex dynamics such as human motion [9]–[11]. In a SLDS, the transition between constituent subsystems (i.e., linear dynamical systems) is modeled in the same time domain as the subsystems. Assuming that the system consists of a subsystem set $Q = \{q_1, ..., q_N\}$, then the SLDS models the transition from subsystem q_i to q_j as a conditional probability $P(s_t = q_j | s_{t-1} = q_i)$, where t is synchronized to the continuous state transition in each subsystem.

In contrast, some hybrid systems model the transition between subsystems independently from the physical-time domain. Thus, the conditional probability becomes $P(s_k = q_j | s_{k-1} = q_i)$, where k represents only the temporal order of the transition. Bregler et al. [2] proposed a multilevel modeling method of human gate motion based this representation. The model comprises multiple linear dynamical systems as its subsystems, and an HMM switches the constituent subsystems. The stochastic linear hybrid systems [4], [12] are the extension of Breglar's model (e.g., motion texture [4] is proposed for motion generation purpose). Segmental models (SMs) [13] have been proposed in speech recognition fields as the unified model of segmental HMMs [8], segmental dynamical systems, and other segment-based models. This model uses the *segment* as a descriptor, which has a duration distribution. The segment represents a temporal region, in which one of the states is activated, and the total system represents phonemes and subwords as a sequence of the segments.

Piecewise linear (PWL) models and piecewise affine systems [14]–[16] are also categorized in the class of hybrid systems. A PWL model constructs a nonlinear map $f(x)$ by partitioning the domain $\mathcal{X} \subset \mathsf{R}^n$ into several regions $\mathcal{X}_1, ..., \mathcal{X}_N$ with polyhedral boundaries, which are referred to as *guardlines*. In each region, a linear mapping function is defined individually, and they are switched by the condition of $x \in \mathcal{X}$. As a result, the mapping function becomes nonlinear as a whole. From the viewpoint of switching condition definitions, we can say that the class of PWL models has static switching conditions, because the switching timing is decided by only predetermined regions. On the other hand, the hybrid systems described in the previous paragraphs have dynamic switching conditions: constituent subsystems are switched based on the state transition of the event-based model (e.g., automaton). In this paper, we concentrate on exploring hybrid systems that have dynamic switching conditions.

2.3 Interval-Based Hybrid Dynamical Systems

An interval-based hybrid dynamical system (interval system) proposed in this paper can be regarded as the extension of SMs. We use the term "intervals" instead of segments because our motivation is bringing Allen's temporal interval

logic [7], which exploits 13 topological relations between two intervals (e.g., meets, during, starts with, etc.), into the class of hybrid systems.

The difference from the SMs is that the interval system models not only the interval duration but the relation among multiple intervals. Once the intervals are explicitly defined, we can fabricate more flexible models to represent complex and concurrent dynamics appeared in man-machine interaction; for example, we can describe the temporal relation among concurrent motions appeared in multiple human parts based on the intervals [17]. In this paper, we concentrate on modeling a single signal from a single source for the simplest case, and describe the relation of adjacent intervals based on the correlation of their duration lengths.

Another advantage of explicit modeling of interval relations, and duration lengths, is that it enhances robustness against outliers during temporal partitioning process; in other words, the meta-level knowledge works as a constraint to the lower-level process. For instance, if the duration distributions of the subsystems are biased toward a long length, the system will not change the subsystem before the activation of the subsystem sustains enough length. As a result, the system improves the robustness of the temporal segmentation, which is a desirable property for iterative parameter estimation described in the next subsection.

2.4 Identification of Hybrid Dynamical Systems

As we described in Sect. 1, the difficulty of the learning process that identifies the hybrid system lies in its paradoxical nature. Let us assume that a large amount of multivariate sequences is given as training data. Then, the identification of the subsystems requires partitioned and labeled training data; meanwhile, the segmentation and labeling processes of training data require an identified set of subsystems. Moreover, the number of the subsystems is also unknown in general. Therefore, the parameter estimation problem requires us to simultaneously estimate temporal partitioning of training data (i.e., segmentation and labeling) and the set of subsystems (i.e., the number of the subsystems and their parameters).

In the following paragraphs, we introduce relevant works as for the identification of hybrid systems. We distinguish the cases when the number of subsystems is known or unknown.

(1) The number of subsystems is known:

The EM algorithm [18] is one of the most common approaches to solve this kind of paradoxical problems when the number of subsystems is given. The algorithm estimates parameters based on the iterative calculation. In each step, the algorithm fit the model to the training data using the model parameters updated in the previous step. Then, the parameters are updated based on the result of the current model fitting process.

Many of the hybrid system identification methods exploit the EM algorithm; for example, SMs [13], motion textures [4], stochastic linear hybrid systems [12] (which follows the method in [4]), and SLDSs [9]–[11], [19] applied this algorithm. A well-known identification technique for PWL models is introduced as k-mean like clustering [14]. This clustering technique can be regarded as an approximation (i.e., hard clustering) of the EM algorithm, which is often referred to as soft clustering [20].

Although the convergence of the EM algorithm is guaranteed, it strongly depends on the selection of initial parameters and often converges to a locally optimal solution, especially if the model has a large parameter space to search. As the alternative approach, an identification problem of PWL models can be recasted as a mixed integer programming (MIP), which find the globally optimal solution [15], [16]. We can apply the method when the logical switching conditions between subsystems can be transformed into a set of inequalities; however, it is difficult to transform the dynamic switching conditions, which are modeled by an automaton. Another disadvantage of MIP lies in its computational complexity. The problem is well known to be NP-hard in the worst case [14], [15]. For these reasons, we exploit the EM algorithm rather than the MIP-based methods to identify the interval system.

(2) The number of subsystems is unknown:

Most of previous works in hybrid system identification assumed that the number of subsystems is given, because the number often corresponds to that of manually defined operative conditions in the controlled object. In contrast, a set of dynamic primitives in human motion (e.g., primitive motion appeared in facial expressions) is undefined in most of cases. Whereas some heuristic sets of dynamic primitives are defined by human observation, they do not guarantee that the set is appropriate for man-machine interaction systems (e.g., automatic recognition of facial motion patterns). Hence, we should estimate the number of the subsystems (primitives) from a given training data set. The problem is often referred to as the cluster validation problem [20] (see Sect. 5.1.4 for details).

In stochastic linear hybrid systems [4], [12], an online clustering based on a greedy algorithm is applied to determine the number of subsystems (i.e., linear dynamical systems) and initialize the EM algorithm [4], [12]. The drawback is that it depends on the order of data presentation, and is also sensitive to outliers in training data. Moreover, the algorithm requires deciding appropriate thresholds beforehand. As a result, the algorithm tends to return too many subsystems.

2.5 Multiphase Learning for the Interval Systems

As we described in the previous subsection, the identification of interval systems requires solving the following problems:

1. initialization of the EM algorithm
2. estimation of the number of subsystems

In this paper, we propose a multiphase learning method that solves the problems above. The key idea is that we estimate the initial parameter set and the number of subsystems simultaneously based on a bottom-up (agglomerative) hierarchical clustering technique, which we propose in this paper (see Sect. 5.1 for details). This clustering algorithm is applied to a comparatively small number of typical training data that is selected from the given training set. For the second phase, we exploit the EM algorithm as a refinement process for all the system parameters, including parameters of the automaton. The process is applied to all the given training data, whereas the clustering process is applied to a selected typical training set. Due to the estimated parameters in the clustering process, the EM algorithm can be initialized by the parameter set that is relatively close to the optimum compared to randomly selected parameters (see Sect. 5.2 for details).

3. Interval-Based Hybrid Dynamical Systems

3.1 System Architecture

An interval system is a stochastic generative model that can generate multivariate vector sequences with interval-based structures. Each generated sequence can be partitioned by temporally ordered intervals, and each interval represents a single linear dynamics (see Fig. 2).

The system has a two-layer architecture. The first layer has a finite state automaton as an event-based model that models stochastic transitions between intervals. The automaton has an ability of generating interval sequences, in which each interval is labeled by the combination of the state of the automaton and the duration. The second layer consists of a set of multiple linear dynamical systems $\mathcal{D} = \{D_1, ..., D_N\}$ as clock-based models. Each dynamical system has an ability of generating multivariate vector sequences based on its linear dynamics (see Sect. 3.2).

