Multi-Agent Event Detection: Localization and Role Assignment*

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 joint estimation technique of event localization and role assignment when the target video event is described by a scenario. Specifically, to detect multi-agent events from video, our algorithm identifies agents involved in an event and assigns roles to the participating agents. Instead of iterating through all possible agent-role combinations, we formulate the joint optimization problem as two efficient subproblems—quadratic programming for role assignment followed by linear programming for event localization. Additionally, we reduce the computational complexity significantly by applying role-specific event detectors to each agent independently. We test the performance of our algorithm in natural videos, which contain multiple target events and nonparticipating agents.

1. Introduction

Event detection—identification of a predefined event in videos—typically suffers from an enormous combinatorial complexity. It is aggravated by additional challenges such as the agents' unknown roles and the extra agents who do not participate in the event. We are interested in such challenging problem, *event localization* and *role assignment*, which refers to identifying the target event and recognizing its participants in the presence of non-participants, while estimating the roles of the participants automatically at the same time. Although computer vision community has broad interest in the problems related to event detection, the joint estimation of event localization and role assignment has rarely been studied before. This is particularly due to the computational complexity induced by various factors, for example, multiple agent participation in an event, temporal-logical variations of an event or potential activities irrelevant to the target event.

In this paper, we introduce a novel algorithm for video event understanding, where the event localization problem is solved jointly with the role assignment by formulating the problem as an optimization framework. Our algorithm incorporates [10] to describe and detect a target event based on a scenario. A naïve extension of [10] for event localization and role assignment however requires the consideration of all possible agent-role combinations, which may be intractable computationally. In this work, we attempt to reduce the search space dramatically by removing infeasible agent-role combinations and construct a mathematical formulation for solving the problem efficiently.

For this purpose, we first generate agent groups, each of which is composed of agents potentially having mutual interactions defined in a scenario. The confidence and time interval of each role are estimated for each agent by role-specific event detection. The event localization and role assignment are solved by assigning roles to agents optimally per group and selecting groups whose role assignments are feasible. The contribution of this paper is three-fold. 1) The event localization and role assignment problems are jointly managed in a single optimization framework. 2) Our framework handles some challenging situations in event detection in a principled way, *e.g.*, presence of nonparticipants and multiple occurrences of the same event. 3) Our algorithm is efficient due to the significantly reduced search space.

The rest of the paper is organized as follows. Sec. 2 reviews previous work closely related to multi-agent event detection. After we discuss the event detection framework based on the constraint flow in Sec. 3, our multi-agent event localization and role assignment algorithm is presented in Sec. 4. We illustrate experimental results in Sec. 5.

2. Related work

The simplest tasks related to event detection include atomic action detection (*e.g.*, [21]) and abnormality detection (*e.g.*, [9]) in videos. An atomic action is a short-term event generated by a single agent, and abnormality detection is a binary decision with respect to video contents; both approaches are insufficient to deduce semantic interpretations from videos. In this paper, we consider an event as a composition of atomic actions, which we call *primitives*, under a temporal-logical structure. The structure of

*This work was supported by MEST Basic Science Research Program through the NRF of Korea (2012-0003697), the IT R&D program of MKE/KEIT (10040246), and Samsung Electronics Co., Ltd.

the event is described by a scenario.

Video events have been detected by stochastic grammars [8, 14] or graphical models including hidden Markov models (HMMs) [2], variants of the basic HMM [5] and dynamic Bayesian networks of given structures [12, 19]. However, they have limitations in describing events; grammar-based approaches cannot cover concurrent streams of primitives, and graphical models represent scenarios only by temporal orders between primitives. Recently, scenarios are often described by logic such as temporal interval algebra [1] and event logic [20] to represent various aspects of events more fluently. Logic-based scenarios have been adopted by various event detection algorithms including hierarchical constraint satisfaction [17], multithread parsing [24], randomized stochastic local search [4], and probabilistic inference on constraint flow [10] and Markov logic network [15, 22]. However, most of these approaches have focused on events with a single agent only [2, 12, 14, 19], or events that can be detected without role identification [4, 5, 15]. Otherwise, roles should be initialized by prior information or human intervention [8, 10, 17, 24].

