Scenario-Based Video Event Recognition by Constraint Flow

Suha Kwak Bohyung Han Joon Hee Han
Department of Computer Science and Engineering, POSTECH, Korea
{mercury3,bhhan,joonhan}@postech.ac.kr

Abstract

We present a novel approach to representing and recognizing composite video events. A composite event is specified by a scenario, which is based on primitive events and their temporal-logical relations, to constrain the arrangements of the primitive events in the composite event. We propose a new scenario description method to represent composite events fluently and efficiently. A composite event is recognized by a constrained optimization algorithm whose constraints are defined by the scenario. The dynamic configuration of the scenario constraints is represented with constraint flow, which is generated from scenario automatically by our scenario parsing algorithm. The constraint flow reduces the search space dramatically, alleviates the effect of preprocessing errors, and guarantees the globally optimal solution for recognition. We validate our method to describe scenario and construct constraint flow for real videos and illustrate the effectiveness of our composite event recognition algorithm for natural video events.

1. Introduction

Automatic video event recognition is one of the most important goals of many video-based intelligent systems including visual surveillance and content-based video retrieval. The recognition of composite video events, which are represented by temporal-logical arrangements of several activity patterns, is useful for extracting high-level semantic information from videos. A composite event typically has its own semantic structure called *scenario*; it also works as a set of constraints used to recognize events in an ambiguous situation. For these reasons, many scenario-based approaches have been proposed to recognize composite events [4, 7, 8, 9, 11, 13].

An important issue of the scenario-based approach is the flexibility of scenarios because it bounds the range of composite events recognized. We propose a novel scenario description method, in which various arrangements of activity patterns can be described fluently by temporal-logical relations; we can describe from simple to complex structures of composite events hierarchically. In general, however, it is difficult to simultaneously satisfy flexibility of scenarios and plausibility of recognition due to semantic gaps[1]. Also, most scenario-based methods suffer from huge a search space because they employ combinatorial optimization whose domain is the combinations of sequential activity patterns. To overcome the limitations of existing techniques, we introduce *constraint flow*, which bounds the search space to a constant size and bridges the semantic gap. The constraint flow also reduces the effect of preprocessing errors, and guarantees the optimal solution at each time step without any assumptions or heuristics to prune the search space.

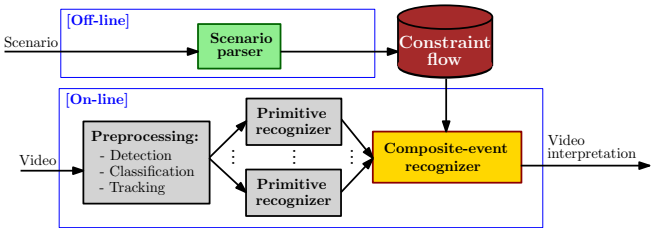


Figure 1. An outline of the overall recognition system. Our system works with the constraint flow pre-compiled in the off-line step and primitives, atomic and meaningful activity patterns, estimated in the on-line step.

The overall system proposed in this paper consists of on-line and off-line processes (Fig. 1). In the off-line process, the parser analyzes the scenario of the target event and generates the corresponding constraint flow automatically. In the on-line event recognition step, the preprocessing units extract information of objects from the input video; this information is then used to estimate the occurrences of the atomic activity patterns, called *primitive*s, which form the target composite event. The composite event recognition is performed based on the estimated primitives from on-line observation as well as the generated constraint flow.

[1]Here, the semantic gap characterizes the difference between scenarios described by human and the associated machine understanding. More flexible scenario causes broader semantic gap in general.

The rest of this paper is organized as follows. Section 2 reviews previous work related to event recognition, and Section 3 introduces our scenario description method. In Section 4, we define the constraint flow and present how it is generated automatically. Composite event recognition using the constraint flow is described in Section 5, and experimental results are presented in Section 6.

2. Related work

There has been a lot of research done on automatically recognizing what happens in videos. A large number of methods for abnormality detection have been proposed recently [3, 5]. They are useful in real world applications because they are flexible and easy to implement even when annotated data is not sufficient. However, semantic interpretations of videos are not straightforward in the above approaches. So, we focus on knowledge-based semantic reasoning for composite events [6].

Stochastic context-free grammar (SCFG) has been widely used to recognize composite events in videos. The SCFGs effectively model and recognize composite events with uncertainties, but cannot handle simultaneous occurrences of primitives because CFGs generate only 1-dimensional languages [4, 8]. Dynamic Bayesian networks (DBNs) address more flexible scenarios than SCFGs, where structures of DBNs determine the corresponding scenarios. The Hidden Markov model (HMM) still have the same limitation as SCFGs, but [2] attempted to overcome such limitation by introducing several variations of the basic HMM. Other DBNs have been studied for the same purpose; propagation network [10, 11] and activity DBN [7] were proposed to recognize more complicated composite events with sampling-based inference algorithms. However, DBNs are still insufficient to describe various events because their scenarios are bounded by their conditional dependency structures. Also, the inference algorithms for DBNs involve heuristics such as beam search [10, 11] or greedy pruning of candidates [7] to explore the huge search space. Temporal-logical predicates [1] have been used to enhance the flexibility of scenarios [9, 13], but the huge search space problem still remains; their reasoning algorithms exhaustively search the primitive recognition results in a temporally backward direction [13] or assume that a primitive appears only once in an execution of the target event—an extreme case of the time window search [9].