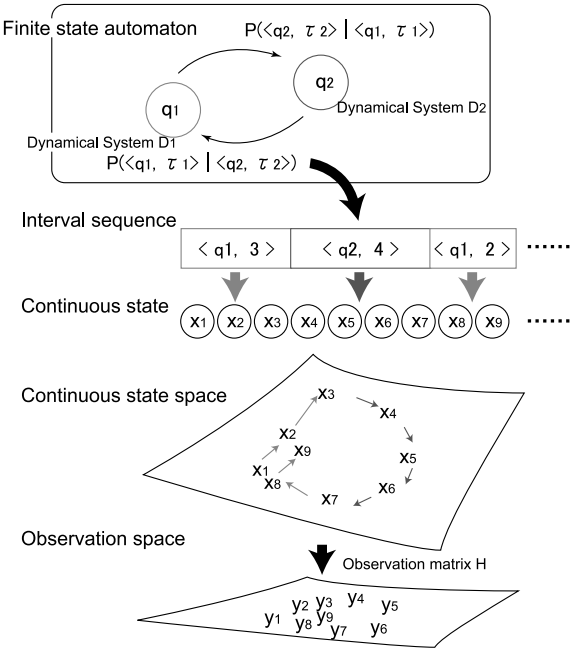


Fig. 2 Multivariate sequence generation by an interval-based hybrid dynamical system.

We define some terms and notations for the interval system in the following paragraphs. We simply use the term "dynamical systems" to denote linear dynamical systems.

Observation: Each dynamical system generates observation sequences of multivariate vector $y \in \mathbf{R}^m$ in a m-dimensional observation space.

Continuous states: All the constituent dynamical systems are assumed to share a n-dimensional continuous state space. Each activated dynamical system can generate sequences of continuous (real valued) state vector $x \in \mathbf{R}^n$, which are mapped to the observation space by a linear function that is also shared by all the dynamical systems.

Discrete states: The finite state automaton has a discrete state set $Q = \{q_1, ..., q_N\}$. The state $q_i \in Q$ corresponds to the dynamical system D_i.

Duration of discrete states: The duration that the automaton sustains a same discrete state becomes a positive integer, because the interval system is a discrete time model. To reduce parameter size, we set a minimum duration $l_{\min}$ and a maximum duration $l_{\max}$. Therefore the duration is defined by $\tau \in \mathcal{T} \triangleq \{l_{\min}, ..., l_{\max}\}$.

Intervals: The intervals generated by the automaton are defined as a combination of a discrete state and a duration length. We use notation $< q_i, \tau > \in Q \times \mathcal{T}$ to represent the interval that has state q_i and duration τ.

3.2 Linear Dynamical Systems

The state transition in the continuous state space by dynamical system D_i, and the mapping from the continuous state space to the observation space is modeled as follows:

$$x_t = F^{(i)} x_{t-1} + g^{(i)} + \omega_t^{(i)}$$
$$y_t = H x_t + \upsilon_t,$$

where $F^{(i)}$ is a transition matrix and $g^{(i)}$ is a bias vector. Note that each dynamical system has $F^{(i)}$ and $g^{(i)}$ individually. H is an observation matrix that defines linear projection from the continuous state space to the observation space. $\omega^{(i)}$ and υ is the process noise and the observation noise, which has Gaussian distribution $\mathcal{N}(0, Q^{(i)})$ and $\mathcal{N}(0, R)$, respectively. We use the notation $\mathcal{N}(a, B)$ to denote a Gaussian distribution with average vector a and covariance matrix B.

We assumed that all the dynamical systems share a single continuous state space. The main reason is that we want to reduce parameters in the interval system; it is, however, possible to design the system with an individual continuous state space for each dynamical system. In such cases, observation parameters $H^{(i)}$ and $R^{(i)}$ are required for each dynamical system. Although they provide more flexibility

in models, a large parameter space causes problems such as overfitting and high calculation costs.

Using the formulation and notation mentioned above, we can consider probability distribution functions as follows:

$$p(x_t|x_{t-1}, s_t = q_i) = \mathcal{N}(F^{(i)}x_{t-1}, Q^{(i)})$$
$$p(y_t|x_t, s_t = q_i) = \mathcal{N}(Hx_t, R),$$

where the probability variable s_t is an activated discrete state at time t (i.e., dynamical system D_i is activated).

Let us assume that the continuous state has a Gaussian distribution at each time t. Then, the transition of the continuous state becomes a Gauss-Markov process, which is inferable in the same manner as Kalman filtering [6]. Thus, the predicted state distribution under the condition of observations from 1 to $t-1$ is calculated as follows:

$$p(x_t|y_1^{t-1}, s_t = q_i) = \mathcal{N}(x_{t|t-1}^{(i)}, V_{t|t-1}^{(i)})$$
$$p(y_t|y_1^{t-1}, s_t = q_i) = \mathcal{N}(Hx_{t|t-1}^{(i)}, HV_{t|t-1}^{(i)}H^{\mathrm{T}} + R),$$

where the average vector $x_{t|t-1}^{(i)}$ and covariance matrix $V_{t|t-1}^{(i)}$ are updated every sampled time t.

Probability calculation: Suppose that the dynamical system D_j represents an observation sequence $y_{t-\tau+1}^{t} \triangleq y_{t-\tau+1}, ..., y_t$, which has a duration length τ. Then the likelihood score that the system D_j generates the sequence is calculated by the following equation:

$$d_{[t-\tau+1,t]}^{(j)} \triangleq P(y_{t-\tau+1}^{t}| < q_j, \tau >)$$
$$= \prod_{t'=t-\tau+1}^{t} \gamma^m \; p(y_{t'}|y_1^{t'-1}, s_{t'} = q_j), \qquad (1)$$

where we assume Gaussian distribution $N(x_{\mathrm{init}}^{(j)}, V_{\mathrm{init}}^{(j)})$ for the initial state distribution in each interval represented by dynamical system D_j. γ^m is a volume size of observations in the observation space to convert probability density values to probabilities. Note that m is the dimensionality of observation vectors. γ is assumed to be provided manually based on the size of the observation space (the range of each element in observation vectors). This likelihood score is used in Sect. 3.4 to evaluate the fitting of interval sequences to the given multivariate sequences.

3.3 Interval Transitions of the Automaton

Suppose that the automaton generates an interval sequence $\mathcal{I} = I_1, ..., I_K$, in which the adjacent intervals have no temporal gaps or overlaps. We assume first-order Markov property for the generated intervals; that is, the interval I_k depends on only the previous interval I_{k-1}. Therefore, the automaton models not only the transition of discrete states but also the correlation between the adjacent interval duration lengths.

We model the Markov process of intervals as the following conditional probability:

$$P(I_k =< q_j, \tau > |I_{k-1} =< q_i, \tau_p >),$$

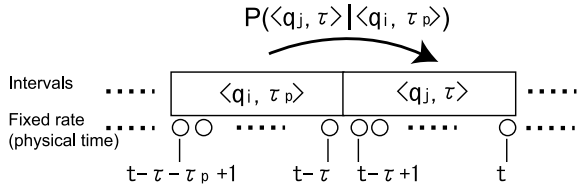


Fig. 3 First-order Markov process is assumed for a sequence of intervals.

where it denotes that the interval $< q_j, \tau >$ occurs after the interval $< q_i, \tau_p >$ (see Fig. 3). Thus, the probability of the interval $< q_j, \tau >$ can be calculated using all the possible intervals that have occurred just before the interval $< q_j, \tau >$ by the following equation:

$$P(I_k =< q_j, \tau >)$$
$$= \sum_{<q_i,\tau_p> \in Q \times \mathcal{T}} P(I_k =< q_j, \tau > |I_{k-1} =< q_i, \tau_p >)$$
$$\times P(I_{k-1} =< q_i, \tau_p >).$$

where the summation for $< q_i, \tau_p >$ does not include q_j (i.e., there are no self loops such as $q_j \to q_j$).

