Multiobject Behavior Recognition Event Driven Selective Attention Method

Toshikazu Wada, *Member*, *IEEE*, and Takashi Matsuyama, *Member*, *IEEE*

Abstract—Recognizing multiple object behaviors from nonsegmented image sequences is a difficult problem because most of the motion recognition methods proposed so far share the limitation of the *single-object assumption*. Based on existing methods, the problem can be solved only by *bottom-up image sequence segmentation followed by sequence classification*. This straightforward approach totally depends on bottom-up segmentation which is easily affected by occlusions and outliers. This paper presents a completely novel approach for this task without using bottom-up segmentation. Our approach is based on *assumption generation and verification*, i.e., feasible assumptions about the present behaviors consistent with the input image and behavior models are dynamically generated and verified by finding their supporting evidence in input images. This can be realized by an architecture called the *selective attention model*, which consists of a *state-dependent event detector* and an *event sequence analyzer*. The former detects image variation (event) in a limited image region (focusing region), which is not affected by occlusions and outliers. The latter analyzes sequences of detected events and activates all feasible states representing assumptions about multiobject behaviors. In this architecture, event detection can be regarded as a verification process of generated assumptions because each focusing region is determined by the corresponding assumption. This architecture is sound since all feasible assumptions are generated. However, these redundant assumptions imply ambiguity of the recognition result. Hence, we further extend the system by introducing 1) *colored-token propagation* to discriminate different objects in state space and 2) integration of *multiviewpoint image sequences* to disambiguate the single-view recognition results. Extensive experiments of human behavior recognition in real world environments demonstrate the soundness and robustness of our architecture.

Index Terms—Behavior recognition, HMM, nondeterministic finite automata, selective attention mechanism, toke propagation, multiviewpoint image.

1 Introduction

Motion understanding is essential for a wide variety of vision applications, such as visual surveillance, human-machine interface, and virtual reality. Motion understanding problems can be categorized into the following three levels:

Physical motion analysis. Measure 3D or 2D geometric features of objects and analyze them along the time axis.

Object behavior recognition. Classify object motions into a set of behavior classes which emerge from constraints on the object's properties and its surrounding physical environment.

Object action understanding. Reason about intentions from motions, e.g., gesture analysis, sign language, flag semaphore, and so on.

In this paper, we address the visual behavior recognition problem, especially in cases when multiple objects are present in the scene.

1.1 Multiobject Behavior Recognition Problem

Since an object behavior observed by a camera is a spatio-temporal pattern, the *single object behavior recognition problem* can be defined as a temporal pattern classification problem. Much research has been done on this problem and some applications for gesture recognition [3], [6], [7], [11], hand sign recognition [4], [20], and lip reading [5], [21] are already realized. However, in most surveillance tasks, we cannot directly apply these existing methods, because multiple objects may be present in an image frame captured by a surveillance camera.

This *multiple object behavior recognition* problem is an extended behavior recognition problem under the *multiple object condition*:

- An unknown number of objects is present in each image frame.

This condition makes the original behavior classification problem quite difficult. In the single object behavior classification problem, a behavior belongs to a single class and all classes are known. This properly guarantees similarity-based classification which first compares similarity measures between the input behavior and all classes, then decides that the pattern belongs to the most similar class. If this method is directly applied to the multiobject behavior recognition problem, the following problem arises.

Since an input can be a mixture pattern of multiple behaviors, the class set should be all possible combinations of single behavior classes, which are essentially infinite[1] and nondisjoint. For example, suppose a problem classifying car

• *The authors are with the Department of Intelligence Science and Technology, Graduate School of Informatics, Kyoto University, Yoshida Hon-machi, Sakyo, Kyoto, 606-8501, Japan.*
E-mail: {twada, tm}@i.kyoto-u.ac.jp.

Manuscript received 21 Apr. 1999; revised 16 Mar. 2000; accepted 28 Mar. 2000.
Recommended for acceptance by R. Collins.
For information on obtaining reprints of this article, please send e-mail to: tpami@computer.org, and reference IEEECS Log Number 109640.

1. The observing area is limited and, hence, the number of combinations can be bounded by using domain specific heuristics.

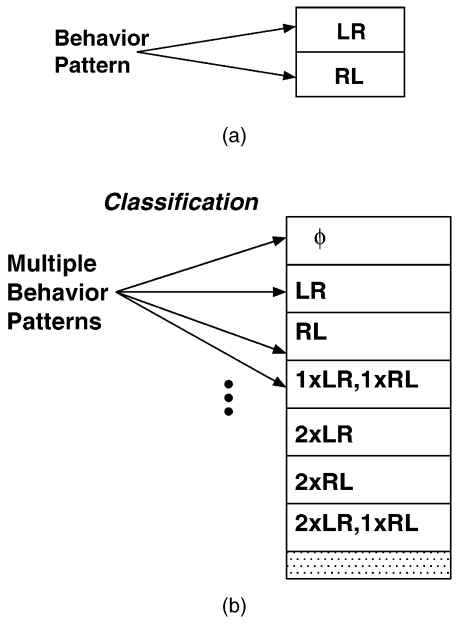

Fig. 1. Difference between single and multiple object behavior classification problems: In multiobject behavior classification problem, the events are essentially infinite and nondisjoint. (a) Single object behavior pattern classification. (b) Multiobject behavior classification.

behaviors to class "LR:" (from left to right) and class "RL:" (from right to left). As shown in Fig. 1a, the problem is simple for single car behaviors, but when multiple cars are present in an image frame, we have to assume infinite number of nondisjoint classes:

$$\{\phi, 1 \times LR, 1 \times RL, 2 \times LR, 1 \times LR \text{ and } 1 \times RL, 2 \times RL, \cdots\}.$$

See Fig. 1b.

That is, the multiple object condition converts the original *closed world classification problem* into an *open world classification problem* where the events are essentially infinite and nondisjoint. One may think that a probability density function can be defined on an infinite set Ω. However, for defining the function, Ω must have a continuous topology which cannot be defined in this problem. Also, elements of Ω are not disjoint, and hence, the total probability $P(\Omega) \neq 1$. Thus, Bayesian classification cannot be directly applied to this problem.

In the multiobject behavior classification problem, there is no *exclusive relationship* between different recognition results. For example, in a single object behavior recognition problem described above, if an input is classified into "LR," then it never belongs to "RL." However, in the multiobject behavior recognition problem, a recognition result that the input image sequence includes a behavior belonging to "LR" does not impose any constraints on other recognition results for the input, because many object behaviors may be included in the input. Thus, we cannot use exclusive constraints for behavior classification. For example, a large dissimilarity measure between an input and "LR" cannot be the evidence supporting classifying the input as "RL." In other words, we cannot use negative evidence, only positive evidence. This property is inherent in this problem.

1.2 Possible Approaches

The conventional approach to this problem is image-level segmentation, i.e., if all single object behavior patterns present in the input image sequence are successfully extracted, a simple classification method can be applied to each extracted pattern (Fig. 2a). Whether the input is a temporal pattern or not, this approach, i.e., *bottom-up segmentation followed by classification*, is the standard framework for multiobject recognition in computer vision. The problems arising in this approach are:

- How to guarantee the robustness of segmentation against occlusions and outliers.
- How to develop a consistent segmentation with classification under a single principle.

This paper presents a completely novel approach to solve this multiobject behavior classification problem without using bottom-up spatial and temporal segmentation. Our method is based on *assumption generation and verification (testing)*, i.e., by analyzing the input images, feasible assumptions on the present behaviors are dynamically generated, and the generated assumptions are verified by finding the evidence (image feature) supporting these assumptions (Fig. 2b).

Our principle is not to obtain the *optimal* solution but to *generate all assumptions* consistent with both image data and given behavior models. In other word, this principle is to compute all *feasible* solutions of given Constraint Satisfaction Problem (CSP) [1], where the constraints are the image data and the behavior models. We call this principle *feasibility*. A behavior analyzer based on this principle can be realized by an architecture named *selective attention mechanism* consisting of the following components:

State-Dependent Event Detector. An *event* is a predicate representing whether an image region[2] called *focusing region* is filled up by anomalous pixels obtained by background subtraction or not. (Details are described in Section 4.) Since the event detection is performed in a limited image region, it is not affected by image variation outside of the region (outliers). Each focusing region is switched according to the corresponding active states in the event sequence analyzer described below. The event can be regarded as a positive evidence of a certain behavior at a certain behavior stage.

Event Sequence Analyzer. A state transition model representing the order relationship between the behavior stages is necessary to generate the assumptions. We call the state transition model driven by the detected events the *sequence analyzer*. In this paper, we employ Nondeterministic Finite Automaton (NFA) as a sequence analyzer, because NFA is a simple example satisfying the following properties:

Instantaneousness. States are instantaneously activated whenever an input is injected. This is because 1) the focusing region depends on active states and 2) the start and the end of each behavior can be detected by monitoring state activation patterns without using bottom-up temporal segmentation.

2. In this paper, we will use a word "region" as a set of pixels.

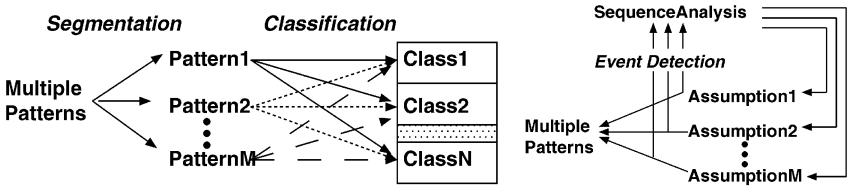

Fig. 2. Two approaches to solve multiple object behavior classification problem. (a) Segmentation+classification. (b) Assumption Generation+verification.

Pure-Nondeterminism. *All feasible states* can be simultaneously activated for each input to realize multicontext search. This property is *sufficient* for the multiple object behavior recognition and *necessary* to guarantee the feasibility. The reason why we use the word *pure-nondeterminism* instead of *nondeterminism* is that there are two types of state transition models:

- Nondeterministic as a model, but deterministic in optimization stage, e.g., HMM.
- Nondeterministic as both of a model and a state machine, e.g. NFA.