From the above observations, we mainly focus on 1) the flexibility of event scenarios and 2) the exact inference without assumptions or heuristics. So, we design a novel scenario description method based on temporal-logical predicates. Also, the constraint flow is proposed to make probabilistic inference subject to given scenario constraints in a dynamic programming framework to obtain the globally optimal solution without approximation.

3. Video event description

A video event can be represented with a sequence of patterns, which are extracted from objects in videos. We regard an event as a meaningful sequence of patterns whose characteristics are defined in advance.

This section describes a novel event description method to define the structures of the sequential patterns. We identify primitives and define several temporal-logical relations among the primitives or their groups. Also, we present a systematic description method based on the relations.

3.1. Event categorization

We divide events into two categories: *primitive*s and *composite event*s. Primitives are morphemes to construct composite events and they cannot be divided into smaller events. Also, their occurrences can be estimated at each time step by analyzing the motion and spatial information of objects. Therefore, the occurrences of primitives are discrete and instantaneous; we define a *time interval* of a primitive as a group of discrete but consecutive occurrences of the primitive. A composite event is organized by the time intervals of primitives, which are arranged by the temporal-logical relations among them. A specification of a composite event, i.e., a set of primitives and corresponding relations among them, is called *scenario*. A composite event is defined by a unique scenario; a composite event may involve multiple scenarios in hierarchical manners because a composite event can be a part of other composite events.

3.2. Scenario description for composite events

A scenario describes how the corresponding composite event should happen. It consists of primitives forming the target event and their relations describing how they participate in the configuration of the target event. Let E be a composite event and $\mathbf{e}$ be a set of one or multiple primitives of E. The scenario $\mathcal{S}$ of E is defined as follows:

$$\mathcal{S}(E) = \left\{ r_k(\mathbf{e}_i, \mathbf{e}_j) \middle| \mathbf{e}_i \subset E, \mathbf{e}_j \subset E, r_k \in \mathcal{R} \right\}. \quad (1)$$

A relational predicate r, which is included in the set of relations $\mathcal{R}$, is a constraint for the arrangements of time intervals of the corresponding primitives. We define four temporal relations and two logical relations for $\mathcal{R}$.

The temporal relations constrain the order of the time intervals. Let $start(\mathbf{e})$ and $end(\mathbf{e})$ be the start time and the end time of the occurrence of $\mathbf{e}$, respectively. The temporal relations are denoted and defined as follows.

$$\begin{aligned}
\mathbf{e}_i < \mathbf{e}_j \quad &\Leftrightarrow \quad end(\mathbf{e}_i) \text{ is earlier than } start(\mathbf{e}_j) \\
\mathbf{e}_i \wedge \mathbf{e}_j \quad &\Leftrightarrow \quad start(\mathbf{e}_i) \text{ is earlier than } start(\mathbf{e}_j) \text{ and} \\
&\qquad\quad end(\mathbf{e}_j) \text{ is earlier than } end(\mathbf{e}_i) \\
\mathbf{e}_i \sim \mathbf{e}_j \quad &\Leftrightarrow \quad start(\mathbf{e}_i) \text{ is earlier than } start(\mathbf{e}_j)
\end{aligned}$$

Comp. event	Scenario
TakeItem[A]	Move[A, desk] < Move[A, out] ∧ With[A, item]
BringItem[A]	(With[A,item]∼Move[A,desk])<Move[A,out]
TakeMoney[A]	Without[A,money]<With[A,money]∧Move[A,out]
BringMoney[A]	(With[A,money]∧Move[A,desk])<Without[A,money]
ScanItem[A]	With[A,item]∧(Move[A,scanner]<Move[A,desk])
Payment†	(BringMoney[ctm]∼TakeMoney[csh])< . . . #|(BringMoney[csh]∼TakeMoney[ctm])
Transaction†	(BringItem[ctm]$^+$ ∼ScanItem[csh]$^+$)< . . . Payment<TakeItem[ctm]$^+$

Table 1. Scenarios of composite events that happen during a transaction between a customer (ctm) and a cashier (csh). For example, TakeItem[A] means 'an actor A moves to the desk, and then moves away with an item'. The two events with † mark are described in a hierarchical manner.

$$\mathbf{e}_i{}^+ \quad \Leftrightarrow \quad end(\mathbf{e}_i^m) \text{ is earlier than } start(\mathbf{e}_i^{m+1}) \quad \forall m = 1, 2, 3, \ldots$$

The last unary relation is applied to the pairs of identical primitive sets to describe an unknown number of recurrences of a specified primitive set; the superscript m represents the sequential index of the occurrences.

Two logical relations are denoted and defined as follows.

$\mathbf{e}_i \& \mathbf{e}_j \quad \Leftrightarrow$ both of $\mathbf{e}_i$ and $\mathbf{e}_j$ occur; they are independent temporally and logically.

$\mathbf{e}_i \mid \mathbf{e}_j \quad \Leftrightarrow$ only one of $\mathbf{e}_i$ and $\mathbf{e}_j$ occurs.