The probability $P(I_k =< q_j, \tau > |I_{k-1} =< q_i, \tau_p >)$ requires a large parameter set, which cause not only calculation costs but also the problem of overfitting during a training phase. We therefore use a parametric model for the duration distribution:

$$h^{(ij)}(l_k, l_{k-1}) \triangleq P(l_k, l_{k-1}|s_k = q_j, s_{k-1} = q_i),$$

where the two-dimensional distribution models a joint probability density function of duration lengths in adjacent interval pairs that has state q_i and q_j in this order. s_k and l_k is a probability variable of the discrete state and the duration length in the interval I_k, respectively. Note that we can assume an arbitrary density function as $h^{(ij)}(l_k, l_{k-1})$. For convenience, we use a two-dimensional Gaussian distribution normalized in the range of $[l_{\min}, l_{\max}]$; thus, the parameter set of the function $h^{(ij)}(l_k, l_{k-1})$ becomes $\{h_m^{(i)}, h_v^{(i)}, h_c^{(ij)}\}$, where $h_m^{(i)}$ and $h_v^{(i)}$ denotes mean and variance of duration lengths in discrete state q_i, and $h_c^{(ij)}$ denotes covariance between the adjacent duration lengths in discrete state sequences $q_i\, q_j$ $(i \neq j)$.

Using the above notations and assuming that the current discrete state is independent on the duration of the previous interval, we can calculate the interval transition probability as follows:

$$P(I_k =< q_j, \tau > |I_{k-1} =< q_i, \tau_p >) = \hat{h}^{(ij)}(\tau, \tau_p)A_{ij},$$

where $\hat{h}^{(ij)}(l_k, l_{k-1})$ is a one-dimensional Gaussian distribution:

$$\hat{h}^{(ij)}(l_k, l_{k-1}) \triangleq P(l_k|s_k = q_j, s_{t-\tau} = q_i, l_{t-\tau})$$
$$= \frac{h^{(ij)}(l_k, l_{k-1})}{\sum_{l_{k-1}} h^{(ij)}(l_k, l_{k-1})},$$

and A_{ij} is a discrete state transition probability:

$$A_{ij} \triangleq P(s_k = q_j | s_{t-\tau} = q_i),$$

where $i \neq j$. Note that, in the conventional discrete state models such as HMMs and SLDSs, the diagonal elements of the matrix $[A_{ij}]$ define the probabilities of the self loops. In the interval system, the diagonal elements are separated from the matrix and defined as duration distributions. As a result, the balance between diagonal and non-diagonal elements varies due to the duration length of the current state.

3.4 Inference of the Interval System

This section describes a probabilistic inference method that searches the optimal interval sequence to represents an input multivariate sequence. The method assumes that the interval system have been trained beforehand. As we will see in the following paragraphs, the method recursively finds the intervals that provide the highest likelihood score with respect to the input by generating all the possible intervals and selecting the optimal interval sets at every time t based on a dynamic programming technique. As a result, the input sequence is partitioned and labeled by discrete states that determine the most likely dynamical system to represent a multivariate sequence in each interval. In other words, the inference is a model fitting process that fits intervals to the given multivariate sequences. The likelihood of the trained model with respect to the input sequence is obtained simultaneously as the score of the fitting precision. This inference process is required in the EM algorithm of the interval system identification as we will see in Sect. 5.2.

The most naive method for the search is that first calculates the likelihood scores of the model from all the possible interval sequences independently, and then finds the best interval sequence that provides the largest likelihood. However, the calculation cost becomes order of $O(N^T)$ in this case. To avoid unnecessary calculation, we exploit a recursive calculation similar to HMMs.

Let us first consider the forward algorithm of the interval system. Suppose that input multivariate data y have been observed from time 1 to t, and the interval $I_k =< q_j, \tau >$ ends at time t. Considering all the possible intervals that have occurred just before the interval I_k, we can decompose the joint probability $P(I_k =< q_j, \tau >, y_1^t)$ as the following recursive equation:

$$\begin{aligned} &P(I_t =< q_j, \tau >, y_1^t) \\ &= \sum_{<q_i,\tau_p> \in Q \times \mathcal{T}} \Big\{ P(I_t =< q_j, \tau > | I_{t-\tau} =< q_i, \tau_p >) \\ &\quad \times P(I_{t-\tau} =< q_i, \tau_p >, y_1^{t-\tau}) \Big\} p(y_{t-\tau+1}^{\tau} | I_t =< q_j, \tau >), \end{aligned}$$

where the subscript t in the interval I_t denotes the ending time of the interval. Calculating the joint probability at every sampled time t, we get probabilities of all the possible interval sequences.

The forward algorithm often causes numerical underflow when the length of the input becomes longer. In the following paragraphs, we describe the Viterbi algorithm in which we can take logarithm of the formulation to avoid the numerical underflow problem. Let us use the following notation for the maximum occurrence probability of interval $< q_j, \tau >$ at time t that have been searched for the intervals before $t - \tau$:

$$\delta_t(q_j, \tau) \triangleq \max_{I_1,\ldots,I_{t-\tau}} P(I_t =< q_j, \tau >, y_1^t).$$

Suppose that the algorithm have found the most likely interval sequence before the previous interval, we can estimate $\delta_t(q_j, \tau)$ based on the following dynamic programming similar to the Viterbi algorithm of HMMs:

$$\delta_t(q_j, \tau) = \max_{<q_i,\tau_p>} \left\{ \hat{h}^{(ij)}(\tau, \tau_p) A_{ij} \delta_{t-\tau}(q_i, \tau_p) \right\} d^{(j)}_{[t-\tau+1,t]}. \tag{2}$$

Then, we can take logarithm of each term, because the summation in the forward algorithm have been changed to maximization in the formulation above.

The algorithm requires searching all the possible intervals for all the current intervals at every sampled time t. Therefore, the calculation cost becomes $O((NL)^2 T)$ ($L = l_{\max} - l_{\min} + 1$), which requires a greater cost compared to HMMs, which requires only $O(N^2 T)$. However, the cost can be reduced drastically in the case that the range L is small.

After the recursive search has ended at time T, we can backtrack to find the optimal interval sequence that provides the highest probability for the input sequence in generable interval sequences. Consequently, the maximum likelihood with respect to the given observation sequence and the optimal interval sequence is obtained by $\max_{<q_j,\tau>} \delta_T(q_j, \tau)$.

4. Verification of the Inference Algorithms

In this section, we verify the capability of the interval system and the inference algorithms (i.e., the forward and Viterbi algorithms) described in the previous section. First, the parameters of an interval system were given manually, and interval sequences were generated by the system for test data. Then, the data was used as input of the forward and Viterbi algorithms.

To concentrate on the interval-based state transition in the interval system, we set the observation matrix $H = I$ (unit matrix) and the observation noise covariance $R = O$ (zero matrix). The number of discrete states was $N = 3$ and the dimensionality of the continuous state space was $n = 3$. The range of the interval duration was $[l_{\min} = 1, l_{\max} = 30]$. We set the probability matrix of the discrete state transition as $A_{12} = A_{23} = A_{31} = 1$ and 0 for all the other elements to generate loops such as $q_1 \to q_2 \to q_3 \to q_1$. The initial distribution of the discrete state was 1 for q_1 and 0 for the remaining states q_2 and q_3.