A few recent approaches discuss role assignment in the detection of complex multi-agent events. In [11], an MRF captures relationships between roles and performs per-video event categorization; the extensions to handle non-participants and localize events in spatio-temporal domain are not straightforward. An MCMC technique is introduced to identify agents' roles and detect an event at the same time in [18]. However, it is a heuristic-based method presenting results in videos with a single event occurrence only. Our method is formulated in a more principled way and can handle multiple occurrences of an event naturally.

3. Scenario-based event detection

Our framework incorporates [10] to describe and detect target events based on scenarios. In this section, we briefly review the scenario description method and event detection procedure using constraint flow presented in [10].

3.1. Scenario description

A scenario describes a target event based on primitives and their temporal-logical relationships. The relationships constrain arrangements of time intervals of the primitives. Let ρ_i and ρ_j be primitives organizing a target event. Four temporal relationships define temporal orders between starting and ending times of two primitives:

$$\begin{aligned}
\rho_i < \rho_j &\Leftrightarrow \mathit{end}(\rho_i) \text{ is earlier than } \mathit{start}(\rho_j).\\
\rho_i \sim \rho_j &\Leftrightarrow \mathit{start}(\rho_i) \text{ is earlier than } \mathit{start}(\rho_j).\\
\rho_i \wedge \rho_j &\Leftrightarrow \mathit{start}(\rho_i) \text{ is earlier than } \mathit{start}(\rho_j) \text{ and}\\
&\qquad \mathit{end}(\rho_j) \text{ is earlier than } \mathit{end}(\rho_i).\\
{\rho_i}^{+} &\Leftrightarrow \mathit{end}(\rho_i^k) \text{ is earlier than } \mathit{start}(\rho_i^{k+1}),\quad \forall k \in \mathbb{N}.
\end{aligned}$$

Condition	**Meaning** (The primitive ...)	**Change**†
active (a)	occurs currently.	finished
ready (r)	does not occur yet.	active
finished (f)	ends its activation.	(none)
waiting (w)	waits for next activation.	active
excluded ($\times$)	does not participate.	(none)

† indicates possible changes from the original condition.

Table 1. The quinary conditions to specify flow vertices

The last unary relationship is to describe an unknown number of recurrences of a specified primitive; k is an index for the number of recurrences. Two logical relationships are denoted and defined as follows:

$$\begin{aligned}
\rho_i \& \rho_j &\Leftrightarrow \rho_i \text{ and } \rho_j \text{ occur;}\\
&\qquad \text{they are independent temporally and logically.}\\
\rho_i \mid \rho_j &\Leftrightarrow \text{only one of } \rho_i \text{ and } \rho_j \text{ occurs.}
\end{aligned}$$

In principle, logical relationships take precedence over temporal relationships, but we can use parentheses to override the precedence of the relationships.

The following is the scenario of Delivery event that we design by using the above description method:

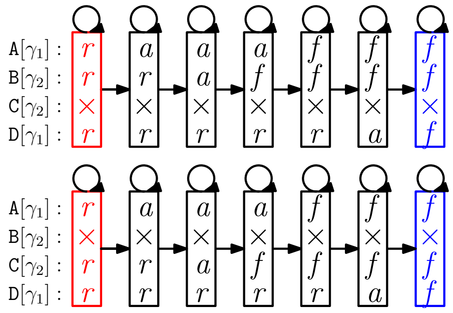
$$\begin{aligned}
&\mathsf{Delivery}[\gamma_{\text{del}}, \gamma_{\text{rec}}] \Rightarrow\\
&\quad \mathsf{GetOff}[\gamma_{\text{del}}] < \mathsf{ComeClose}[\gamma_{\text{rec}}] \ \&\ \mathsf{HoldObj}[\gamma_{\text{del}}] < \cdots\\
&\quad (\mathsf{GetAway}[\gamma_{\text{rec}}] \wedge \mathsf{HoldObj}[\gamma_{\text{rec}}]) \ \&\ \mathsf{GetInto}[\gamma_{\text{del}}],
\end{aligned}$$

where GetOff, ComeClose, HoldObj, GetAway, and GetInto are primitives. γ_{del} and γ_{rec} indicate two roles in Delivery, deliverer and receiver, respectively. A role associated with each primitive is specified in the attached bracket.