We also add the concept of a dummy denoted by '#'. It is used to represent "no occurrence" and is paired with the '|' relation.

The relations in parentheses take precedence over relations outside of parentheses, and this rule is identically applied to nested parentheses. Except for parentheses, logical relations take precedence over temporal relations and there is no precedence among temporal relations.

We manually designed the scenario for the composite event Transaction by our description method (Table 1). In this example, three types of primitives are defined: Move[A, B], With[A, B], and Without[A, B] which mean "A moves to B", "A is very close to B", and "A is far from B", respectively.

4. Constraint flow

Scenario-based event recognition can be considered as a constrained optimization problem; the domain of the objective function is a set of all possible interpretations of the video observation, and the scenario is a set of constraints of the optimization problem. After the objective function is defined, the problem is solved by finding the optimal interpretation from the feasible set, which is the set of interpretations that satisfies the scenario.

An interpretation is a combination of occurrence histories of the scenario primitives; an interpretation at the t^{th} time step is a binary matrix X_t whose row refers to each primitive and whose column represents the time axis up to t. For example, $X_t[i, j]$ shows whether the primitive e_i occurred at the j^{th} time step or not ($j \leq t$). This definition is straightforward to represent all possible situations. For recognition, it is not sensitive to missing primitives or false alarms because an interpretation considers all primitives at once. In addition, it is useful to recognize the repetition of a composite event because an interpretation can contain many occurrences of a target event.

When using this approach, we cannot manage all interpretations because their space increases exponentially over time. Also, it is difficult to check the feasibility of interpretations when the scenario is complex and involves many primitives. For these reasons, we propose the *constraint flow*. The constraint flow generates feasible interpretations sequentially, as it simultaneously picks only the interpretations that are necessary to track the globally optimal solution.

4.1. Definition of constraint flow

A column of an interpretation matrix, $X_t[:, j]$ is a combination of binary conditions of the primitives at a time step j. An interpretation of the observations until the t^{th} time step is a trajectory of the combinations, and a scenario becomes a set of discrete gating functions, which determine next possible combinations for the given combinations. Then a set of gating functions can be represented with an unweighted directed graph whose vertices are the combinations of the primitives and whose links restrict transitions among them. Note that, the constraint flow involves a scenario and is defined by the unweighted directed graph, where the vertices are combinations of *quinary conditions* of the primitives (Fig. 2(a)). If a primitive is currently occurring, its condition is 'active'. Otherwise, it is 'inactive'. Because these binary conditions are not sufficient to represent temporal-logical structures of scenarios, we subdivide 'inactive' into four conditions as in Table 2.

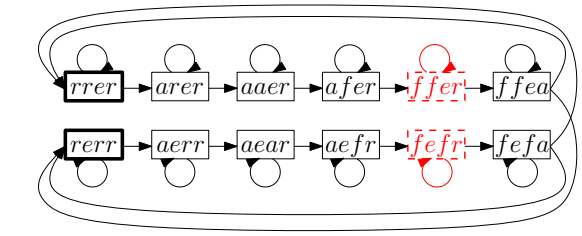
The constraint flow is a dynamic configuration of the given scenario; its property is equivalent to the scenario constraints. It means that all trajectories by tracing the flow always satisfy the scenario. Given the constraint flow of the scenario, therefore, we can generate feasible interpretations by projecting the flow tracing results to the space of the binary conditions (Fig. 2(b)).

4.2. Construction of constraint flow

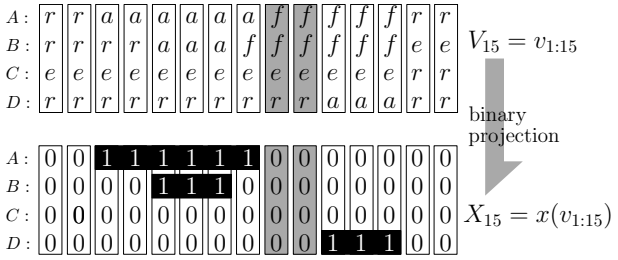
We introduce a scenario parsing algorithm to automatically build the constraint flow given a scenario. The parsing algorithm works similar to the Breadth-First Search (BFS) algorithm, but the graph structure is unknown in our case.

Condition	Meaning (The primitive . . .)	Possible shifts
active(a)	occurs currently.	active, finished
ready(r)	does not occur yet. (inactive)	ready, active
finished(f)	ends its activation. (inactive)	finished
waiting(w)	waits next activation. (inactive)	waiting, active
excluded(e)	does not participate. (inactive)	excluded

Table 2. The quinary conditions to specify flow vertices.



(a) The constraint flow structure



(b) Generating a feasible interpretation by tracing the flow

Figure 2. The constraint flow and its utilization for the scenario "(A∧B|C)<D". (a) There are two initial vertices (the bold boxes) and two idle vertices (the dashed boxes). (b) A flow trace (V_{15}) becomes a feasible interpretation (X_{15}) by quinary to binary projection; the time intervals, which consist of consecutive activations, are painted black, and the idle intervals, generated by the idle vertices, are painted gray.

Therefore, the parser generates vertices while traversing the graph. Also, the preprocesses for vertex generation—primitive grouping and ancestor search—are performed before the flow construction. The entire parsing procedure is described in Algorithm 1, and we discuss several important steps below.