This architecture iterates the following loop:

1. **focusing region** $\rightarrow$ event detection
2. **event detection** $\rightarrow$ state transition (activation)
3. activated state $\rightarrow$ **focusing region**.

This loop produces mutual interactions between these two components, which tightly bind them.

1.3 State Transition Models

In this section, we examine suitability of some state transition models for our event sequence analyzer. The criteria are *instantaneousness* and *pure-nondeterminism*.

The most popular state transition model employed in previous motion recognition research (e.g., [3], [4], [5], [6]) is the Hidden Markov Model (HMM) [2]. The HMM finds the most likely state transition path for given data sequences. Although the HMM provides flexible classification, we cannot employ HMM as a sequence analyzer in our architecture. This is because:

- The HMM based classification requires an optimization process to find the most likely state transition path by using Viterbi or EM algorithm for a certain length of input sequence. It implies that the HMM-based classification does not produce instantaneous state transition.
- As a model, the HMM is a nondeterministic probabilistic state transition model, i.e., it does not determine the active state for each input before optimization. But in the optimization process, most HMM-based systems activate a single state for each input. It means the property of pure-nondeterminism is not satisfied. Recently, higher-order HMMs [8], [9], [10], [11] have been developed, which commonly have *compositional states*, i.e., they do not activate all possible states but a known number of multiple states for each input. This implies that N-context search can be realized if we know the number of objects N. Or, a higher-order HMM consisting of N basic HMMs can recognize M-object behaviors $(M \leq N)$ by thresholding the total probabilities of Markov chains obtained by optimization. Even by this extension, pure-nondeterminism is not satisfied, i.e., the necessary condition for feasibility cannot be satisfied.

The basic principle of HMM is *optimality*, i.e., finding the most likely Markov chain having the maximum probability from a bounded number of candidates. The higher-order HMMs mentioned above are based on an extended principle to find the optimal set of N-Markov chains under some constraints. For finding unknown numbers of Markov chains, we need a criterion to determine the number of chains. The Minimum Description Length (MDL) [12] criterion or Akaike Information Criterion (AIC) [13] can be used with HMMs to obtain the optimal solution having maximal probability. This approach is clearer than introducing a threshold. On the other hand, HMM-based motion recognition systems have to use a time window for temporal segmentation, which is necessary to handle endless input sequences. Temporal segmentation using time window may also affect the basic principle: Optimality and, hence, the MDL criterion or AIC should also be used for computing the optimal time window size and positioning for each behavior. However, optimal segmentation in spatio-temporal space for an endless input image sequence including an unknown number of multiple objects is very difficult. As a result, most designers introduce nonoptimal segmentation methods with HMM-based optimal classifiers for practical motion recognition systems. Nonoptimality in part of the system and optimality in the other mean that, as a whole, the system is nonoptimal and no longer rests on a clear principle of optimality.

Recently, the proposed time-delay neural network [19] can be another candidate for a temporal sequence analyzer, which has been successfully applied to phoneme classification [19], hand gesture recognition [20], and lip reading [21]. A TDNN observes temporal data within a small time window which slides over the input data while the network makes a series of local decisions. These local decisions are integrated in its hierarchically layered structure. A TDNN can extract and classify patterns from nonsegmented temporal sequences because it uses a sliding time window. Also, it can use back propagation learning. However, since a TDNN has a delay, it is not instantaneous and, hence, we cannot use it as a sequence analyzer. Since a TDNN is based on neural networks, its basic principle might be optimal

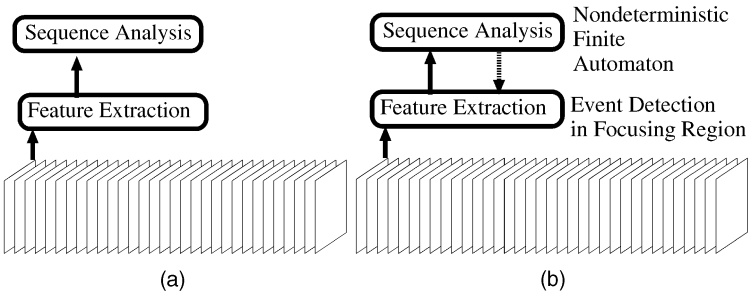

Fig. 3. Behavior recognition system architectures. (a) Bottom-up system. (b) Bottom-up and top-down system.

classification. However, there is no proof whether optimal temporal segmentation is realized by a TDNN or not.

The basic principle of our architecture is rather looser than optimality, i.e., feasibility. Our behavior analyzer produces all feasible assumptions for the input image sequence and behavior models. This is much easier than finding the optimal recognition result because it only performs multiplex spatio-temporal pattern matching and does not examine the optimality of the result. Our architecture can use any instantaneous and pure-nondeterministic state transition models. In this paper, we selected an NFA as a sequence analyzer, because of its simplicity and well-known properties. Although NFAs are well studied, taught in most basic computer science classes, and widely used, they are never used in any motion recognition research. One may think that all discrete state machines are sensitive and affected by noise. In spite of this, we employed an NFA as a sequence analyzer, because we can assume that the feasibility, i.e., multiple state activation, can compensate for the sensitivity.

1.4 Properties and Extensions

In this section, we show the properties of the selective attention mechanism from several viewpoints and discuss necessary extensions.

System Architecture. Most conventional systems are bottom-up systems (Fig. 3a), but selective attention mechanism has both bottom-up and top-down flows (Fig. 3b). It implies that the image-level processing can use temporal analysis results as well as image information.

Segmentation. The selective attention mechanism seems to be performing spatial and temporal segmentations. Spatial segmentation is realized by introducing a focusing region and all instantaneous state transition models can segment given temporal data. That is, the systematic combination of an event detector and a sequence analyzer produces this property.

Speed. The selective attention mechanism iterates simple image processing and state transition and, hence, the processing speed is fast.

Robustness. Event detection is sensitive to noise and lack of data because an event is a predicate which produces a binary value. In the case of single-context search, if an event value is flipped by noise from one to zero, then the system may lose track of the successive event sequence. However, multiple active states produced by nondeterministic state transition can compensate for the sensitivity. For robustness against outliers, it is clear that our system is not affected by random image variations outside of the focusing region (outliers). If a consistent outlier sequence similar with known behavior pattern appears in the image sequence, our architecture simply produces its corresponding assumption. Note that this assumption generation will not affect the "true" assumptions about other behaviors.

Soundness. This is directly connected with our principle. For example, when an object moving along the view direction is observed by a camera, our architecture will produce assumptions that a line of objects are moving along the view direction. This result can be justified because there is no evidence which negates this assumption.

Although multiple assumption generation contributes to the robustness and soundness, it also implies ambiguity, i.e., multiple active states involve two cases:

Unpreciseness. Active states generated by a single object.

Multiple objects. Active states generated by multiple objects.

To discriminate between these two cases, we introduce *colored-token propagation*, where a color represents an object. By propagating colored tokens on active states, different object behaviors can be discriminated in state space. In other word, we have to add a minimal object discrimination mechanism to produce recognition results from multiple assumptions.

Based on the selective attention mechanism incorporating the colored-token propagation, a robust and sound multibehavior recognition system can be designed. However, all appearance-based behavior recognition methods share the limitation that 3D object behaviors are projected onto 2D image plane and, hence, it is difficult to discriminate multiple objects along the viewing direction. To remove this limitation, we further extend the above system for multiviewpoint images.

In the following sections, we describe the behavior recognition method, multiviewpoint extension, practical design, and experimental results, and discuss the contributions we made and future work.

2 Behavior Recognition

In this section, we describe a selective attention mechanism for multiobject behavior recognition.

2.1 Selective Attention Mechanism

In image sequences of object behaviors belonging to a class, we can find an image variation sequence specifying the

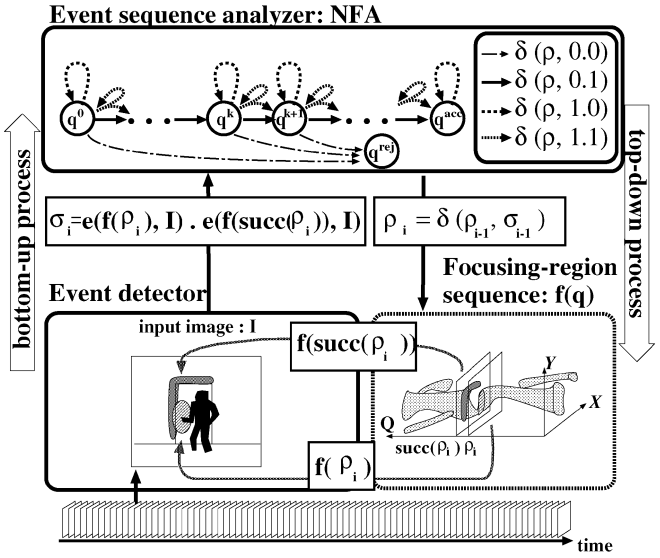

Fig. 4. Behavior identifier based on Selective Attention Mechanism.

behavior class. For example, when we open and go out through a door, we can find image variations at 1) the door knob, 2) the edge of the door, and so on.

If the camera is fixed, such image variations can be detected in specific image regions (*focusing region*). By using a temporal sequence of focusing regions specifying a behavior class, we can identify behaviors belonging to the class by sequentially detecting image variations (*events*) in those focusing regions.

In this case, focusing regions should be changed according to the current behavior stage. This can be realized by linking the focusing regions to the states in the sequence analyzer (NFA). That is, detected events activate states in the NFA, and the event detector changes its focusing regions according to the activated states. We call this bottom-up and top-down behavior identification the *selective attention mechanism* (Fig. 4). A behavior identifier based on the selective attention mechanism is described below:

A behavior identifier is a simple classifier that accepts image sequences in a behavior class and rejects those in other classes. The behavior identifier consists of 1) a focusing region sequence, 2) a sequence analyzer, and 3) an event detector:

Definition 1 (Focusing region sequence). *A focusing-region sequence $f(q)$ is a sequence of image regions where event detection is performed at state q. Formally, $f(q)$ is defined as a mapping from state space Q to power set $\mathcal{B}(I)$ of image space I consisting of pixels:*

$$f:\ Q \mapsto \mathcal{B}(I),$$

where $I = \{(x,y) | X_{min} \le x \le X_{max}, Y_{min} \le y \le Y_{max}\}$, and $X_{min}, X_{max}, Y_{min}, Y_{max}$ are the minimum and maximum coordinate values for x and y.