The two dimensional distribution of the duration length $h^{(ij)}(l_k, l_{k-1})$ had {mean ($h_m^{(i)}$), variance ($h_v^{(i)}$)} of {6, 5}, {12, 30} and {16, 50} for q_1, q_2, and q_3, respectively. The covariance ($h_c^{(ij)}$) between the pairs of $\{q_1, q_2\}, \{q_2, q_3\}$ and $\{q_3, q_1\}$ was 12, 35 and 15, respectively. These covariances were

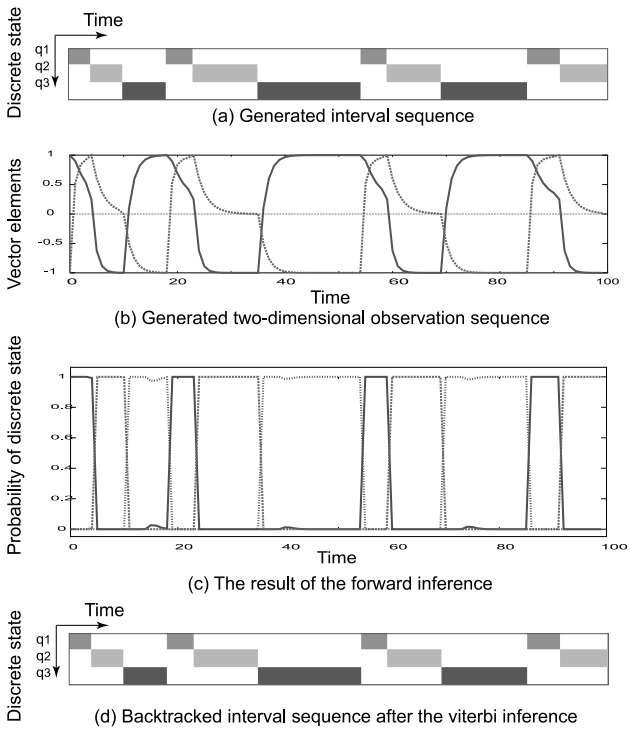


Fig. 4 Verification of the forward and Viterbi algorithms. (a) A generated interval sequence by a finite state automaton. (b) A generated observation sequence by three dynamical systems using interval sequence of (a). The solid line denotes the first element; the dashed line denotes the second element. (c) The result of the forward inference using (b) as input. Each line represents probability $P(s_t = q_j|y_1^t)\,(j = 1, 2, 3)$. (d) The result of the backtracked interval sequence after the Viterbi inference using (b) as input. We can see that the original interval sequence is obtained correctly.

designed to generate interval sequences that the duration lengths of the intervals were monotonically increased in the sequence of q_1, q_2, and q_3.

The transition matrices, bias vectors, and initial mean vectors of the dynamical systems were as follows:

$$F^{(1)} = \begin{bmatrix} 0.6 & -0.1 \\ -0.1 & 0.2 \end{bmatrix}, g^{(1)} = \begin{bmatrix} 0.2 \\ 0.8 \end{bmatrix}, x_{\text{init}}^{(1)} = \begin{bmatrix} 1.0 \\ -1.0 \end{bmatrix}$$

$$F^{(2)} = \begin{bmatrix} 0.3 & 0.0 \\ 0.0 & 0.6 \end{bmatrix}, g^{(2)} = \begin{bmatrix} -0.7 \\ 0.0 \end{bmatrix}, x_{\text{init}}^{(2)} = \begin{bmatrix} 0.3 \\ 1.0 \end{bmatrix}$$

$$F^{(3)} = \begin{bmatrix} 0.5 & 0.1 \\ -0.1 & 0.3 \end{bmatrix}, g^{(3)} = \begin{bmatrix} 0.6 \\ -0.6 \end{bmatrix}, x_{\text{init}}^{(3)} = \begin{bmatrix} -1.0 \\ 0.0 \end{bmatrix}$$

The process noise covariance matrices $Q^{(i)}\,(i = 1, 2, 3)$ and the covariance matrices $V^{(i)}\,(i = 1, 2, 3)$ of the initial state distribution were set to zero in the generation step, and was set to $0.001I$ (where I is a unit matrix) in the fitting step of intervals.

Figure 4(a) shows a generated interval sequence from the finite state automaton. The length of the sequence was $T = 100$. We see that the duration of the intervals increases monotonically in the sequence of q_1, q_2, and q_3, because we set positive correlation between the adjacent intervals (i.e., q_1 to q_2 and q_2 to q_3). Each dynamical system was activated by the label of each interval. Figure 4(b) shows a generated observation sequence, which corresponds to th continuous state sequence in this experiment (i.e., as a result of observation parameters $H = I$ and $R = O$). We see that the time-varying pattern of the observation changes based on the transition of the intervals.

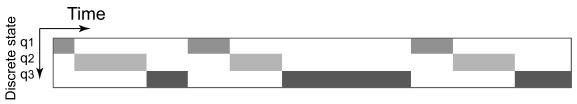


Fig. 5 A generated interval sequence using zero covariance duration distribution.

To verify the algorithms of forward and Viterbi inference, we input a generated sequence shown in Fig. 4(b) to the original interval system (i.e., the system that generated the input sequence). Figure 4(c) shows the result of the forward inference. Each line denotes the following probabilities of the discrete states under the condition of observations from time 1 to t:

$$P(s_t = q_j|y_1^t) = \frac{\sum_\tau P(I_t =< q_j, \tau >, y_1^t)}{P(y_1^t)}$$

Figure 4(d) shows the result of backtracked intervals after the Viterbi algorithm. We see that both of the forward and Viterbi inferences have found optimal interval sequences that are consistent with the original interval sequence in Fig. 4(a).

To see the influence of parameters in the duration distributions, we generated interval sequences with zero covariance between adjacent intervals. Figure 5 shows an example of the generated sequence. We see that the first state sequence of q_1, q_2, and q_3 has non-monotonic changes of duration (i.e., q_3 is shorter than q_2, while q_2 is longer than q_1), which implies that the duration of each discrete state is decided independently. Consequently, the correlation modeling with covariances between the adjacent intervals is necessary to represent rhythms and tempo as patterns of duration lengths.

5. Multiphase Learning for the Interval System

The goal of the interval system identification is to estimate the parameters N and the parameter set $\Theta = \{\theta_i, A_{ij}, \pi_i, h_m^{(i)}, h_v^{(i)}, h_c^{(ij)} | i, j = 1, ..., N, i \neq j\}$, where $\theta_i = \{F^{(i)}, g^{(i)}, Q^{(i)}, x_{\text{init}}^{(i)}, V_{\text{init}}^{(i)}\}$ denotes the parameter set of dynamical system D_i. In this paper, we assume that the observation matrix H and noise covariance matrix R are given. In case that the matrices are unknown, the subspace system identification techniques such as [21] can be applied by assuming a single transition matrix. Although the subspace identification technique has large sensitivity to noise, it successfully identifies an observation matrix especially from visual feature sequences [22].

As we described in Sect. 2.4, the parameter estimation of the interval system necessitates to solve two problems: initialization of system parameters and estimation of the

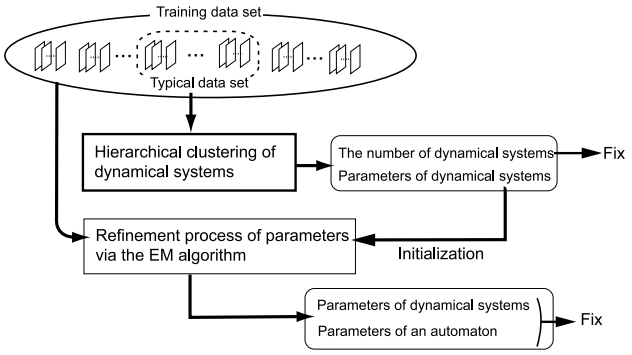


Fig. 6 A multiphase learning for the interval system.

number of linear dynamical systems. To overcome the problem, we propose a multiphase learning method described in the following paragraphs (see also Fig. 6) to estimate parameters of the interval system. The key idea is that we divide the estimation process into two steps: a clustering process of dynamical systems using a typical training data set (i.e., a subset of the given training data set), and a refinement process for all the parameters using all the training data.

Clustering of Dynamical Systems: The first step is a clustering process that finds a set of dynamical systems: the number of the systems and their parameters. This step requires that a set of typical sequences is given for the training data, and the sequences have been already mapped to the continuous state space. Then, we propose an agglomerative hierarchical clustering technique that iteratively merges dynamical systems.

Refinement of the Parameters: The second step is a refinement process of the system parameters based on the EM algorithm. Whereas the algorithm strongly depends on its initial parameters, the previous clustering step provides an initial parameter set that is relatively close to the optimum compared to a randomly selected parameter set.