Grouping Primitives by '+' (Line 2). A unary temporal-relation '+' is applied to a primitive or a set of primitives enclosed by parentheses. The parser checks the existing '+' relations in the scenario, and the set of primitives affected by each '+' relation forms a group. Consequently, the grouping result $\mathcal{U}$ is a set of primitive groups associated with each '+' relation.

Search for Start and End Time Ancestors (Line 3). The set of start time ancestors of a primitive e, denoted by

Algorithm 1 Constraint flow generation by scenario parsing

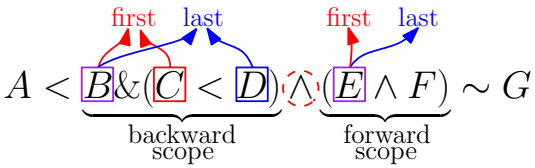
Require: a scenario $\mathcal{S}$ with n primitives
1: Empty the following data structures.
   $\mathcal{V}$: set of vertices, $\mathcal{L}$: set of links,
   Q: queue structure, $\{v^{\text{last}}\}$: set of the last vertices
2: $\mathcal{U} \leftarrow$ GroupPrimitivesBy+($\mathcal{S}$)
3: $\{\mathcal{A}_{\text{start}}(e_i), \mathcal{A}_{\text{end}}(e_i)\}_{i=1}^{n} \leftarrow$ SearchForAncestors($\mathcal{S}$)
4: $\{v^{\text{init}}\} \leftarrow$ GenerateInitialVertices($\mathcal{S}$)
5: Enqueue($\{v^{\text{init}}\}, Q$).
6: **while** Q is not empty **do**
7: $v^k \leftarrow$Dequeue(Q).
8: $\{v^{\text{suc}(k)}\} \leftarrow$ GenerateSuccessiveVertices $(v^k, \{\mathcal{A}_{\text{start}}(e_i), \mathcal{A}_{\text{end}}(e_i)\}_{i=1}^{n}, \mathcal{U})$
9: **for** each $v^{\text{suc}(k)}$ **do**
10: **if** $v^{\text{suc}(k)}$ consists of 'finished' and 'excluded' only **then**
11: Add v^k to $\{v^{\text{last}}\}$ and go to Line 9.
12: **end if**
13: Add a link $\langle v^k \rightarrow v^{\text{suc}(k)} \rangle$ to $\mathcal{L}$.
14: **if** $v^{\text{suc}(k)}$ is not in $\mathcal{V}$ **then**
15: Add $v^{\text{suc}(k)}$ to $\mathcal{V}$ and Enqueue($v^{\text{suc}(k)}, Q$).
16: **end if**
17: **end for**
18: **end while**
19: Add a set of links $\{\langle v^{\text{last}} \rightarrow v^{\text{init}} \rangle\}$ to $\mathcal{L}$.
20: **return** The constraint flow $\mathcal{G} = \{\mathcal{V}, \mathcal{L}\}$

$\mathcal{A}_{\text{start}}(e)$, is the set of primitives, which constrain $start(e)$ by the temporal relations '<', '∼', and '∧'. In a similar way, $\mathcal{A}_{\text{end}}(e)$, the set of end time ancestors of e, is the set of primitives, which restrict $end(e)$ by '∧' relation.

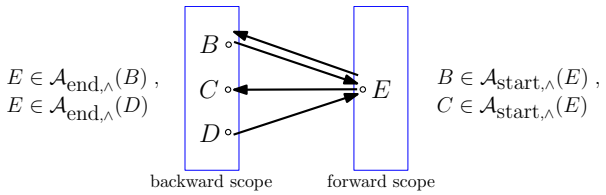
The ancestor search is performed on each binary temporal-relation in the scenario. A binary temporal-relation has backward and forward scopes determined by adjacent parentheses and logical relations (Fig. 3(a)). From each of the two scopes, the first and last primitives in the temporal order are chosen. Then, the ancestors are determined by bipartite matching between the selected primitives (Fig. 3(b)). Note that, if a primitive serves a dummy as its ancestor, the primitive should also serve the ancestors of the dummy.

Generating Initial Vertices (Line 4). An initial flow-vertex consists of 'ready' and 'excluded' conditions only. If there exist '|' relations in the scenario, multiple initial vertices are created (Fig. 2(a)). The parser checks the existing '|'s and then generates initial vertices for all possible combinations of 'excluded' primitives. Also, all dummies are set to 'excluded'.

Generating Successive Vertices (Line 8). To traverse unknown graphs, the parser should generate successive ver-



(a) Scope analysis



(b) Search for ancestors by bipartite matching

Figure 3. An example for ancestor search. Search at the circled '$\wedge$' is performed with the sets of primitives $\{B, C, D\}$ and $\{E\}$, which are temporally first or last on its backward and forward scopes. The set of ancestors for one of the primitives includes the other elements, which are bipartitely matched with the primitive.

tices of the vertex, where it is currently positioned. First, the parser checks how each primitive can shift its condition from the current condition. The primitives can keep their conditions, or each primitive can shift its condition if its ancestors do not restrict the shift at the current vertex. The possible shifts of each condition are summarized in Table 2. Once the possible condition-shifts of each primitive are identified, the successors are generated by the combinations of the shifts. Note that the set of the successive vertices always include the current vertex.