A practical method to obtain a focusing region sequence from training samples is described in Section 4.

Definition 2 (Sequence Analyzer). *The sequence analyzer (NFA) is represented by $(Q, q^0, \Sigma, \delta, F)$, where*

- *Q : finite set of states,*[3] *$Q = \{q_0, q^1, \cdots, q^m, q^{rej}\}$,*
- *q^0: initial state ,$q^0 \in Q$,*
- *Σ: finite set of event codes,*
- *δ: state transition function, $\delta(q, \sigma) : Q \times \Sigma \mapsto Q$,*
- *F: finite set of final states, $F = \{q^m, q^{rej}\}$.*

When an input sequence $\sigma_i \in \Sigma (i = 0, 1, \cdots)$ is given, the active state ρ_i at the ith input is defined by δ as:

$$\begin{cases} \rho_0 &= q^0, \\ \rho_{i+1} &= \delta(\rho_i, \sigma_i), \end{cases} \tag{1}$$

Note that:

The state transition is not applied to those states in the final state set F.

δ may have multiple values for a single input. This is called nondeterministic state transition.

A state q^m represents the final stage of a behavior, where the input sequence is accepted. Hence, q^m is also represented by q^{acc}.

A state q^{rej} represents that the input is rejected.

Definition 3 (Event Detector). *An event is a predicate representing the presence of an image variation in a focusing region. An event within a focusing region f of image I is denoted by $e(f, I) (\in \{0, 1\})$. If an event is detected, $e(f, I) = 1$, otherwise, $e(f, I) = 0$.*

*An event detector checks image variations within multiple focusing regions corresponding to successive states and combines obtained events into an **event code**. For example, an event code of length two at state ρ is represented by:*

$$\sigma = e(f(\rho), I) \cdot e(f(succ(\rho)), I), \tag{2}$$

where "$\cdot$" represents the event combination operator, and $succ(\cdot)$ the successor function, i.e., if $\rho = q^k$, then $succ(\rho) = q^{k+1}$.

The event detection method is detailed in Section 4.

Definition 4 (Behavior identifier). *The components described above are assembled into a behavior identifier in the following manner:*

- *The **event code** is used as an input to the sequence analyzer,*
- *The **active states** of the sequence analyzer determine the focusing region,*
- *The **focusing region** is used in the event detector.*

For example, in the case that the event code length is two, the active state ρ_i at the ith input is defined as:

$$\begin{cases} \rho_0 &= q^0, \\ \rho_{i+1} &= \delta(\rho_i, \sigma_i), \\ \sigma_i &= e(f(\rho_i), I_i) \cdot e(f(succ(\rho_i)), I_i). \end{cases} \tag{3}$$

If a final state q^{acc} is activated, then the input sequence is identified as the behavior.

For an event code length of two and $\rho_i = q^k$, ρ_{i+1} can be determined by $e(f(q^k), I_i)$ and $e(f(q^{k+1}), I_i)$, because an event $e(f(q^k), I_i) = 1$ represents the evidence for $\rho_{i+1} = q^k$, and $e(f(q^{k+1}), I_i) = 1$ represents $\rho_{i+1} = q^{k+1}$. Such homogeneous state transitions can be described by a state

3. A state sequence $(q^1, \ldots, q^m)$ represents an abstracted time axis, i.e., each state in this sequence corresponds to a behavior stage. See Section 4.

TABLE 1
State Transition Table at $\rho_i = q^k$ for Event Code Length = 2

$e(f(q^k), I_i) \cdot e(f(q^{k+1}), I_i)$	ρ_{i+1}
$0 \cdot 0$	q^{rej}
$0 \cdot 1$	q^{k+1}
$1 \cdot 0$	q^k
$1 \cdot 1$	q^k or q^{k+1}

transition table. A simple state transition table is shown in Table 1.

Here, an event code $\sigma_i = 1 \cdot 1$ will cause a nondeterministic state transition from q^k to both q^k and q^{k+1}. This property enables parallel tracking of all feasible event sequences. If all the past event codes are $1 \cdot 0$ or $0 \cdot 1$ and the current event code is $0 \cdot 0$, there will be no active state in the sequence analyzer. However, since the focusing region obtained by our learning method shown in Section 4 is always smaller than the object apparent size at each behavior stage for all training samples, at least one focusing region will be in the object silhouette on the image plane at each behavior stage. As well, for continuous object behaviors, the focusing regions of neighboring states are similar. Hence, in practice, two or more focusing regions are included in each object silhouette and $1 \cdot 1$ event codes are frequently produced. This situation will produce redundant active states, but also, a $0 \cdot 0$ event code will be produced by useless active states without evidence, which will inactivate the useless states. Then, the combination of *redundant state activation* and *useless state inactivation* will produce appropriate system behavior because useless states are pruned.

2.2 Colored-Token Propagation

By using the selective attention mechanism, multiobject behaviors can be identified from a single image sequence. In this mechanism, however, multiple states are simultaneously activated for a single object. Since each active state represents a candidate for an object behavior, multiple active states can be interpreted as 1) multiple objects at different behavior stages, or 2) a single object at uncertain behavior stages. This ambiguity prevents us from recognizing the number of objects present in the scene.

To disambiguate between these interpretations and count the number of objects, we have to link active states to objects. In this paper, we introduce colored-tokens to represent this correspondence where a color is regarded as an object. By assigning and propagating colored tokens, different object behaviors can be discriminated in the NFA.

2.2.1 Token Assignment

In the case of continuous object behaviors, since the focusing regions of neighboring states are similar, neighboring contiguous states are simultaneously activated for a single object. Hence, the contiguous active states are considered to be activated by an object.

Based on this heuristic, colored token assignment can be designed as follows:

A *neighboring active state set* C is a subset of active states connected by state transition function δ, which satisfies:

$$\forall_{\sigma \in \Sigma}(\rho \in C \Rightarrow (\delta(\rho, \sigma) \in C)) \vee (\delta(\rho, \sigma) \in C \Rightarrow \rho \in C). \quad (4)$$

An active state set P can be decomposed into disjoint sets of neighboring active states C^k, i.e., $P = \cup C^j$, and $C^k \cap C^j = \phi$ for any $C^k \neq C^j$.

By marking all states in C^k by a token having a single color $z^k \in Z$, active states can be identified with objects, where Z represents the integer set, and $C^k \neq C^j \Longleftrightarrow z^k \neq z^j$.

2.2.2 Token Propagation

In the token assignment procedure described above, however, the number of token colors may vary with time, even if the number of objects is constant. To maintain the consistency of the assigned token colors along the time axis, assigned tokens should be propagated simultaneously with the state transition.

Since state transitions represent progress of object behaviors, tokens should also be propagated by the state transition function δ. That is, tokens assigned to C_i^j are propagated to those C_{i+1}^ks satisfying:

$$\left(\bigcup_{\sigma \in \Sigma} \bigcup_{\rho \in C_i^j} \delta(\rho, \sigma) \right) \cap C_{i+1}^k \neq \phi, \quad (5)$$

where ϕ is empty set, and C_i^j represents a neighboring active state set at ith input. In this case, we call that C_i^j has a *link* to C_{i+1}^k.

If C_i^j has a single link, all tokens assigned to C_i^j are simply propagated through the link. But, when C_i^j has multiple links, we should not propagate the copies of tokens through those links because any active states in different C^ks should not have the same token simultaneously.

Based on the discussion above, tokens should be propagated as follows (Fig. 5):

- When C_i^j has no links, tokens are discarded.
- When C_i^j has a single link, tokens are propagated through the link.
- When C_i^j has multiple links, tokens are divided into disjoint sets, which are distributed through the links. If the number of links is greater than that of assigned tokens, new tokens are generated after the propagation. Any division minimizing token generation is applicable[4].

Based on the discussion above, colored tokens are dynamically assigned and propagated by the following procedure:

Step 1. For each $C_i^j \in P_i$, if no token is assigned, a new token is generated and assigned to it.

Step 2. Compute state transitions for I_i and form $\{C_{i+1}^k\}$ from activated states ρ_{i+1}.

Step 3. Propagate tokens from $\{C_i^j\}$ to $\{C_{i+1}^k\}$ through the links.

Step 4. Count the token colors at q^{acc}. Increment i and go to **Step 1**.

4. This procedure doesn't guarantee the tracking of independent objects along the time-axis. However, it is capable of counting the number of objects.

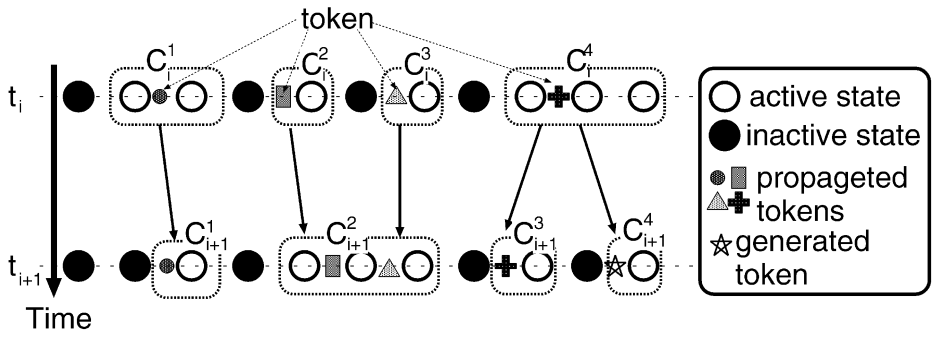

Fig. 5. Token propagation patterns on active states in NFA.

2.3 Behavior Classification

To recognize object behaviors, we have to classify behavior patterns. In the multiple behavior classification problem, a classification result does not exclude other results because there can be multiple objects. This implies that no inhibition mechanism should be employed in the classifier. Hence, multiple behavior classification can simply be realized by parallel behavior identification.