5.1 Hierarchical Clustering of Dynamical Systems

Let us assume that a multivariate sequence $y_1^T \triangleq y_1, ..., y_T$ is given as a typical training data (we consider a single training data without loss of generality), then we simultaneously estimate a set of linear dynamical systems $\mathcal{D}$ (i.e., the number of linear dynamical system N and their parameters $\theta_1, ..., \theta_N$) with an interval set $\mathcal{I}$ (i.e., segmentation and labeling of the sequence), from the training sample y_1^T. Note that the number of intervals K is also unknown. We formulate the problem as the search of the linear dynamical system set $\mathcal{D}$ and the interval set $\mathcal{I}$ that maximizes the overall likelihood with respect to the training data: $\mathcal{L} = P(y_1^T|\mathcal{I}, \mathcal{D})$. Because the likelihood monotonically increases with an increase in the number of dynamical systems, we need to determine the right balance between the likelihood and the number N. A hierarchical clustering approach provides us an interface such as the history of model fitting errors in each merging steps to decide the number of dynamical systems

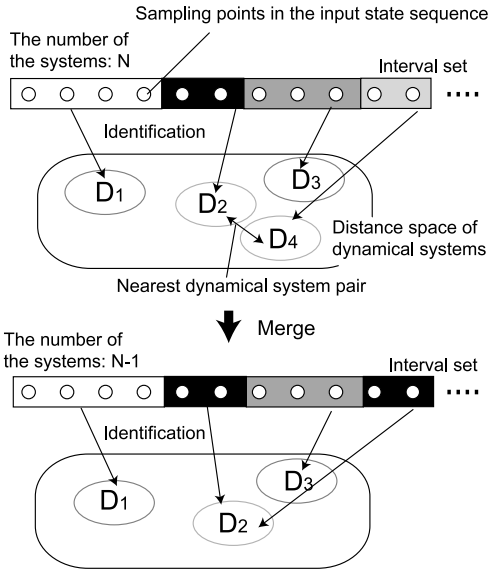


Fig. 7 Hierarchical clustering of dynamical systems.

Algorithm 1 Agglomerative Hierarchical Clustering.

for I_i in $\mathcal{I}_{\text{init}}$ **do**
  $D_i \leftarrow \mathsf{Identify}\,(I_i)$
end for
for all pair$\left(D_i,\ D_j\right)$ where $D_i,\ D_j \in \mathcal{D}$ **do**
  $\mathsf{Dist}\,(i,\ j) \leftarrow \mathsf{CalcDistance}\left(D_i,\ D_j\right)$
end for
while $N \geq 2$ **do**
  $(i^*,\ j^*) \leftarrow \arg\min_{(i,\ j)} \mathsf{Dist}\,(i,\ j)$
  $\mathcal{I}_{i^*} \leftarrow \mathsf{MergeIntervals}\left(\mathcal{I}_{i^*},\ \mathcal{I}_{j^*}\right)$
  $D_{i^*} \leftarrow \mathsf{Identify}\,(\mathcal{I}_{i^*})$
  erase D_j^* from $\mathcal{D}$
  $N \leftarrow N - 1$
  for all pair$\left(D_i^*,\ D_j\right)$ where $D_j \in \mathcal{D}$ **do**
    $\mathsf{Dist}(i^*,\ j) \leftarrow \mathsf{CalcDistance}\left(D_{i^*},\ D_j\right)$
  end for
end while

(see Sect. 5.1.4 for details).

Algorithm 1 shows the details of the clustering process. This algorithm requires initial partitioning of the training sequence, which we refer to as an initial interval set $\mathcal{I}_{\text{init}}$. We will discuss about the initialization in Sect. 5.1.1. In the first step of the algorithm, dynamical systems are identified (which is denoted by Identify in Algorithm 1 from each interval in the initial interval set $\mathcal{I}_{\text{init}}$ individually based on the identification method proposed in Sect. 5.1.2. $\mathcal{I}_i$ is the interval set that comprises intervals labeled by D_i. Then, we calculate the distances (denoted by CalcDistance) for all the dynamical system pairs based on the distance definition described in Sect. 5.1.3. In the following steps, the nearest dynamical systems are merged (which is denoted by MergeIntervals) iteratively. Two interval sets that belong to the nearest dynamical system pair are also merged simultaneously. As a result, all the dynamical systems are merged to a single dynamical system. Figure 7 shows an example of the step in which the dynamical system D_2 and D_4 are

merged.

5.1.1 Initialization of the Clustering Algorithm

Because the clustering algorithm iteratively merges intervals in the initial interval set $\mathcal{I}_{\text{init}}$, the estimation of the set $\mathcal{I}_{\text{init}}$ is an important problem. In this paper, we set some conditions to determine $\mathcal{I}_{\text{init}}$. First, we require the number of the intervals $|\mathcal{I}_{\text{init}}|$ to be relatively smaller than the length of the training sequence, because the calculation cost depends on $|\mathcal{I}_{\text{init}}|$. Therefore, the simplest partitioning approach that divides the training sequence into fixed length small intervals is undesirable. Second, we require that the initial intervals divide stationary pose from motion in the training sequence. Third, we require the intervals $I_i \in \mathcal{I}_{\text{init}}$ to be simple in the sense that the patterns in the initial intervals appear frequently in the training sequence.

To satisfy the conditions above, we exploit monotonicity of the motion. Many of the motions observed in man-machine interaction, especially visual feature sequences, can be divided by monotonic patterns, because human moves his body using the combination of intermittent muscle controls. We therefore divide the training sequence by zero-crossing points of the first-order difference of feature vectors. Moreover, we set a single interval to the temporal range, in which the value remains smaller than the given threshold.

5.1.2 Constrained System Identification

To identify the system parameters from only a small amount of training data, we need constraints for estimation of an appropriate dynamics. In this paper, we concentrate on extracting human motion primitives observed in such as facial motion, gaits, and gestures; therefore, constraints based on stability of dynamics are suitable to find motion that converges to a certain state from an initial pose. The key idea to estimate stable dynamics is the method that constrains on eigenvalues. If all the eigenvalues are lower than 1, the dynamical system changes the state in a stable manner.

Given a continuous state sequence mapped from the observation space, the parameter estimation of a transition matrix $F^{(i)}$ from the sequence of continuous state vectors $x_b^{(i)}, .., x_e^{(i)}$ becomes a minimization problem of prediction errors. Let us use the notations $X_0^{(i)} = [x_b^{(i)}, ..., x_{e-1}^{(i)}]$ and $X_1^{(i)} = [x_{b+1}^{(i)}, ..., x_e^{(i)}]$, if the temporal range $[b, e]$ is represented by linear dynamical system D_i. Then, we can estimate a transition matrix $F^{(i)}$ by the following equation:

$$\begin{aligned} F^{(i)*} &= \arg\min_{F^{(i)}} \|F^{(i)} X_0^{(i)} - X_1^{(i)}\|^2 \\ &= \lim_{\delta^2 \to 0} X_1^{(i)} X_0^{(i)\top} (X_0^{(i)} X_0^{(i)\top} + \delta^2 I)^{-1}, \end{aligned} \tag{3}$$

where I is the unit matrix and δ is a positive real value.

For the constraints on the eigenvalues, we stop the limit in Eq. (3) before $X_0^{(i)\top}(X_0^{(i)} X_0^{(i)\top} + \delta^2 I)^{-1}$ converges to a pseudo-inverse matrix of $X_0^{(i)}$. Using Gershgorin's theorem in linear algebra, we can determine the upper bound of eigenvalues in the matrix from its elements. Suppose $f_{uv}^{(i)}$ is an element in row u and column v of the transition matrix $F^{(i)}$. Then, the upper bound of the eigenvalues is determined by $\mathcal{B} = \max_u \sum_{v=1}^{n} |f_{uv}^{(i)}|$. Therefore, we search a nonzero value for δ, which controls the scale of elements in the matrix, that satisfies the equation $\mathcal{B} = 1$ via an iterative numerical calculation.

5.1.3 Distance Definition between Dynamical Systems

In order to determine a pseudo distance between two linear dynamical systems, we compare the following three approaches: (a) direct comparison of model parameters, (b) decreased likelihood of the overall system after the merging of two models [23], and (c) distance measure of distributions (e.g., KL divergence [24]).