Additionally, each group in $\mathcal{U}$ can be activated again when the entire group primitives are deactivated. Therefore, the entire primitives grouped by the same '$+$' relation can shift their conditions to 'waiting' (or 'excluded' when there exist '$|$' relations among them) at once. Different groups have different labels for the 'waiting' condition to handle nested '$+$' scopes.

Constraint Flow Construction From the initial flow vertices, the parser constructs a flow structure by iterative generation (Line 8) and search of the structure. The search strategy is similar to the BFS algorithm, and is implemented by a queue (Line 7 and 15). During the construction, the parser finds the last vertices whose entire conditions are 'excluded', 'finished' or ready to be 'finished' (Line 11). After the search, the parser adds links from the last vertices to the initial vertices for recurrences of the target event (Line 19).

5. Composite event recognition

Event recognition is challenging due to severe preprocessing noises. In addition, most scenario-based approaches suffer from huge search spaces because the set of (even feasible) interpretations grows exponentially over time.

The constraint flow addresses both of the above problems. First, in the constraint flow, the sequential probabilistic inference with the scenario constraints effectively suppresses noisy observations. Second, the search space to maintain the globally optimal interpretation is bounded by a fixed size through the constraint flow; the optimal solution is obtained by dynamic programming efficiently. Therefore, one can infer the optimal solution without any assumptions or heuristics.

5.1. Constrained optimization for recognition

We regard composite-event recognition as a constrained optimization problem. The search space is restricted to the set of feasible interpretations. The constraint flow is traced to generate feasible interpretations. Let $v_{1:t}$ be a tracing result up to the t^{th} time step, and x be the binary projection for the quinary conditions of the trace; $x(v_{1:t})$ is a feasible interpretation at the t^{th} time step (*e.g.*, X_{15} in Fig. 2(b)). The optimal solution is selected from the feasible set. The objective function, the measure of optimality, is designed by considering the two properties of an interpretation: *the observation agreement* and *the length of idle intervals*.

The observation agreement is formulated by the posterior probability of an interpretation, which can be factored in a sequential manner as

$$p\big(x(v_{1:t})|O_{1:t}\big) \propto p\big(O_t|x(v_t)\big) \cdot p\big(x(v_t)|x(v_{1:t-1})\big) \cdot p\big(x(v_{1:t-1})|O_{1:t-1}\big), \quad (2)$$

where O_t represents the observation at time step t, and is assumed to be dependent only on $x(v_t)$. Also, we assume that a primitive occurs independently of the others. Then, the conditional observation probability in Eq. (2) becomes the product of the likelihoods with respect to occurrence of each primitive:

$$p\big(O_t|x(v_t)\big) = \prod_{i=1}^{n} p\big(O_t^i|x(v_t)\big) = \prod_{i=1}^{n} p\big(O_t^i|x(v_t^i)\big), \quad (3)$$

where n is the number of primitives of the scenario. The state transition probability, $P(x(v_t)|x(v_{1:t-1}))$, is assumed to be uniform because it is difficult to be learned in advance; its value is fixed to 2^{-n} because there are the 2^n number of binary vectors in the n-dimensional space. Note that the constraint flow is not a graphical model but a guide to sieve the feasible interpretations from the huge set of all interpretations. Therefore, the transition probability is not affected by the constraint flow structure.

The length of idle intervals measures cohesion among the time intervals that organize an occurrence of the target event. If no primitive is activated while the target composite event is in progress at a time step, then we say the time

step is *idle*. Idle time steps are induced by *idle vertices* of the constraint flow. An idle vertex only contains 'inactive' primitives, but there is at least one 'finished' primitive among them. An idle interval is a set of idle time steps arranged consecutively (Fig. 2(b)). Because idle intervals are generally short in real situations, the entire length of idle intervals of an interpretation is adopted as a penalty term, which is given by

$$\mathcal{B}(v_{1:t}) = \prod_{j=1}^{t} \exp\big(-\beta I_b(v_j)\big),$$
$$\text{where } I_b(v_j) = \begin{cases} 1, & \text{if } v_j \text{ is idle,} \\ 0, & \text{otherwise.} \end{cases} \quad (4)$$

The objective function, f is defined by the product of the two measurements in Eq. (2) and (4) as

$$f(v_{1:t}) = p\big(x(v_{1:t})|O_{1:t}\big) \cdot \mathcal{B}(v_{1:t})$$
$$\propto \exp\big(-\beta I_b(v_t)\big) \cdot p\big(O_t|x(v_t)\big) \cdot f(v_{1:t-1}). \quad (5)$$

Because the penalty of Eq. (4) needs the constraint flow information, the objective function takes a flow trace (*e.g.*, V_{15} in Fig. 2(b)) instead of an interpretation as its input. Then, the optimal interpretation becomes the binary projection of the trace (*e.g.*, X_{15} in Fig. 2(b)), which maximizes the objective function.