This classification mechanism can be designed by introducing an initial state q^0 and ϵ-state transitions[5] from q^0 to initial states of independent identifiers as shown in Fig. 6.

The classifier constructed in this manner can be regarded as an NFA and event detectors. In this NFA, when the number of active states increases, the processing speed slows down. This property prevents us from realizing real-time behavior recognition system. Fortunately, both the NFA and the event detectors can be transformed into an equivalent deterministic model that is not affected by the number of activated states as described below.

An NFA can be converted into its equivalent deterministic finite automaton (DFA) by a systematic and formal method. The details of this method is described in typical textbooks, e.g., [16]. An example of this conversion is illustrated in Fig. 7. In this figure, we can notice the converted DFA has states which consist of combinations of original states in the NFA. Also, the focusing regions of the event detector can be reorganized by the set operation according to the transformation from NFA to DFA.

3 Multiviewpoint Behavior Recognition

Appearance based behavior recognition methods share the limitation that 3D objects along the viewing direction are projected to similar regions on the 2D image plane, and it is difficult to distinguish these objects only by analyzing the image. In this case, many states in the NFA are activated and the number of objects cannot be recognized correctly even if we apply colored token propagation.

A simple approach to solve this problem is to extend the behavior recognition method to use images from multiple viewpoints. To recognize object behaviors from multiviewpoint images, we have to integrate information across views. This integration can be categorized into the following three levels (Fig. 8):

Image-level integration. Multiviewpoint images are combined into a single image and the recognition system takes it as input.

Event-level integration. Independently detected event codes from multiviewpoint images are integrated into an event code, which drives a sequence analyzer.

State-level integration. Independent behavior identification systems mutually interact to inhibit redundant state transitions.

Image-level integration is the simplest method for utilizing multiviewpoint images. But, this method cannot verify the cooccurrence of events in different viewpoint images, i.e., this method does not take into account that different events in multiple images can actually be different views of the same event.

In event-level integration, event codes detected at different viewpoints are integrated by a bitwise "AND" operation, which checks the cooccurrence of events corresponding to an event in 3D space.[6]

State-level integration inhibits redundant active states according to *feasible state combinations* that represent valid combinations of states in identifiers at different viewpoints. Feasible state combinations can be learned from training samples as shown in Section 4. When the time intervals corresponding to states are synchronized among the identifiers at different viewpoints, this integration is equivalent to event-level integration. That is, this integration includes the event-level integration as a special case.

The feasible state combinations can be represented by a set of points in state product space of identifiers at different viewpoints (Fig. 9). This integration can be regarded as an augmentation of the domain of state transition and token propagation from states to feasible state combinations.

4 A Practical Design

By using background subtraction, the anomalous region at time t can be obtained as:

$$a(t) = \{(x, y) \mid |I(x, y; t) - I_{bg}(x, y)| > \nu\}, \tag{6}$$

5. The ϵ-state transition means a state transition caused by null input.

6. "An event in 3D space" means a variation of 3D scene that is detected as simultaneous image variations in multiviewpoint images. In this detection, we do not use explicit geometric constraints between the multiple viewpoints because the simultaneous image variation pair implicitly includes the geometric constraints which can be learned from training data.

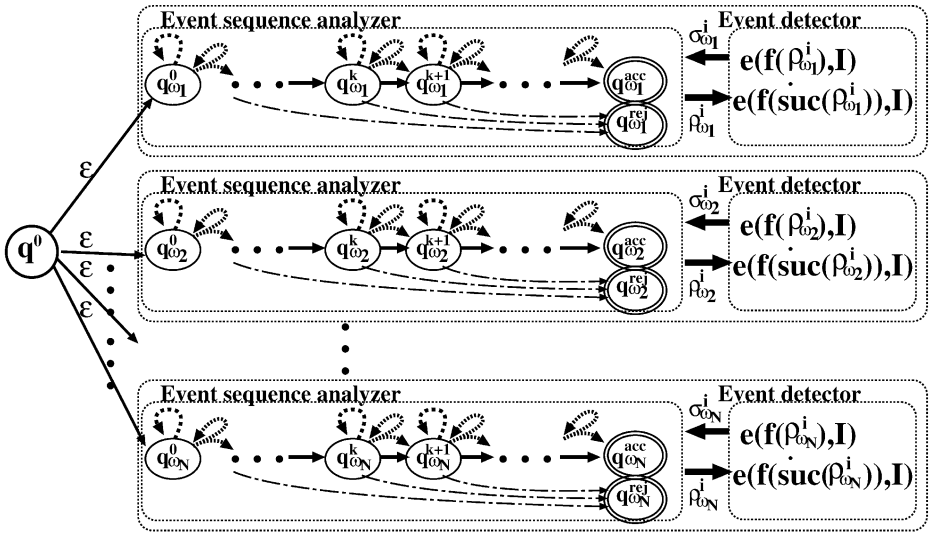

Fig. 6. Behavior classifier consists of different identifiers.

where $I(\cdot,\cdot;t)$ represents the input image at time t, $I_{bg}(\cdot,\cdot)$ a background image, and ν is a threshold. Here, we describe a practical event detection and learning method using anomalous features.

Event detection. An event can be detected when anomalous pixels fill the focusing region, which is defined as:

$$e(f, I(t)) = \begin{cases} 1, & f = \phi \; or \; \frac{|f \cap a(t)|}{|f|} > \theta \\ 0, & otherwise, \end{cases} \tag{7}$$

where θ $(0 < \theta \leq 1)$ represents a threshold, f the focusing region, and ϕ the null focusing region.

Focusing region at initial state. We assign the null focusing region ϕ to state q^0. This guarantees that the current state set always includes the initial state q^0 as a gatekeeper which always checks events at $f(q^1)$ to produce event codes. Hence, we can recognize successive behaviors by using this configuration.

Learning a focusing region sequences. A focusing-region sequence can easily be acquired from training samples of anomalous regions $a(t)$. When n samples of anomalous region sequences $a^i(t)(i = 1, 2, \cdots, n)$ in a class are given, the time axes of these samples can be normalized so as to maximize the following matching measure by dynamic programming:

$$\int \frac{|a(t) \cap a^i(\tau_i(t))|}{|a(t) \cup a^i(\tau^i(t))|} dt, \tag{8}$$

where $a(t)$ is the standard sample of the class, τ^i is a monotone increasing function, and $|\cdot|$ represents the number of pixels. The result of this process depends on having a good standard sample $a(t)$. In practice, to get a good result, a long and wide $a(t)$ should be used as a standard sample. A longer $a(t)$ than $a^i(t)$ for all t will not cause frame merging in $a^i(\tau_i(t))$, i.e., $\tau_i(t) > t$, and a wider $a(t)$ than $a^i(t)$ can keep the intersection result $|a(t) \bigcap a^i(\tau_i(t))|$ bigger.

From the normalized anomalous region sequences, the common anomalous region sequence among the training samples $f(t)$ can be extracted as:

$$f(t) = \bigcap_{i=1}^{n} a^i(\tau^i(t)). \tag{9}$$

The common anomalous region sequence is represented in the normalized time axis t. Intervals along this time axis

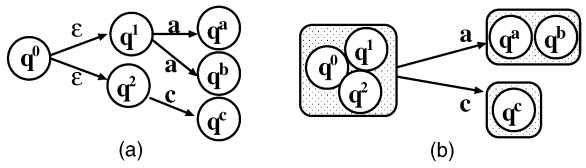

Fig. 7. An example of transformation from NFA to equivalent DFA. (a) NFA. (b) Equivalent DFA.

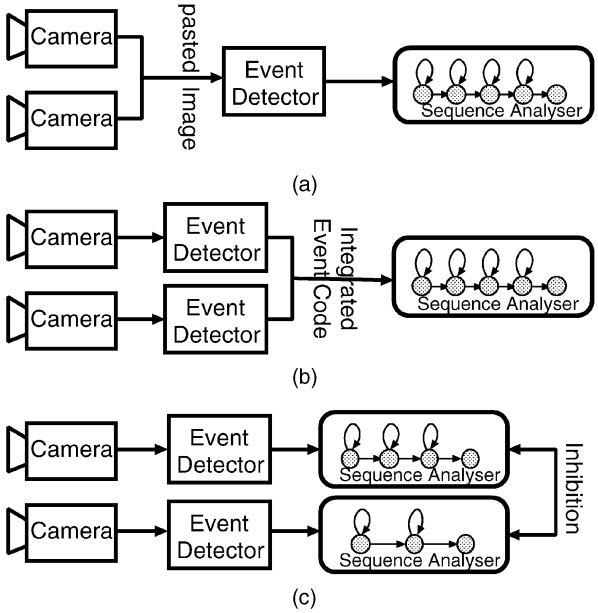

Fig. 8. Three levels of integration. (a) Image-level integration. (b) Event-level integration. (c) State-level integration.

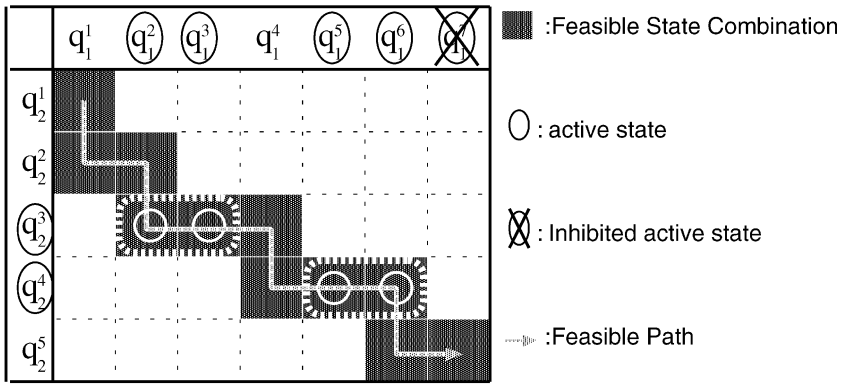

Fig. 9. Feasible state combinations in 2D state product space consists of two 1D state spaces.