The approach (a) is not desirable particularly with regards to our bottom-up approach, because a linear dynamical system has a large parameter set that often causes overfitting. The approach (b), which is often exploited with stepwize-optimal clustering [20], performs well in the ideal condition, but it requires the calculation cost of likelihood scores for all the combination of the linear dynamical system pairs. We observed from our experience that (c) provides stable dissimilarity measure for our algorithm with a realistic calculation cost. Therefore, we define the distance between linear dynamical system D_i and D_j as an average of two asymmetric divergences:

$$Dist(D_i, D_j) = \{KL(D_i \| D_j) + KL(D_j \| D_i)\}/2,$$

where each of the divergences is calculated as an approximation of KL divergence:

$$KL(D_i \| D_j) \sim \frac{1}{C_i} \sum_{I_k \in \mathcal{I}_i} \left\{ \log P(y_{b_k}^{e_k} | D_i) - \log P(y_{b_k}^{e_k} | D_j) \right\},$$

where $y_{b_k}, ..., y_{e_k}$ is an observation sequence partitioned by interval I_k. C_i is the summation of interval length in the interval set $\mathcal{I}_i$ that is labeled by a linear dynamical system D_i. Note that we can calculate the likelihoods based on Eq. (1).

5.1.4 The Cluster Validation Problem

The determination of the appropriate number of dynamical systems is an important problem in real applications. The problem is often referred to as the cluster validation problem, which remains essentially unsolved. There are, however, several well-known criteria, which can be categorized into two types, to decide the number of clusters. One is defined based on the change of model fitting scores, such as log-likelihood scores and prediction errors (approximation of the log-likelihood scores), during the merging steps. If the score decreased rapidly, then the merging process is stopped [25]. In other words, it finds *knee* of the log-likelihood curve. The other is defined based on information

theories, such as minimum description length and Akaike's information criterion. The information-theoretical criteria define the evaluation functions that consist of two terms: log-likelihood scores and the number of free parameters.

Although information-theoretical criteria work well in simple models, they tend to fail in evaluating right balance between the two terms, especially if the model becomes complex and has a large number of free parameters [26]. Because this problem also arises in our case, we use model fitting scores directly. First, we extract candidates for the numbers of the dynamical systems by finding peaks in the difference of model fitting errors between current and the previous steps. If the value exceeds a predefined threshold, then the number of dynamical systems in that step is added to the candidates. We consider that user should finally decide the appropriate number of the dynamical systems from the extracted candidates.

5.2 Refinement of Parameters via the EM Algorithm

This subsection describes a refinement process of the overall system parameter set Θ including the parameters of the automaton by exploiting the result of the clustering process. In this process, the number of dynamical systems N is fixed.

In order to refine the parameters, we apply an iterative estimation based on the EM algorithm. This algorithm starts from a given initial parameter set Θ_{init}, and repeats two steps: E (expectation) step and M (maximization) step. The E step fits the model to the given training samples based on the current model parameters, and calculates the expectation of log-likelihood with respect to the given samples and all the possible fitting instances. The M step updates the parameters to maximize the expectation of log-likelihood using the result of the statistics in the E step. After the iteration steps, the algorithm converges to the optimal parameters, and finds the result of the model fitting simultaneously.

Although the EM algorithm has been proved to converge, the obtained solutions are often trapped by local optima. This fact becomes critical problem especially if the size of the parameters increases. For a large parameter models, the algorithm therefore requires a parameter set that is relatively close to the optimum for the initialization.

The multiphase learning method proposed in this paper overcomes the initialization problem by exploiting the result of the clustering process applied for a typical training data set. From the viewpoint of the likelihood maximization, the clustering process returns approximately estimated parameters for the dynamical systems. To initialize remaining parameters such as the interval transition matrix $[A_{ij}]$ and interval duration distributions $h^{(ij)}(l_k, l_{k-1})$, we exploit Viterbi algorithm, which is introduced in Sect. 3.4, with some modification; that is, we set all the transition probabilities to be equal: $A_{ij} = 1/(N-1)$, and we also assume the interval duration $\hat{h}^{(ij)}(l_k, l_{k-1})$ to be uniform distributions in the range of $[l_{\min}, l_{\max}]$, where $i, j = 1, ..., N (i \neq j)$. As a result, Eq. (2) in the Viterbi algorithm becomes

$$\delta_t(q_j, \tau) = \max_{<q_i, \tau_p>} \left\{ \delta_{t-\tau}(q_i, \tau_p) \right\} d^{(j)}_{[t-\tau+1,t]}.$$

After the initialization above, parameter refinement iterations are applied to all the training data $\mathcal{Y} \triangleq \{y_1^{T_1}, ..., y_1^{T_M}\}$, where M is the number of the training sequences and $T_1, ..., T_M$ are the lengths of each sequence. In this paper, we approximate the EM algorithm, because the original EM algorithm requires forward/backward inferences that often cause numerical underflow. We do not calculate the statistics for all the possible hidden variables (possible interval sequences), but use only a single interval sequence that gives the highest log-likelihood in each step. Thus, the total refinement process is described as follows:

1. Initialize $\Theta = \Theta_{\text{init}}$.
2. Repeat the following steps while the total likelihood changes greater than a given threshold ϵ:

   E step: Search an optimal interval sequence, which is labeled by the discrete states, for all the training sequences based on the current parameter set Θ:

   $$\mathcal{I}^* = \arg\max_{\mathcal{I}} \log P(\mathcal{Y}, \mathcal{I}|\Theta),$$

   where $\mathcal{I}^*$ is the set of the searched interval sequences for all the training sequences.

   M step: Update the parameters based on the searched interval sequences in the E step:

   $$\Theta = \arg\max_{\Theta'} \log P(\mathcal{Y}, \mathcal{I}^*|\Theta').$$

This algorithm is easily extended to use multiple interval sequences that give relatively high likelihood score rather than to use only an optimal interval sequence.

6. Experimental Results

6.1 Evaluation on Simulated Data

To evaluate the proposed parameter estimation methods, we first used simulated sequences for training data, because it provides the ground truth of the estimated parameters. Multivariate sequences were generated from the system that had the same parameters as the system in Sect. 4, except that the state transition probability A_{31} was set to zero to generate non-cyclic sequences. In the sequences, three dynamic primitives were appeared in the temporal order of $D_1 \rightarrow D_2 \rightarrow D_3$, and only the length of the duration varied. Figures 8(a) and (b) shows an example of the generated interval sequence and the two-dimensional observation sequence, respectively.

The proposed clustering process was evaluated by the sequence in Fig. 8(b). First, the initial interval set was determined by zero-crossing points of the first-order difference as we described in Sect. 5.1.1. Then, the initial dynamical systems were identified from the interval set. The number of the initial dynamical systems was $N = 6$. Figure 8(c) shows the obtained interval sequences during the clustering process. The number of the dynamical systems was reduced

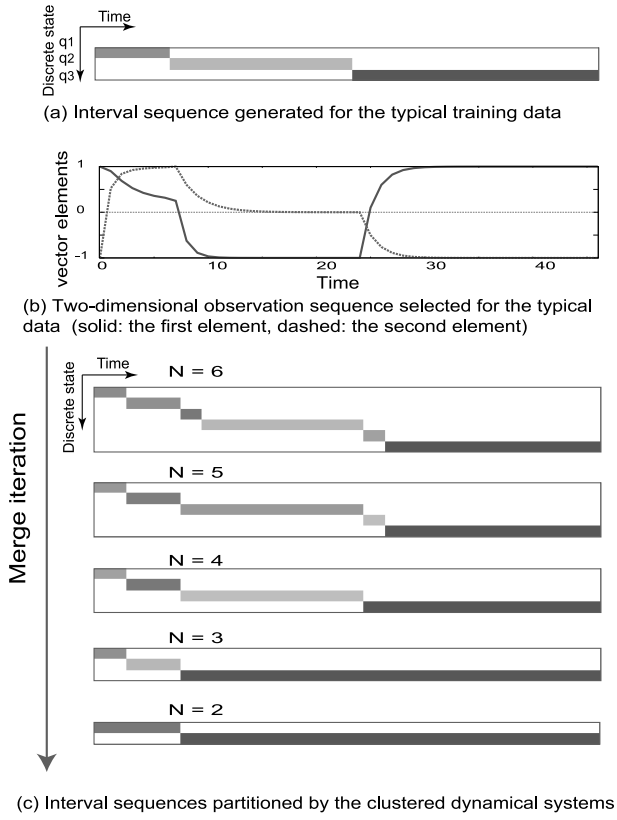


Fig. 8 The result of the clustering process applied to the typical training data shown in (b). The process was initialized by $N = 6$ and reduced the number of the dynamical systems (discrete states) by merging the nearest dynamical system pairs in each iteration steps.