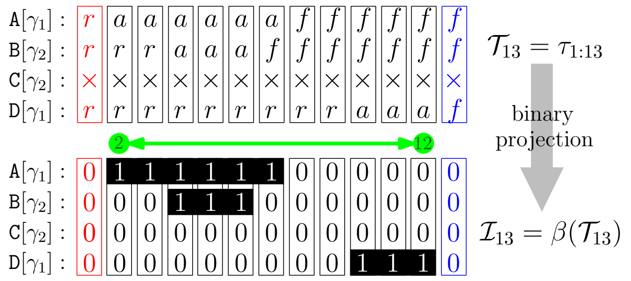
3.2. Constraint flow and event detection

Constraint flow is a state transition machine characterizing the organization of a target event. The state of a target event at a time step is represented by a combination of conditions of all primitives. The condition of a primitive is *active* if the primitive occurs at the time step, and it is *inactive* otherwise; the inactive condition is subdivided into four hypotheses (Table 1) so that various temporal-logical relationships among primitives can be described. The constraint flow is a directed graph, whose vertices are combinations of quinary conditions of the primitives (Fig. 1(a)). When a scenario is given, the corresponding constraint flow is generated automatically by the scenario parsing technique described in Algorithm 1 of [10].

The objective of event detection is to find the best feasible interpretation of a given video. An interpretation is defined as a configuration of time intervals of the primitives. It is feasible if it satisfies all constraints in the scenario. Because a scenario and its constraint flow are equivalent representations to describe an event, we can generate feasible



(a) Constraint flow



(b) Generating a feasible interpretation

Figure 1. An example of constraint flow. The scenario is "$(\mathsf{A}[\gamma_1] \wedge \mathsf{B}[\gamma_2] \mid \mathsf{C}[\gamma_2]) < \mathsf{D}[\gamma_1]$". (a) There are two initial vertices (red boxes) and two final vertices (blue boxes) in the constraint flow. (b) A flow traverse ($\mathcal{T}_{13}$) becomes a feasible interpretation ($\mathcal{I}_{13}$) by projecting quinary conditions to binary conditions; the projection is denoted by β. The time intervals of the primitives are marked black. The time interval of the target event is obtained by searching for the first and last activations in the interpretation; in this example, the time interval of the target event is [2, 12].

interpretations by traversing the constraint flow (Fig. 1(b)). Let $\mathcal{T}_t$ be a flow traverse up to time t and $\mathcal{I}_t = \beta(\mathcal{T}_t)$ be the feasible interpretation obtained by the binary projection of $\mathcal{T}_t$. Event detection is formulated by the maximum a posteriori (MAP) estimate of the optimal traverse $\widehat{\mathcal{T}}_t$ as

$$\widehat{\mathcal{T}}_t = \underset{\mathcal{T}_t}{\operatorname{argmax}}\, P(\mathcal{T}_t|O_{1:t}), \tag{1}$$

where observation O indicates primitive detection, and the optimal interpretation is the binary projection of $\widehat{\mathcal{T}}_t$, *i.e.*, $\widehat{\mathcal{I}}_t = \beta(\widehat{\mathcal{T}}_t)$. The exact solution of Eq. (1) is obtained by dynamic programming on the constraint flow. For the details for solving Eq. (1), refer to [10].

4. Multi-agent event localization

The event detector described in Sec. 3 assumes that roles of all agents are given manually. We call a group of agents involved in a target event with assigned roles a true *lineup*, and there are typically many lineup candidates (*i.e.*, agent-role combinations) in a video. Let m be the number of agents existing in a video, and n be the number of roles defined in a scenario. We assume that an agent takes part in at most one target event in a video and that the agents and the roles are bijective in a target event. Then there exist $\frac{m!}{(m-n)!}$ lineup candidates for the target event. It is impractical to apply event detection algorithm to all the candidates.

We introduce a technique that efficiently localizes multi-agent events by identifying true lineups in an optimization framework. Our method exploits role confidences and time intervals of agents, which are obtained by role-specific event detection per agent (Sec. 4.2), as criteria for lineup evaluation. Despite a huge number of potential lineups, we maintain efficiency by separating the event localization procedure into two steps: role estimation per group and group selection (Sec. 4.4). These are solved by quadratic and linear programming, respectively.