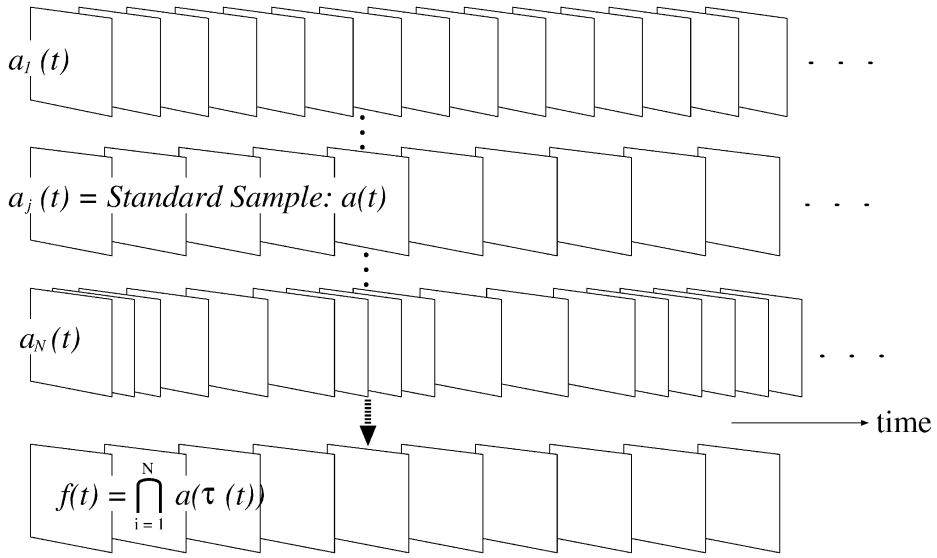

Fig. 10. Method to obtain *common anomalous region sequence:* A common anomalous region sequence is obtained by intersection of temporally normalized training samples (anomalous region sequences).

corresponds to the states of an NFA. If a state q corresponds to a given time interval $t_s \leq t < t_e$, the focusing region $f(q)$ at state q is computed as:

$$f(q) = \bigcap_{t=t_s}^{t_e-1} f(t) \tag{10}$$

For determining these time intervals, we employ a method described below:

This method is designed to minimize the number of active states by providing two criteria:

1. Focusing region should not be empty and
2. maximize the difference between neighboring focusing regions. The algorithm is described below:

Step 1: (initialization) $j = 0$, $q_j^i = t^i$, $N_j = T$, where T is the number of sampling points on normalized time axis t.
Step 2: If N_j is less than or equal to the specified number of states, then **stop**. Otherwise, find i which maximizes the similarity measure between $f(q_j^i)$ and $f(q_j^{i+1})$:

$$\frac{|f(q_j^i) \bigcap f(q_j^{i+1})|}{|f(q_j^i) \bigcup f(q_j^{i+1})|}.$$

Step 3: If the maximum similarity measure is less than a given threshold Θ, then **stop.** Otherwise, $N_{j+1} = N_j - 1$ and merge $f(q_j^i)$ and $f(q_j^{i+1})$ by the following equation:

$$f(q_{j+1}^k) = \begin{cases} f(q_j^k), & k < i \\ f(q_j^k) \bigcap f(q_j^{k+1}), & k = i \\ f(q_j^{k+1}) & k > i \end{cases}. \tag{11}$$

Step 4: $j = j + 1$, and go to **Step 2.**

The resulting states, their number, and corresponding focusing regions are q_j^i, N_j , and $f(q_j^i)$, respectively.

Learning Feasible State Combinations. For multiviewpoint behavior recognition, feasible state combinations must be learned from training samples. By using standard samples of $a^{cam1}(t), \cdots, a^{camN}(t)$ for the same behavior where the time axes are synchronized, we obtain common anomalous region sequences $f^{cam1}(t), \cdots, f^{camN}(t)$. Note that these sequences share a common time axis. When we compute focusing region sequences $f^{cami}(q)$ using above described method, we can refer to the corresponding time intervals and, hence, the feasible state combinations can be obtained by coupling states whose time intervals are overlapping on the common time axis. The time intervals

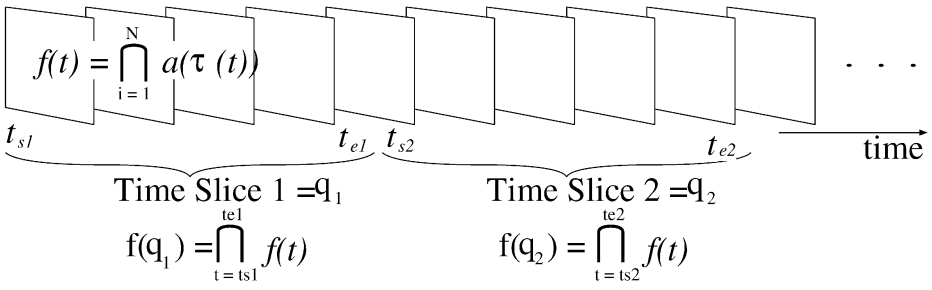

Fig. 11. Method to obtain a *focusing region sequence* from a common anomalous region sequence.

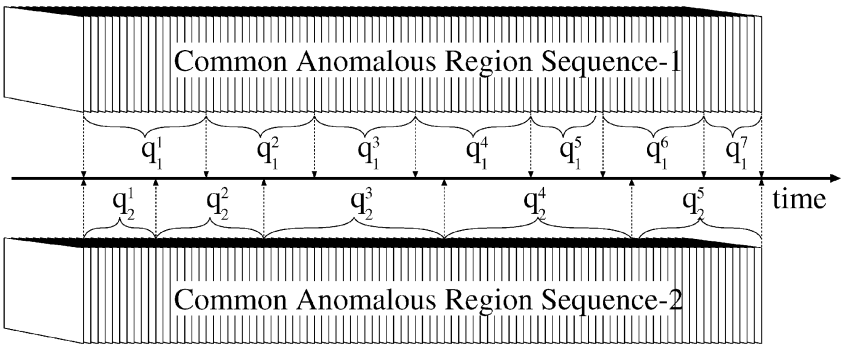

Fig. 12. Feasible state combinations from two common anomalous region sequences: From these time intervals, feasible state combinations shown in Fig. 9 will be obtained.

corresponding to the feasible state combinations in Fig. 9 is illustrated in Fig. 12.

5 Experiments

Here, we show experimental results of focusing region learning, recognizing human behaviors entering and exiting through a door by using one and two cameras, and total recognition performance. These behaviors are named "enter" and "exit." For each behavior class, 20 training image sequence pairs, i.e., 40 image sequences of single object behaviors taken by two cameras (camera 1 and camera 2), are used for learning focusing regions. Examples of these training data captured by camera1 are shown in Fig. 13.

5.1 Learning

The anomalous regions are obtained by background subtraction with fixed intensity difference threshold 20. By applying temporal normalization and intersection, the common anomalous region is computed. The focusing region is generated from the common anomalous region sequence by intersecting them within the time interval obtained by the algorithm shown in Section 4 with the minimal number of states: 28 and the similarity threshold Θ: 10 percent. The resulting focusing region sequences for camera 1 are shown in Figs. 14 and 15. In this process, feasible state combinations in state product spaces for both behaviors are also obtained. The result is shown in Fig. 16.

5.2 Recognition Example by a Single View

An example of the test data taken by a single camera and its corresponding state transitions are shown in Figs. 17 and 18, respectively. In Fig. 17, two persons enter into the room. These people walk along the view direction, then turn, and walk right. Its corresponding state transitions of "enter" and "exit" are shown in Fig. 18. In this figure, the horizontal and vertical axes represent time and states from initial to final, respectively. The gray and black regions represent activated states with different token colors. This figure shows that while these persons move along the viewing direction, almost half of the "enter" automaton states are activated. This corresponds to an assumption that many persons are entering the room. While the persons move from left to right, their corresponding thin state transition paths appear from the thick activated states. One can find that a new token color is generated when the first person starts moving from left to right. On the other hand, states of "exit" automaton are almost quiet. A small number of states near the initial state are activated when the persons reach the right side of the image frame because the anomalous region at the end of "enter" is similar with that of "exit" at the beginning. By counting token colors at the final states of these automata, the system recognizes that two "enters" have occurred.

5.3 Recognition Example by State-Level Integration Using Two Views

A test data and its corresponding recognition result using two cameras are shown in Figs. 19 and 20. In this test data, two "enters" and two "exits" are performed with some outliers. Fig. 20 shows the result of state-level integration, which shows four state transitions corresponding to the combinations of {"enter," "exit"} and {"camera 1," "camera 2"}. In this figure, token colors are omitted, but the activated states and inhibited states are discriminated by black and gray regions, respectively. The state inhibition is done by checking the co-occurrence of active states of automata corresponding to "camera 1" and "camera 2" using feasible state combinations shown in Fig. 16. The state transitions without inhibition (black and gray regions) are quite messy, but the inhibited state transitions (black