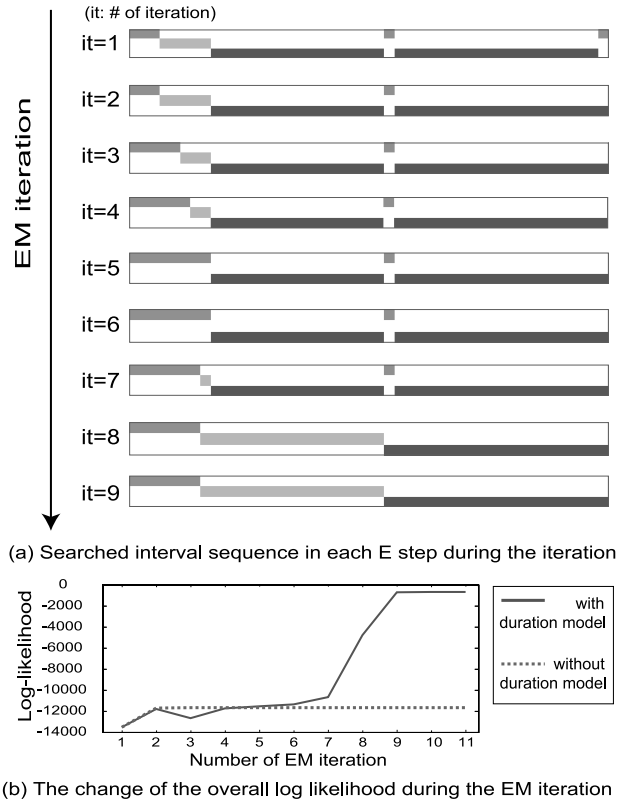


Fig. 9 The result of the refinement process via the EM algorithm using all the training data.

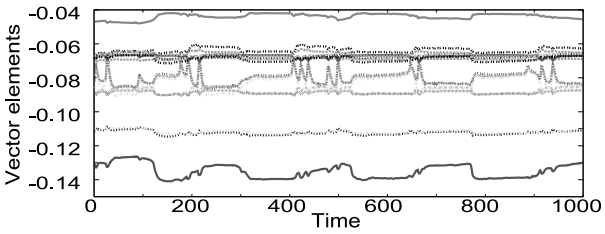
from $N = 6$ to $N = 2$ by the iterative merging process of nearest dynamical system pairs. In each iteration step, two interval sets that belong to the nearest two dynamical systems were also merged.

For the evaluation of the refinement process, we used the extracted dynamical systems in the clustering process. We manually selected $N = 3$ for the number of dynamical systems, which corresponds to the number of the original system that generated the typical sequence. Additional nine sequences were generated for the training data, and all the generated sequences including the typical sequence were used for the input of the EM algorithm. The algorithm was initialized by the parameters found in the clustering process. Figure 9(a) shows the searched interval sequences in the E step of each iteration step. We see that the partitioning of the intervals gradually converges to almost the same partitions as the original interval sequence shown in Fig. 8(a). The solid line in Fig. 9(b) shows the change of the overall log-likelihood of the system. We see that the algorithm almost converged at the ninth iteration. The dashed line in Fig. 9(b) shows the change of the overall log-likelihood when the duration distribution function $\hat{h}^{(ij)}$ was set to be uniform and the adjacent intervals was modeled to have no correlation. We see that the algorithm converges to a local optimum in this case.

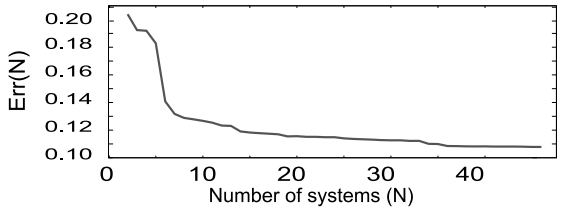
Consequently, the results of the clustering process become inaccurate if inappropriate sequences are selected for the typical training data. In spite of the inaccurate parameter estimation in the clustering process, the evaluation shows that the refinement process applied to all the training data recovers from the initial error. Especially, the meta-level features, such as modeling of duration lengths of intervals and relations between intervals, seem to work as constraints to the partitioning process in the EM algorithm.

6.2 Evaluation on Real Data

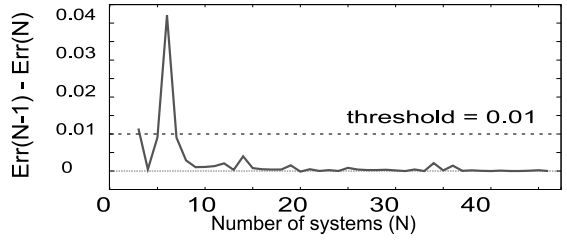
To evaluate the capability of the proposed method for real applications, we applied the clustering method to captured video data. A frontal facial image sequence was captured by 60 fps camera during a subject was smiling four times. Facial feature points were tracked by the active appearance model [27], [28], and eight feature points around the right eye were extracted. The length of the sequence was 1000. We then applied the clustering method to the obtained 16-dimensional vector sequence that comprised x- and y-coordinates of the feature points (both coordinate coefficients were plotted together in Fig. 10(a)). Figure 10(b) shows the overall model fitting errors between the original sequence Y and generated sequences $Y^{\mathrm{gen}}(N)$ by the extracted N dynamical systems. The error in each merge iteration step was calculated by the Euclidean norm: $Err(N) =$



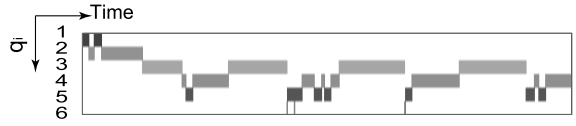
(a) Tracked feature points around the right eye



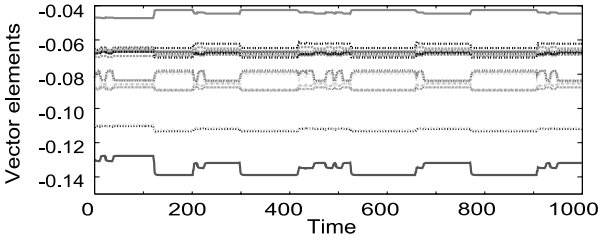
(b) Errors between original and generated sequences



(c) Differences between current and the previous errors



(d) Intervals partitioned by the dynamical systems (N = 6)



(e) Generated sequences from the clustered dynamical systems (N=6)

Fig. 10 Clustering results of the feature sequence around the right eye during a subject was smiling four times.

$\|Y - Y^{\text{gen}}(N)\| = \sqrt{\sum_{t=1}^{1000} \|y_t - y^{\text{gen}}(N)_t\|^2}$, which is the approximation of the overall log-likelihood score. The candidates of the number of the dynamical systems were determined as $N = 3$ and $N = 6$ by extracting the steps in which the difference $Err(N-1) - Err(N)$ exceeded given threshold (we used 0.01 in this experiment). The difference $Err(N-1) - Err(N)$ is shown in Fig. 10(c).

Figure 10(d) shows the intervals extracted from the clustering process when the number of dynamical systems was $N = 6$. Figure 10(e) shows the generated sequence from the extracted six dynamical systems, where each dynamical system was activated based on the partitioned intervals in Fig. 10(d). The dominant dynamical systems D_3 and D_4 correspond to the intervals in which the eye had been remain closed and open, respectively. The other dynamical systems such as D_5 correspond to the eye blink motion.

Consequently, the history of model fitting errors during the clustering process helps us to decide the appropriate number of dynamical systems.

7. Conclusion

This paper proposed a multiphase learning for the interval-based hybrid dynamical system that comprises a finite state automaton as an event-based model and multiple linear dynamical systems as clock-based models. The experimental results on the simulated and real data show that the proposed parameter estimation method successfully finds a set of dynamical systems that is embedded in the training data and the transition probabilities between the dynamics with a modeling of adjacent interval duration lengths.

For future works, we need to discuss the open problems of interval systems including the following areas.