5.2. Recognition with constraint flow

The objective function in Eq. (5) is optimized at each flow vertex v^k (vertex in Fig. 2(a)) and each time step sequentially by dynamic programming:

$$\max_{v_{1:t-1}} f([v_{1:t-1},\ v_t = v^k]) \propto \exp\big(-\beta I_b(v^k)\big) \cdot p\big(O_t|x(v^k)\big)$$
$$\cdot \max_{v_{t-1}} \Big\{ \max_{v_{1:t-2}} f([v_{1:t-2},\ v_{t-1}]) \Big\}, \quad (6)$$

where $v_{1:t-1}$ is a trace matrix and v_t is a column vector. At the initial stage of recognition, the procedure starts from the initial flow vertices; $f(v_0)$ is 1 if v_0 is an initial vertex and 0 otherwise. Let $V_t(v^k)$ be the trace, which maximizes the objective function in a flow vertex v^k at time step t.

$$V_t(v^k) = \Big[\arg\max_{v_{1:t-1}} f([v_{1:t-1},\ v_t = v^k]),\ \ v_t = v^k \Big]. \quad (7)$$

We only need to maintain $V_t(v^k)$ with $f(V_t(v^k))$ for each flow vertex v^k to sequentially optimize the objective function because only the optimal trace in each vertex at the current time step is used to compute the optimal trace at the next time step.[2] The optimal interpretation at the t^{th} time step is given by

$$\widehat{X}_t = x\Big(\arg\max_{v_{1:t}} f(v_{1:t}) \Big) = x\Big(\arg\max_{V_t(v^k)} f(V_t(v^k)) \Big) \quad (8)$$

[2]This means that one can always find the optimal solution by holding only one trace per each flow vertex at each time step.

In summary, our recognition algorithm has the following advantages. First, the scenario constraints can be easily combined with the sequential and run-time inference framework. Second, the number of the traces needed to find the optimal interpretation is bounded by the number of the flow vertices. Therefore, the size of the search space is not a parameter anymore if the number of the flow vertices is tolerable in the system. Third, an iterative recognition of a target composite event is available by linking the last vertices to the initial vertices on the constraint flow.

6. Experiments

We validated our approach with two videos—a surveillance video and a tennis video, which involve 8 transactions between a customer and a casher with complex structure (Table 1) and natural tennis plays between a server and a receiver (Table 3), respectively. Our event recognition algorithm with the constraint flow annotates complex composite events accurately in spite of noisy primitive detections.

6.1. Preprocessing: primitive detection

In both sequences, moving objects in the scene are detected by a pixel-wise background subtraction [12], and each object is identified by the trained appearance model—hand, money and item in the *surveillance* sequence, and players and a ball in the *tennis* sequence. The primitives are recognized based on the prior information of spatial relations among the objects (*e.g.*, a hand and money moving together), and scene contexts (*e.g.*, the locations of the desk and the barcode scanner). In the *tennis* video, we need a more sophisticated action recognizer to extract primitives. The motion context descriptors [14] obtained from the objects are classified by the large margin nearest neighbor (LMNN) classifier [15] to recognize four kinds of actions: 'serve' for the primitive Serve[A], 'forehand stroke' and 'backhand stroke' for the primitive Swing[A], and 'other actions' for others. Note that the primitive recognition corresponds to the likelihood function, $p\big(O_t^i|x(v_t^i)\big)$ in Eq. (3).

Precision and recall rates of the primitive detection in the both sequences are illustrated in Fig. 4. The primitive recognition is significantly noisy because of uncertain video observations and inevitable errors caused by the flexibility of our scenario description method; our method allows multiple occurrences of a primitive in a scenario and a primitive may be interpreted to different meanings. With the noisy signals, however, our system works successfully by using the constraint flow (Fig. 5).

The common scenario parser and composite-event recognizer are employed in all experiments. The entire system requires only two parameters; the scenario for the target event E, $\mathcal{S}(E)$, and the penalty weight for idle time steps, β in Eq. (4).

Comp. event	Scenario
Service[A]	Serve[A] $\wedge$ Part[A, ball]
Stroke[A]	(Meet[A, ball] $<$ Part[A, ball]) $\wedge$ Swing[A]
TennisPlay	Service[ser] $<$ (# | Service[ser]) $<$ . . .
	(# | (Stroke[rec] $<$ Stroke[ser])$^+$) $<$ . . .
	(# | Stroke[rec])

Table 3. The scenario of a tennis play and supporting composite events. Meet[A, B], Part[A, B], Serve[A], and Swing[A] are the primitives. For example, Stroke[A] means 'an actor A swings between "the ball approaches" and "the ball leaves".' The target event, TennisPlay is described in a hierarchical manner.

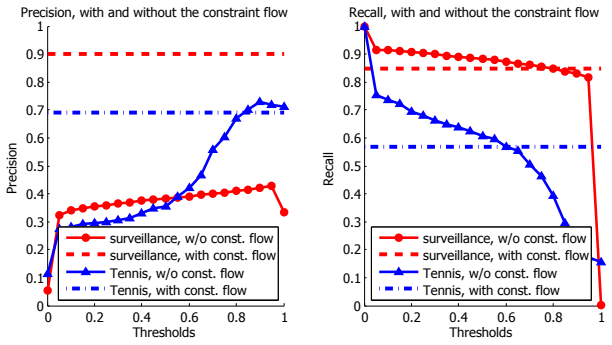
6.2. Surveillance sequence

In the surveillance sequence, our system recognizes multiple Transactions accurately; it is not straightforward to recognize multiple occurrences of the target event in other algorithms. The intermediate composite events for the Transaction are also recognized successfully (see the overlap of black and purple bounding boxes in Fig. 5(a)). Note that the system does not assume any time windows as in [9, 11] or perform greedy pruning as in [7] to reduce search space. It only select potentially optimal traces by dynamic programming—476 traces at each time step, which is equal to the number of flow vertices.