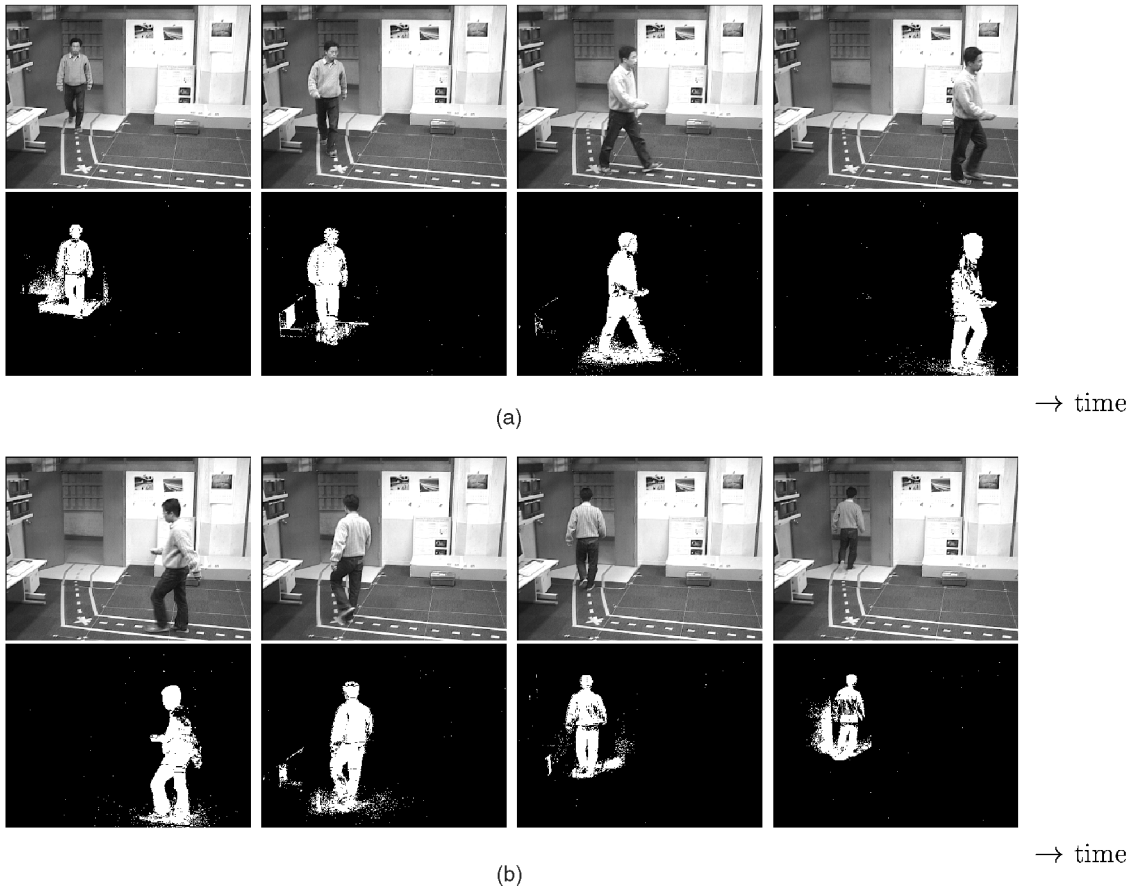

Fig. 13. Examples of training data: Top and bottom in each behavior are observed gray-level images and anomalous regions obtained by background subtraction with intensity threshold 20. (a) "Enter." (b) "Exit."

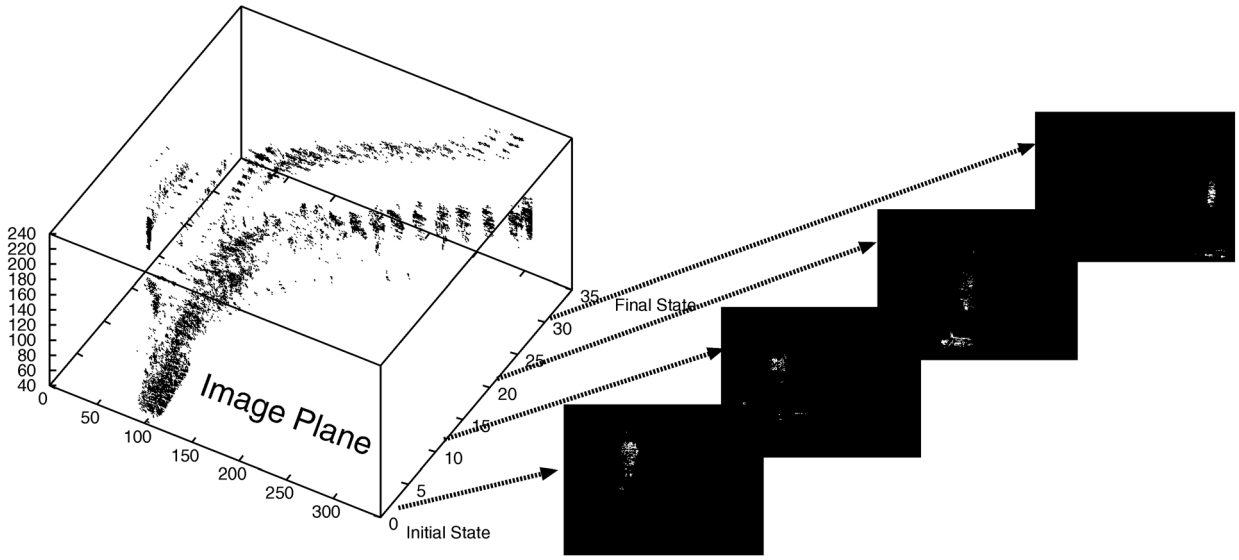

Fig. 14. Focusing region sequence of "enter" for camera 1: Left figure shows focusing region sequence in a 3D space spanned by image space and state space and right images are the slices at certain states.

regions) are much cleaner, and the system successfully recognizes correct results: two "enters" and two "exits."

5.4 Total Performance Evaluation

For measuring the total performance, we used 60 test data, i.e., 120 image sequences of multiple-object behaviors. Note that 1) the 20 training data and 60 test data are disjoint and 2) there is no test data which consists of a single object behavior or nonoverlapped multiple object behaviors. Possible methods are nonintegrated classifications using 1) camera 1, 2) camera 2, and integrated classifications at 3) image level, 4) event level, and 5) state level. These methods are applied to all of the test data.

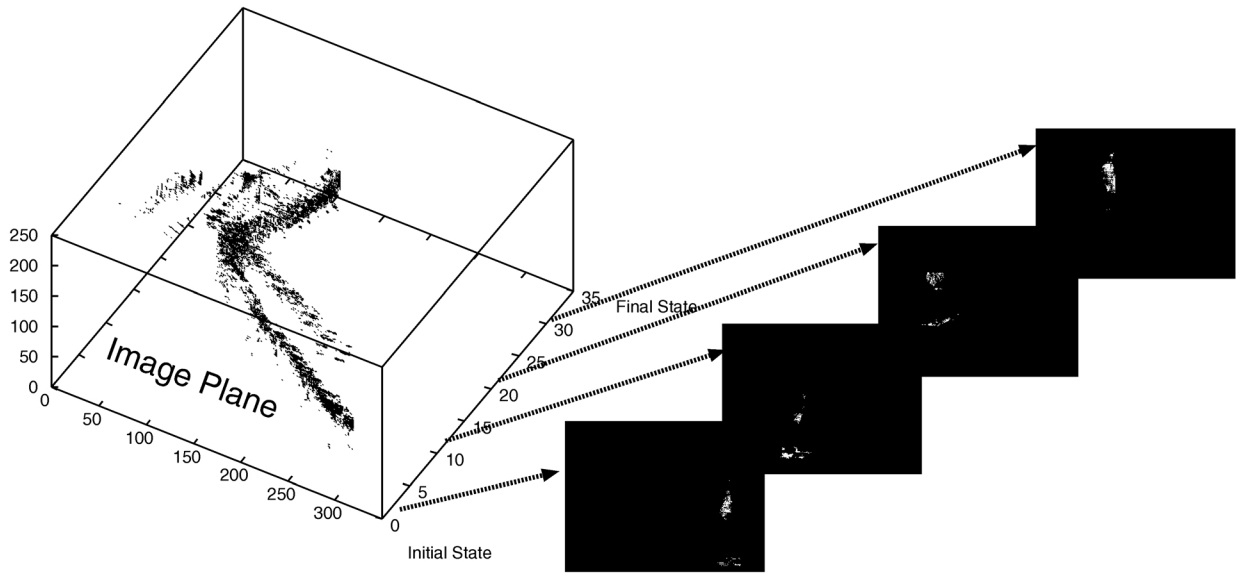

Fig. 15. Focusing region sequence of "exit" for camera 1: Left figure shows focusing region sequence in a 3D spanned by image space and state space and right images are the slices at certain states.

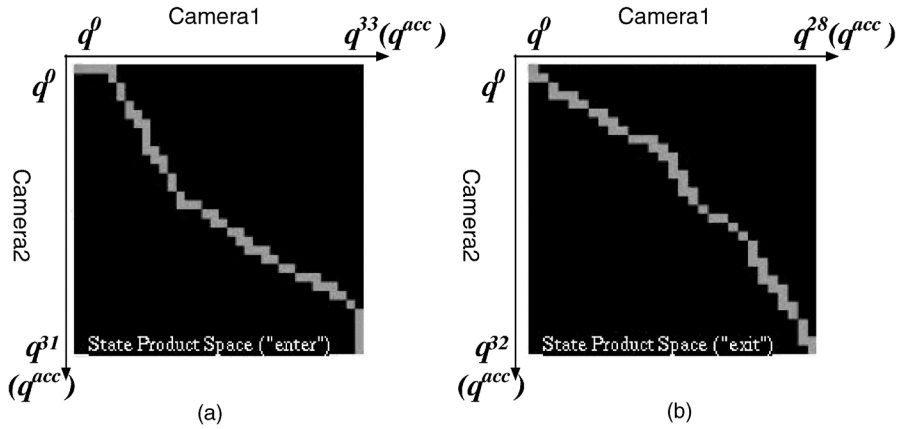

Fig. 16. Obtained state product spaces from training samples for (a) "enter" and (b) "exit": In each graph, the horizontal and vertical axes represent state spaces for camera 1 and 2, respectively. The gray region shows feasible state combinations.

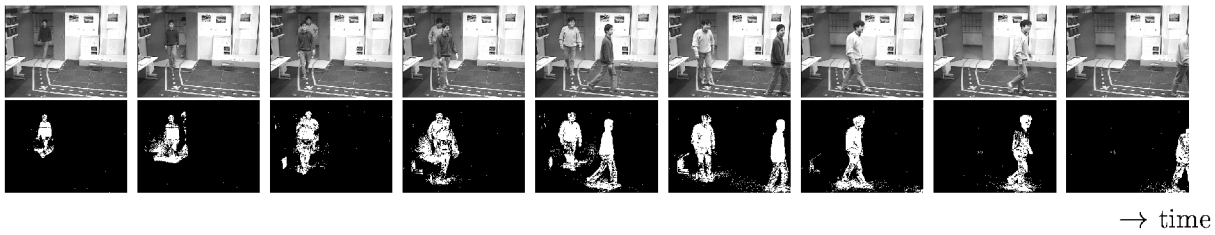

Fig. 17. An example of 2 "enter"s: Top and Bottom are oberseved gray-level images and anomalous regions obtained by background subtraction with intensity threshold 20.