(1) Model selection problems:

The selection of specific clock-based models depends on the nature of features and the design policies of users. We chose linear dynamical systems as clock-based models because linear dynamical systems are appropriate models to represent continuously changing human motion. HMMs can be more reasonable choice for modeling non-linear time-varying patterns such as consonants in human speech. Moreover, some motion generation researches design the total system as a clock-based system such as non-linear dynamical systems [29]. We need further discussion to give a guideline to select models.

(2) Modeling structures in multiple streams:

Modeling relations among concurrent multiple streams (e.g., temporal relations of motions among facial parts) and relations among multimodal data (e.g., lip motion and speech data) is one of the important objectives for the interval-based representation. A general modeling method of relations among intervals will be required to represent concurrent events.

(3) Modeling transitional process between dynamical systems:

The state in the continuous state space often changes discontinuously when the automaton changes the dynamical systems. To model coarticulated dynamics such as phonemes in speech data, we need transitional process modeling between two dynamical systems (e.g., the modulation of dynamics by the preceding dynamics).

(4) Grammatical structures of dynamics:

We exploited a simple finite state automaton for an event-based model in order to concentrate on modeling human body actions and motions, which have relatively simple grammatical structures compared to natural languages. To model languages such as human speech and signs, more complex grammatical structures such as N-gram models and context free grammars will be required with its parameter estimation method.

Acknowledgments

This work is in part supported by Grant-in-Aid for Scientific Research of the Ministry of Education, Culture, Sports, Science and Technology of Japan under the contract of 13224051 and 16700175.

References

[1] S. Nakaoka, A. Nakazawa, K. Yokoi, and K. Ikeuchi, "Leg motion primitives for a dancing humanoid robot," Proc. IEEE Int. Conference on Robotics and Automation, pp.610–615, 2004.
[2] C. Bregler, "Learning and recognizing human dynamics in video sequences," Proc. IEEE Conference on Computer Vision and Pattern Recognition, pp.568–574, 1997.
[3] A.V. Nefian, L. Liang, X. Pi, X. Liu, and K. Murphy, "Dynamic bayesian networks for audio-visual speech recognition," EURASIP J. Applied Signal Processing, vol.2002, no.11, pp.1–15, 2002.
[4] Y. Li, T. Wang and H.Y. Shum, "Motion texture: A two-level statistical model for character motion synthesis," SIGGRAPH, pp.465–472, 2002.
[5] M. Kamachi, V. Bruce, S. Mukaida, J. Gyoba, S. Yoshikawa, and S. Akamatsu, "Dynamic properties influence the perception of facial expressions," Perception, vol.30, pp.875–887, 2001.
[6] B.D.O. Anderson and J.B. Moor, Optimal Filtering, Prentice-Hall, 1979.
[7] J.F. Allen, "Maintaining knowledge about temporal interval," Commun. ACM, vol.26, no.11, pp.832–843, 1983.
[8] H.D. Huang, Y. Ariki, and M.A. Jack, Hidden Markov Models for Speech Recognition, Edinburgh Univ., 1990.
[9] Z. Ghahramani and G.E. Hinton, "Switching state-space models," Technical Report CRG-TR-96-3, Dept. of Computer Science, University of Toronto, 1996.
[10] V. Pavlovic, J.M. Rehg, T. Cham, and K.P. Murphy, "A dynamic bayesian network approach to figure tracking using learned dynamic models," Proc. IEEE Int. Conference on Computer Vision, pp.94–101, Sept. 1999.
[11] V. Pavlovic, J.M. Rehg, and J. MacCormick, "Learning switching linear models of human motion," Proc. Neural Information Processing Systems, 2000.
[12] H. Balakrishnan, I. Hwang, J.S. Jang, and C.J. Tomlin, "Inference methods for autonomous stochastic linear hybrid systems," Hybrid Systems, Computation and Control, Lecture Notes in Computer Science 2993, pp.64–79, 2004.
[13] M. Ostendorf, V. Digalakis, and O.A. Kimball, "From HMMs to segment models: A unified view of stochastic modeling for speech recognition," IEEE Trans. Speech Audio Process., vol.4, no.5, pp.360–378, 1996.
[14] G. Ferrari, M. Muselli, D. Liberati, and M. Morari, "A clustering technique for the identification of piecewise affine systems," Automatica, vol.39, pp.205–217, 2003.
[15] J. Roll, A. Bemporad, and L. Ljung, "Identification of piecewise affine systems via mixed-integer programming," Automatica, vol.40, pp.37–50, 2004.
[16] J.H. Kim, S. Hayakawa, T. Suzuki, K. Hayashi, S. Okuma, N. Tsuchida, M. Shimizu, and S. Kido, "Modeling of drivier's collision avoidance behavior based on piecewise linear model," Proc. IEEE Int. Conference on Decision and Control, vol.3, pp.2310–2315, 2004.
[17] M. Nishiyama, H. Kawashima, and T. Matsuyama, "Facial expression recognition based on timing structures in faces," Computer Vision and Image Media Technical Report (IPSJ SIG-CVIM 149), vol.2005, no.38, pp.179–186, 2005.
[18] A.P. Dempster, N.M. Laird, and D.B. Rubin, "Maximum likelihood from incomplete data via the EM algorithm," J.R. Statist. Soc. B, vol.39, pp.1–38, 1977.
[19] N. Yamada, T. Suzuki, S. Inagaki, and S. Sekikawa, "On the parameter estimation of stochastic switched ARX model and it's application," Proc. 18th Workshop on Circuits and Systems in Karuizawa, pp.269–274, 2005.
[20] R.O. Duda, P.E. Hart, and D.G. Stork, Pattern Classification, 2nd ed., Wiley-Interscience, 2000.
[21] P.V. Overschee and B.D. Moor, "A unifying theorem for three subspace system identification algorithms," Automata, vol.31, no.12, pp.1853–1864, 1995.
[22] G. Doretto, A. Chiuso, Y.N. Wu, and S. Soatto, "Dynamic textures," Int. J. Comput. Vis., vol.51, no.2, pp.91–109, 2003.
[23] T. Brants, "Estimating HMM topologies," Logic and Computation, 1995.
[24] B.H. Juang and L.R. Rabiner, "A probabilistic distance measure for hidden markov models," AT & T Technical Journal, vol.64, no.2, pp.391–408, 1985.
[25] D.K. Panjwani and G. Healey, "Markove random field models for unsupervised segmentation of textured color images," IEEE Trans. Pattern Anal. Mach. Intell., vol.17, no.10, pp.939–954, 1995.
[26] D.A. Langan, J.W. Modestino, and J. Zhang, "Cluster validation for unsupervised stochastic model-based image segmentation," IEEE Trans. Image Process., vol.7, no.2, pp.180–195, 1998.
[27] T.F. Cootes, G.J. Edwards, and C.J. Taylor, "Active appearance model," Proc. European Conference on Computer Vision, vol.2, pp.484–498, 1998.
[28] M.B. Stegmann, B.K. Ersboll, and R. Larsen, "FAME—A flexible appearance modelling environment," Informatics and Mathematical Modelling, Technical University of Denmark, 2003.
[29] M. Okada, K. Tatani, and Y. Nakamura, "Polynomial design of the nonlinear dynamics for the brain-like information processing of whole body motion," Proc. IEEE Int. Conference on Robotics and Automation, 2002.

Hiroaki Kawashima received the B.Eng. degree in electrical and electronic engineering and the M.S. degree in informatics from Kyoto University, Japan in 1999 and 2001, respectively. He is currently an assistant professor in the Department of Intelligence Science and Technology, Graduate School of Informatics, Kyoto University. His research interests include time-varying pattern recognition, human-machine interaction, and hybrid dynamical systems. He is a member of the IEEE Computer Society.

Takashi Matsuyama received the B.Eng., M.Eng., and D.Eng. degrees in electrical engineering from Kyoto University, Japan, in 1974, 1976, and 1980, respectively. He is currently a professor in the Department of Intelligence Science and Technology, Graduate School of Informatics, Kyoto University. His research interests include knowledge-based image understanding, computer vision, cooperative distributed vision, 3D video, and human-machine interaction. He received six best paper awards from Japanese and international academic societies including the Marr Prize at the International Conference on Computer Vision in 1995. He is a fellow of International Association for Pattern Recognition and a member of the Information Processing Society of Japan, the Japanese Society for Artificial Intelligence, and the IEEE Computer Society.