The system also provides semantic interpretations of the video contents, which shows the flexibility of our framework. For example, a cashier may or may not give change during transactions (6^{th} and 7^{th} transactions). Our system handles the variations by using just one scenario. Also, the interpretation can be used to check the legality. There are two illegal events in the surveillance video—a missing Payment in the 5^{th} transaction and a missing TakeItem[customer] in the 8^{th} transaction. The system is able to identify the illegal events by analyzing the optimal interpretation. For example, in the interpretation of the illegal transaction with a missing Payment, some primitives for Payment have too short time intervals; they are considered as *hallucination*s, which mean missing signals.

6.3. Tennis sequence

The primitive recognition is considerably noisy in the tennis sequence (Fig. 4), and the number of strokes to comprise TennisPlay event is unknown. Despite such challenges, our event recognitions and annotations are mostly successful in this sequence as illustrated in Fig. 5(b). Note that our framework has the capability to recognize complex composite events accurately, even with significant noises in primitive recognition, by applying the constraint flow generated by our scenario description method.



(a) Precision (b) Recall

Figure 4. Comparison between the frame-wise performances with and without the constraint flow. The performance of raw primitive detections was measured by thresholding their likelihoods. The performance of our system is as follows: 0.90 for precision, 0.84 for recall, 0.0053 for false positive rate (*surveillance*), and 0.69 for precision, 0.57 for recall, 0.033 for false positive rate (*tennis*).

7. Conclusion

We proposed a novel framework to describe and recognize complex video events. Our event description method facilitates flexible representations through the various relational predicates. The constraint flow reduces the search space dramatically without any assumptions or heuristics, and the optimal solution is found by dynamic programming in an on-line manner. We demonstrated the effectiveness of event recognition by our algorithm in complex videos.

Acknowledgement: This research was supported in part by the IT R&D program of MKE/IITA (2008-F-031-01, Development of Computational Photography Technologies for Image and Video Contents), and in part by Basic Science Research Program through the National Research Foundation of Korea funded by the Ministry of Education, Science and Technology (2010-0003496).

References

[1] J. F. Allen and G. Ferguson. Actions and events in interval temporal logic. *Journal of Logic and Computation*, 4(5):531–579, 1997. 2

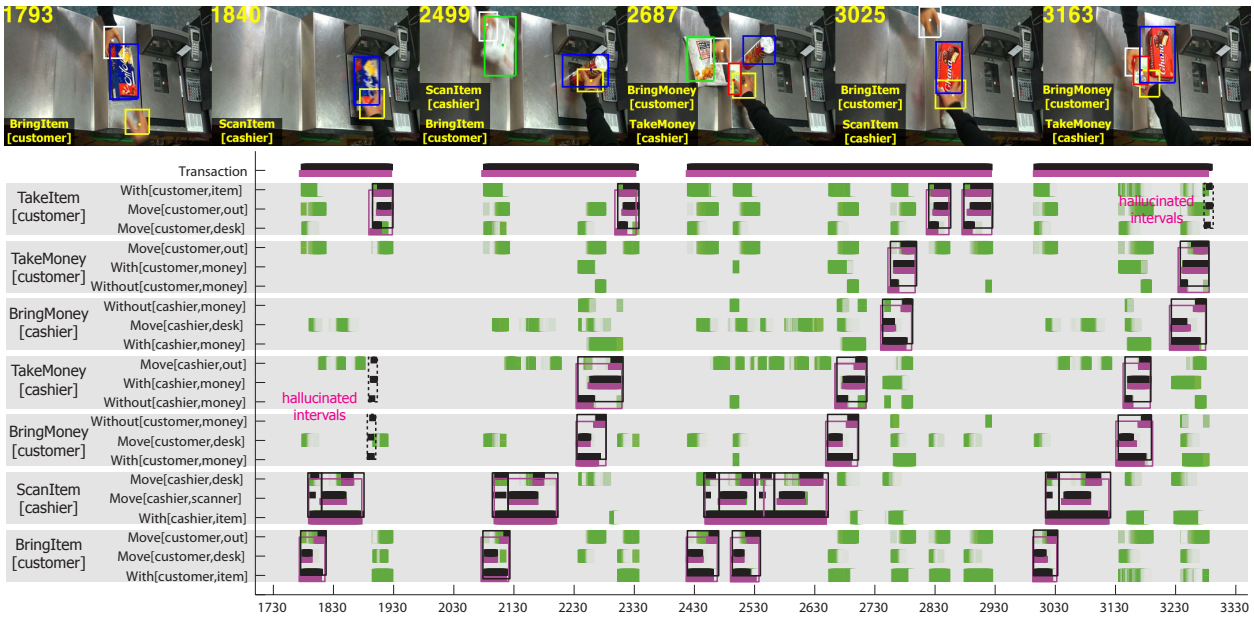
[2] S. Gong and T. Xiang. Recognition of group activities using dynamic probabilistic networks. In *ICCV*, volume 2, pages 742–749, 2003. 2