The classification results are summarized in Fig. 21. In each graph, the vertical axis represents the number of data which are correctly recognized in both senses of "classification" and "number of objects." The horizontal axis represents the threshold θ for event detection. From these graphs, we can notice that event and state-level integrations are more effective than the others. State-level integration includes event-level integration as a special case. However, the latter is slightly superior to the former in this result. One explanation for this might be that each class has been excessively specialized at the learning phase in state-level integration because of its flexibility. In other words, 20 training data are insufficient for generalizing these behavior classes for state-level integration.

6 Discussions and Future Works

Since the selective attention mechanism is related to many other concepts and there still remain unsolved problems, we will discuss them to clarify future work.

The most distinguishing point of this architecture from other existing method is the difference between the

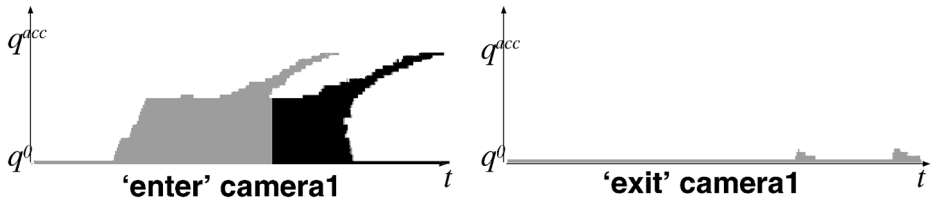

Fig. 18. State transitions of two NFAs corresponding to "enter" and "exit" for input shown in Fig. 17: The horizontal and vertical axes represent time and state space from initial (bottom) to final (top) states. The gray and black regions correspond to the active states having different colors.

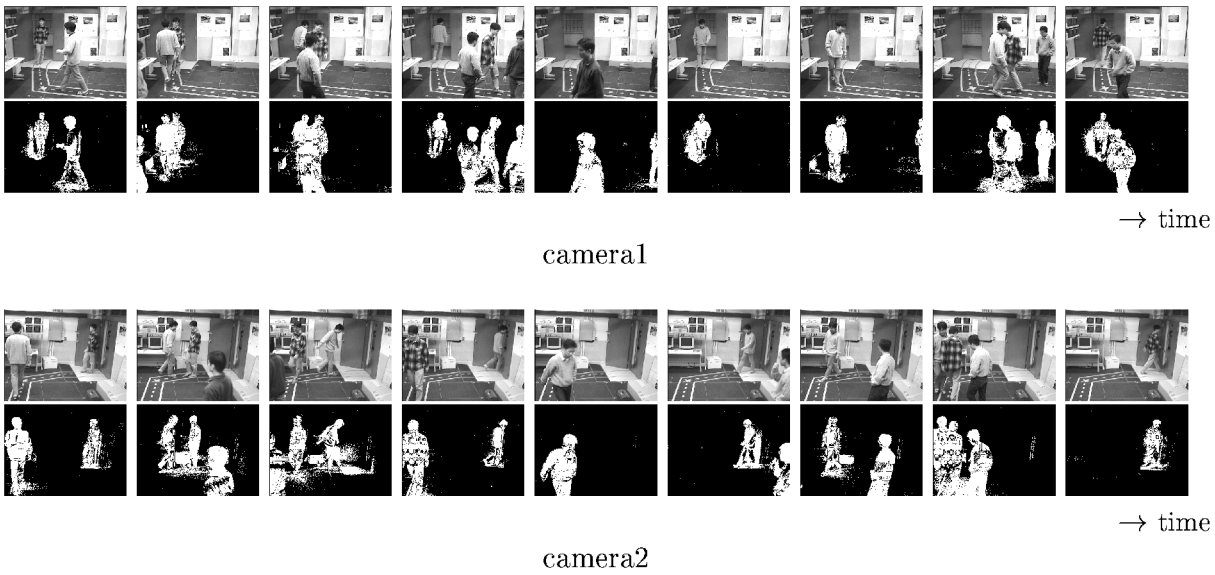

Fig. 19. An example of complicated behaviors: two "enters," two "exits," and outliers: Top and bottom in each behavior are observed gray-level images and anomalous regions obtained by background subtraction with intensity threshold 20.

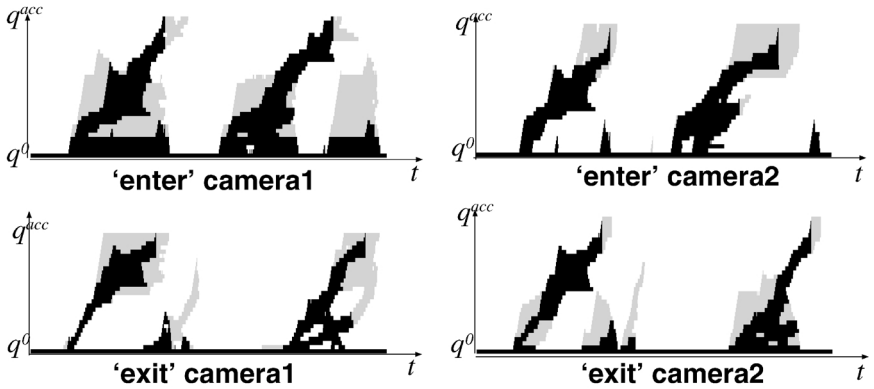

Fig. 20. State transitions of NFAs corresponding to the combinations of {"enter," "exit"} and {"camera 1," "camera 2"} for input shown in Fig. 19: In each graph, the horizontal and vertical axes represent time and state space from initial (bottom) to final (top) states. The gray and black regions corresponds to inhibited and feasible states.

principles: *feasibility* and *optimality*. In the multiobject behavior recognition problem, the number of actual behaviors is unknown and, hence, bottom-up segmentation is required for finding "optimal" recognition result. However, bottom-up segmentation can be affected by occlusions and outliers. This paper shows a way to avoid this dilemma, i.e., if we formalize the problem as a Constraint Satisfaction Problem (CSP) [1] to compute all feasible solutions consistent with given behavior model and input image data, then the problem can easily be solved. Although the purpose of most CSPs is computing only one solution, i.e., satisfiability, our approach is to compute all feasible solutions to realize multiobject behavior recognition. Essentially, all feasible solutions are the intersection between models corresponding to given constraints. By using a simple representation of constraints, i.e., behavior model (focusing region sequence and NFA) and image data (anomalous region), we can directly compute all feasible solutions in a model space spanned by image and state spaces.

The selective attention mechanism actively generates all feasible assumptions (active states in NFA) and verifies them by finding their supporting evidence (events) in the images. From this viewpoint, the NFA can be regarded as a degenerate Assumption-based Truth Maintenance System

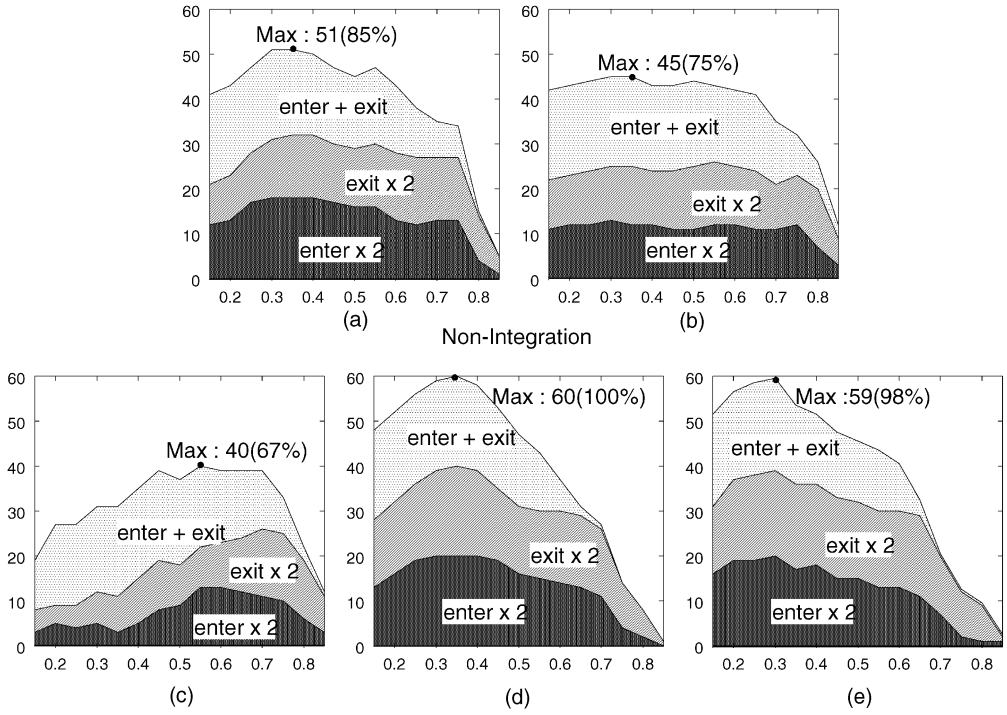

Fig. 21. Total performance evaluation results obtained by changing the event detection threshold θ: (a) Camera1, (b) camera2, (c) image-level integration, (d) event-level integration, and (e) state-level integration. In each, the horizontal and vertical axes represent threshold θ and number of data which are correctly recognized in both senses of "classification" and "number of objects."

(ATMS) [22] that works as a database keeping possible assumptions for multicontext reasoning. An ATMS has a lattice structure to represent the *justification* relationship between the assumptions. However, we use a linear NFA because the "justification" relationship between the assumptions on behavior stages is essentially linear. The ATMS can represent exclusive relationship between the assumptions. Our system can also represent the exclusive relationship between the assumptions using token colors, i.e., activated states having the same colors are exclusive, and others are not.