4.1. Agent grouping

Suppose that there are n roles in a target event. We first form groups with n agents, which are the candidates to be applied to our event detection algorithm. An agent can be associated with multiple groups at this stage. Note that there exist $l \cdot n!$ lineup candidates, where l denotes the number of groups[1], because $n!$ role configurations are available per group; it may be infeasible to identify few true lineups from such a large number of candidates.

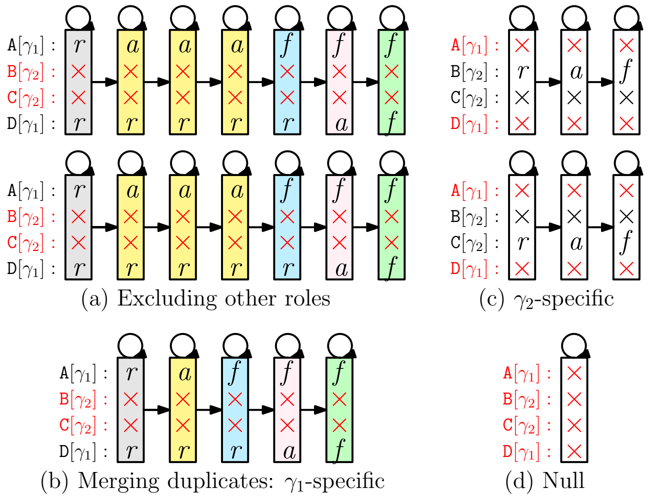
The groups are represented by an $m \times l$ binary matrix $\mathbf{A}$, where $[\mathbf{A}]_{i,k} = 1$ if the i-th agent belongs to the k-th group, and 0 otherwise. For simplicity of notation, we define a function $g_k : \{1, \ldots, n\} \to \{1, \ldots, m\}$, which indicates the association between the members in the k-th group and the agents. That is, $g_k(i') = i$ if the i-th agent is the i'-th member in the k-th group.

4.2. Agent-wise role analysis

The role assignment is essential for multi-agent event detection, but typically available after the completion of event detection. To estimate the role of an agent, we evaluate how appropriate the properties of the agent for each role are, and also estimate potential time interval of each role. In other words, we perform role-specific event detection for each agent; it is done by role-specific constraint flows, each of which focuses only on a single role. The role-specific constraint flow is constructed from the original constraint flow by simply excluding primitives not associated with the target role and merging duplicate vertices, as shown in Fig. 2. We additionally construct the null constraint flow, which is generated by excluding all primitives (Fig. 2(d)) to handle *outsiders*, agents that do not participate in any target events.

For each agent in a video, we apply the role-specific event detectors to all primitive detections. For the i-th

[1]We should consider $\binom{m}{n}$ groups in principle, but can reduce the search space significantly in practice by integrating spatio-temporal or temporal-logical proximity constraint between agents as described in Sec. 5.



(a) Excluding other roles
(c) γ_2-specific
(b) Merging duplicates: γ_1-specific
(d) Null

Figure 2. An example of generating role-specific constraint flows. The scenario and its original constraint flow are the same with those of Fig. 1. (a–b) Procedure for generating γ_1-specific constraint flow: We first exclude primitives irrelevant to the role we currently focus on (*red*); some vertices become identical after the exclusion. The role-specific constraint flow is obtained by merging the duplicate vertices. (c) γ_2-specific constraint flow is obtained in the same manner. (d) We additionally construct the null constraint flow to handle outsiders.

agent, we obtain the MAP traverses $\widehat{\mathcal{T}}^{i,1}_{\text{end}}, \dots, \widehat{\mathcal{T}}^{i,n}_{\text{end}}$ for the n roles in the scenario, and $\widehat{\mathcal{T}}^{i,0}_{\text{end}}$ for the outsider role. Then we evaluate how appropriate the i-th agent is for the j-th role based on the corresponding maximum log-posterior probability; the normalized confidence is given by

$$c^{i,j} = \log P(\widehat{\mathcal{T}}^{i,j}_{\text{end}} | O^i_{1:\text{end}}) - \log \sum_{k=0}^{n} P(\widehat{\mathcal{T}}^{i,k}_{\text{end}} | O^i_{1:\text{end}}). \quad (2)$$