[3] T. Hospedales, S. Gong, and T. Xiang. A markov clustering topic model for mining behaviour in video. In *ICCV*, pages 1165 –1172, 2009. 2

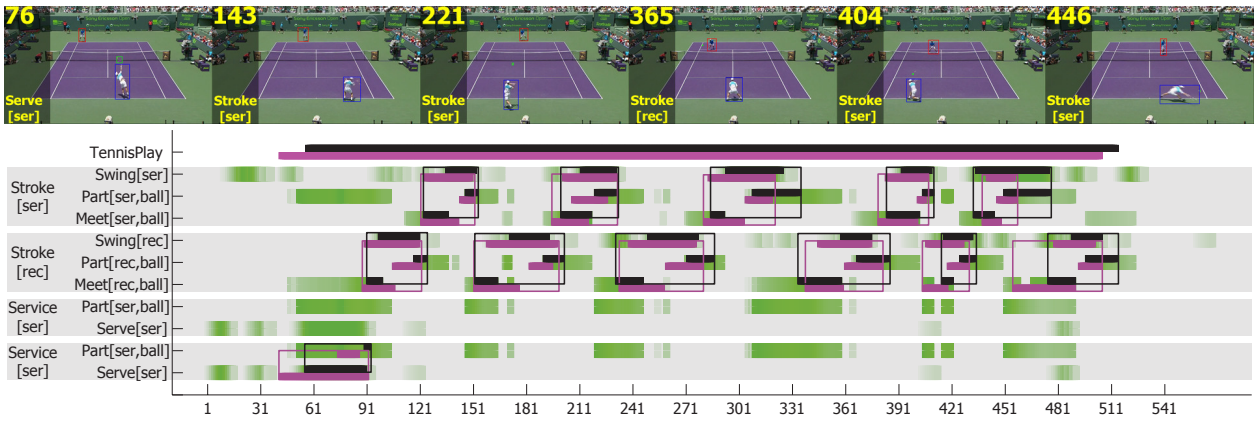
[4] Y. A. Ivanov and A. F. Bobick. Recognition of visual activities and interactions by stochastic parsing. *PAMI*, 22(8):852–872, 2000. 1, 2

[5] D. Kuettel, M. Breitenstein, L. Van Gool, and V. Ferrari. What's going on? discovering spatio-temporal dependencies in dynamic scenes. In *CVPR*, pages 1951 –1958, 2010. 2

[6] G. Lavee, E. Rivlin, and M. Rudzsky. Understanding video events: A survey of methods for automatic interpretation of semantic occurrences in video. *IEEE Transactions on SMC, Part C: Applications and Review*, 39(5):489 –504, sept. 2009. 2



(a) The four consecutive recognition results(5^{th}, 6^{th}, 7^{th} and 8^{th} transactions) of the surveillance video.



(b) The recognition result of the *YouTube* tennis video.

Figure 5. Recognition results: the surveillance sequence (*upper*) and the tennis sequence (*lower*). The horizontal axis represents the time indices and the vertical axis enumerate the target event (*top*) and the associated primitives. The raw responses of the primitive recognizers are painted green; the darker means higher response. The black bars are the intervals estimated by using the constraint flow. The ground-truth is represented by the puple bars, below the black bars. The time intervals of a same composite event are grouped by a bounding box to show the configuration of the estimated target event and the arrangements of the intervals in the composite-event level.

[7] B. Laxton, J. Lim, and D. Kriegman. Leveraging temporal, contextual and ordering constraints for recognizing complex activities in video. In *CVPR*, pages 1–8, 2007. 1, 2, 7

[8] D. Moore and I. Essa. Recognizing multitasked activities from video using stochastic context-free grammar. In *AAAI*, pages 770–776, 2002. 1, 2

[9] M. S. Ryoo and J. K. Aggarwal. Semantic representation and recognition of continued and recursive human activities. *IJCV*, 82(1):1–24, 2009. 1, 2, 7

[10] Y. Shi, A. Bobick, and I. Essa. Learning temporal sequence model from partially labeled data. In *CVPR*, volume 2, pages 1631–1638, 2006. 2

[11] Y. Shi, Y. Huang, D. Minnen, A. Bobick, and I. Essa. Propagation networks for recognition of partially ordered sequential action. In *CVPR*, volume 2, pages 1631–1638, 2004. 1, 2, 7

[12] C. Stauffer and W. Grimson. Adaptive background mixture models for real-time tracking. In *CVPR*, volume 2, pages –252, 1999. 6

[13] V. thinh Vu, F. Bremond, and M. Thonnat. Automatic video interpretation: A novel algorithm for temporal scenario recognition. In *IJCAI*, pages 9–15, 2003. 1, 2

[14] D. Tran, A. Sorokin, and D. Forsyth. Human activity recognition with metric learning. In *ECCV*, pages 548–561, 2008. 6

[15] K. Q. Weinberger, J. Blitzer, and L. K. Saul. Distance metric learning for large margin nearest neighbor classification. In *NIPS*, pages 1473–1480, 2006. 6