Our system is also similar with lexical analyzers that segment and classify predefined symbols from 1D input character streams. The difference from the lexical analyzers is that our system accepts 3D spatio-temporal streams including overlapped spatio-temporal patterns. This is realized by introducing the focusing region and event detection. Similar with the lexical analyzer, our architecture can parse those behaviors obeying a Regular Grammar (RG) which is a subset of a Context Free Grammar (CFG). There exist some methods [14] [15] that can parse those behaviors obeying a CFG. With the analogy of formal language analysis, i.e., higher-level parsers can follow the lexical analysis, we can build up a CFG parser on our system. The essential problem of this approach is how to realize the mutual interaction between the higher and lower units, which may make the system robust. In the other approach to parse CFG behaviors, we can replace the NFA by a nondeterministic Push-Down Automaton (PDA). This is because our architecture only requires *instantaneous* and *pure-nondeterministic* state transition model.

The discussion above leads us to a question: What is the essential difference between the behaviors or motions obeying a RG and a CFG? We cannot answer this question based on formal language theory; however, the following assumption might be valid by the analogy of language analysis: *Those object motions constrained by its surrounding environment (behavior) obey RG, and those motions consists of behaviors obey CFG.* In this case, the CFG motions must include RG behaviors constrained by surrounding environment. However, there is interesting research that shows a varieties of motions can be recognized within RG [7]. In this research, a Parametric Hidden Markov Model (PHMM) is used to recognize the variations of known motions. The PHMM can be used for determining global variables while analyzing motions. For example, a PHMM based motion analyzer can recognize a pointing action and estimate the pointing direction simultaneously. This implies that it can recognize a variety of pointing actions with different directions. This result shows the degree of freedom of RG motions are much higher than the behavior addressed in this paper.

Hence, an extension of our method to recognize a wider variety of behaviors should be addressed in future work. The first target in this extension is position free behaviors. One reason why our current system can not be applied to position independent behaviors is that the event detection result is too simple. For recognizing position independent motions, the system should use a structured event detection result which has enough information to specify behavior stages independent of the object position. An example of the structured event codes is used in event-level integration, which can be regarded as a cue for this work.

By modifying the learning method, the selective attention mechanism can recognize discrete variations of behaviors, e.g., training samples of passing through the

door where in half the door already open and the door is closed in the rest. If we apply our learning method described in Section 4 directly to these samples, focusing regions specifying the behavior might be discarded. In this case, the training set should be divided into two subsets, then the behavior identifier can be obtained by the parallel combination of two identifiers trained by these subsets. Parallel combination can be further simplified if these training sets includes common subbehaviors. That is, semantic class definition is not equivalent to the class definition according to behavior pattern similarity. However, there can be a mapping between them.

Since this method uses simple background subtraction, anomalous regions can be affected by illumination changes. Some research on stabilizing the background subtraction using flexible background model and its maintenance [17] [18] can be utilized for solving this problem. One important issue in this extension is that the behavior recognition results can be utilized for background model maintenance. Also, we can apply the background subtraction with an active camera by using the method proposed in [23]. Hence, we can also extend the behavior recognition system with active cameras.

7 Conclusion

In this paper, we addressed the problem of recognizing multiple object behaviors from nonsegmented image sequences. This problem is difficult because optimization based motion classification requires bottom-up segmentation of input image sequences before classifying object behaviors, and segmentation can be affected by noise and outliers. This paper shows a quite different approach from *segmentation followed by classification*, i.e., *assumption generation and verification.* In this approach, we proposed an architecture, the *selective attention mechanism*, based on the principle of *feasibility*: generating *all assumptions* consistent with given behavior models and the input image sequence. The proposed architecture is a systematic combination of an *event detector* and an *event sequence analyzer*; the system generates all feasible assumptions about object behaviors without using bottom-up segmentation.

Based on this mechanism, *colored token propagation* is introduced to distinguish objects. Furthermore, extended recognition methods integrating multiviewpoint image sequences are proposed and examined by extensive experiments of human behaviors. In this experiment, we have confirmed that *event* and *state* level integration methods have good performance even for complex behaviors with outliers.

As we discussed in Section 6, the selective attention mechanism is related to other many concepts in computer science and can be the foundation for novel research. We hope that this paper provides an idea to establish a new frame-work in the field of computer vision.

Acknowledgments

The authors would like to thank Mr. Masayuki Sato at Kyoto University for his assistance and Dr. James J. Little at the University of British Columbia for discussing the ideas in the paper. This work was supported by the Research for the Future Program of the Japan Society for the Promotion of Science (JSPS-RFTF96P00501) and Grant-in-Aid for Scientific Research ((A)(2)08408010).

References

[1] A. Mackworth, "Consistency in Networks of Relations," *Artificial Intelligence,* vol. 8, no. 1, pp. 99-118, 1977.
[2] L. Rabiner and B. Juang, "An Introduction to Hidden Markov Models," *IEEE ASSP Magazine,* pp. 4-16, 1986.
[3] J. Yamato, J. Ohya, and K. Ishii, "Recognizing Human Action in Time-Sequential Images Using Hidden Markov Model," *Proc. Computer Vision and Pattern Recognition,* pp. 664-665, 1992.
[4] T. Starner and A. Pentland, "Real-Time American Sign Language Recognition from Video Using Hidden Markov Models," *Proc. Int'l Symp. Computer Vision,* pp. 265-270, 1995.
[5] C. Bregler and S.M. Omohundro, "Nonlinear Manifold Learning for Visual Speech Recognition," *Proc. Int'l Conf. Computer Vision,* pp. 494-499, 1995.
[6] A. Wilson and A. Bobick, "Learning Visual Behavior for Gesture Analysis," Technical Report 337, MIT Media Laboratory Perceptual Computing Section, 1995
[7] A. Wilson and A. Bobick, "Nonlinear PHMMs for the Interpretation of Parameterized Gesture," *Proc. Computer Vision and Pattern Recognition,* pp. 879-884, 1998.
[8] Z. Ghahramani and M. Jordan, "Factorial Hidden Markov Models," MIT Computational Cognitive Science Report 9,502, Aug. 1995.
[9] L. Saul and M. Jordan, "Boltzmann Chains and Hidden Markov Models," *Advances in Neural Information Processing Systems 7,* G. Tesauro, D.S. Touretzky, and T.K. Leen, eds., MIT Press, 1995.
[10] M. Jordan, Z. Ghahramani, and L. Saul, "Hidden Markov Decision Trees," MIT Computational Cognitive Science Technical Report 9,606, June 1996.
[11] M. Brand, N. Oliver, and A. Pentland, "Coupled Hidden Markov Models for Complex Action Recognition," *Proc. Computer Vision and Pattern Recognition,* pp. 994-999, 1997.
[12] J. Rissanen, "Modeling by Shortest Data Description," *Automatica,* vol. 14, pp. 465-471, 1978.
[13] H. Akaike, "Information Theory and an Extension of the Maximum Likelihood Principle," *Proc. Second Int'l Symp. Information Theory,* B.N. Petrov and F. Csaki, eds., pp. 267-281, 1973.
[14] M. Brand, "Understanding Manipulation in Video," *Proc. AFGR,* pp. 94-99, 1996.
[15] A. Bobick and Y. Ivanov, "Action Recognition Using Probabilistic Parsing," *Proc. Computer Vision and Pattern Recognition,* pp. 196-202, 1998.
[16] J. Hopcroft and J. Ullman, "Introduction To Automata Theory, Languages, and Computation," chapter 2, Addison-Wesley, 1979.
[17] K. Toyama, J. Krumm, B. Brumitt, and B. Meyers, "Wallflower: Principles and Practice of Background Maintenance," *Proc. Int'l Conf. Computer Vision,* pp. 255-261, 1999.
[18] C. Stauffer and W.E.L. Grimson, "Adaptive Background Mixture Models for Real-time Tracking," *Proc. Computer Vision and Pattern Recognition,* pp. 246-252, 1999.
[19] A. Waibel, T. Hanazawa, G. Hinton, K. Shikano, and K. Lang, "Phoneme Recognition Using Time-Delay Neural Networks," *Trans. of ASSP,* vol. 37, no. 3, pp. 328-339, 1989.
[20] M. Yang and N. Ahuja, "Extraction and Classification of Visual Motion Patterns for Hand Gesture Recognition," *Proc. Computer Vision and Pattern Recognition,* pp. 892-897, 1998.
[21] U. Meier, R. Stiefelhagen, J. Yang, and A. Waibel, "Towards Unrestricted Lipreading," *Proc. Int'l Conf. Multimodal Interfaces,* 1999.
[22] J. De Kleer, "An Assumption-Based TMS," *Artificial Intelligence,* vol. 28, pp. 127-162, 1986.
[23] T. Wada and T. Matsuyama, "Appearance Sphere: Background Model for Pan-Tilt-Zoom Camera," *Proc. Int'l Conf. Pattern Recognition,* vol. A, pp. 718-722, 1996.

Toshikazu Wada received the BEng degree in electrical engineering from Okayama University, the MEng degree in computer science from Tokyo Institute of Technology, and the DEng degree in applied electronics from Tokyo Institute of Technology, in 1984, 1987, and 1990, respectively. He is currently an associate professor in the Department of Intelligence Science and Technology, Graduate School of Informatics, Kyoto University. His research interests include pattern recognition, computer vision, image understanding, and artificial intelligence. He received the Marr Prize at the International Conference on Computer Vision in 1995, the Yamashita Memorial Research Award from the Information Processing Society Japan (IPSJ), and the Excellent Paper Award from Institute of Electronics, Information, and Communication Engineers (IEICE), Japan. He is a member of the IEICE, the IPSJ, the Japanese Society for Artificial Intelligence, and the IEEE Computer Society.

Takashi Matsuyama received the BEng, the MEng, and the DEng 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, image media processing, and artificial intelligence. He has published more than 100 papers and books including two research monographs, *A Structural Analysis of Complex Aerial Photographs*, (Plenum, 1980) and *SIGMA: A Knowledge-Based Aerial Image Understanding System,* (Plenum, 1990). 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 on the editorial board of *Computer Vision and Image Understanding.* He is now leading the five year research project on cooperative distributed vision. The project was started in October 1996 under the support of the Research for the Future Program, the Japan Society for the Promotion of Science. The project members include the nation's leading computer vision researchers. He is a fellow of International Association for Pattern Recognition and a member of the Institute of Electronics, Information, and Communication Engineers of Japan, the Information Processing Society of Japan, the Japanese Society for Artificial Intelligence, and the IEEE Computer Society.