Also, the time interval where the i-th agent plays the j-th role is obtained by searching for the first and last activations in $\beta(\widehat{\mathcal{T}}^{i,j}_{\text{end}})$ as shown in Fig. 1(b); the starting and ending time of the interval are denoted by $s^{i,j}$ and $e^{i,j}$, respectively. Note that we do not consider the time intervals for the outsiders because every element of $\widehat{\mathcal{T}}^{i,0}_{\text{end}}$ is *excluded*. Finally, the agent-wise role information is summarized by $\mathbf{c}^i = [c^{i,1}, \dots, c^{i,n}]^\top$, $\mathbf{s}^i = [s^{i,1}, \dots, s^{i,n}]^\top$, and $\mathbf{e}^i = [e^{i,1}, \dots, e^{i,n}]^\top$ for all $i = 1, \dots, m$. Also, the confidences for the outsider role are concatenated to $\mathbf{c}_0 = [c^{1,0}, \dots, c^{m,0}]^\top$.

In our framework, a lineup candidate is evaluated in terms of the role confidences and time intervals of the corresponding agent-role combination, which are precomputed in this role-specific event detection and can be reused to evaluate different lineup candidates. The reuse of role-specific evidences makes the two-step optimization possible. Also, lineup candidates with different lengths of occurrence (*i.e.*, different number of observations) can be evaluated fairly at the role level because role confidences are not affected by the length of occurrence due to the agent-wise normalization in Eq. (2).

The agent-wise role analysis is significantly faster than the naïve extension of the original event detection because role-specific event detection evaluates each role of each agent independently. However, interactions between roles are ignored in this step, which may result in erroneous lineup evaluations. Therefore, we adopt a post-processing to reject misidentified lineups, which is described in Sec. 4.5.

4.3. Evaluating role assignment per group

For each group, we evaluate the feasibility of the role assignment by considering how appropriate the agents in a group are for the assigned roles and how well their role-specific time intervals satisfy the scenario constraints. The evaluation measure for the k-th group is formulated by a quadratic function, which is given by

$$f_k(\mathbf{x}_k) = \frac{1}{2}\mathbf{x}_k^\top \mathbf{L}_k \mathbf{x}_k + \mathbf{c}_k^\top \mathbf{x}_k. \quad (3)$$

In this function, $\mathbf{x}_k = [\mathbf{x}_k^{1\top}, \dots, \mathbf{x}_k^{n\top}]^\top$ is a role assignment variable, where $\mathbf{x}_k^i = [x_k^{i,1}, \dots, x_k^{i,n}]^\top$ indicates the role assignment of the i-th member in a binary form, *i.e.*, $x_k^{i,j} = 1$ if the j-th role is assigned to the i-th member of the k-th group. $\mathbf{c}_k = [\mathbf{c}^{g_k(1)\top}, \dots, \mathbf{c}^{g_k(n)\top}]^\top$ is a coefficient vector of role confidences corresponding to the members for n roles. $\mathbf{L}_k$ is an $n^2 \times n^2$ coefficient matrix to measure the fidelity of the time intervals of the assigned roles to the scenario constraints:

$$\mathbf{L}_k = \begin{bmatrix} -\infty_{n\times n} & \mathbf{L}_k^{1,2} & \cdots & \mathbf{L}_k^{1,n} \\ \mathbf{L}_k^{2,1} & -\infty_{n\times n} & \cdots & \mathbf{L}_k^{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{L}_k^{n,1} & \mathbf{L}_k^{n,2} & \cdots & -\infty_{n\times n} \end{bmatrix}, \quad (4)$$

where $\infty_{n\times n}$ is an $n \times n$ matrix whose elements are all infinities. $\mathbf{L}_k^{i_1,i_2}$ is an $n \times n$ penalty matrix defined by the temporal relationship between the pairs of members in the k-th group. Specifically, the element of the matrix, $[\mathbf{L}_k^{i_1,i_2}]_{j_1,j_2}$, is negatively proportional to the length of partial time-intervals violating the scenario constraints, between the j_1-th role of the i_1-th member and the j_2-th role of the i_2-th member, which is given by

$$\begin{aligned} \mathbf{L}_k^{i_1,i_2} = -\alpha \cdot \Big\{ & \max\left[\mathbf{0}_{n\times n},\ \mathbf{S}_{i_1} \circ \mathbf{R}_{\text{ss}} + \mathbf{S}_{i_2}^\top \circ \mathbf{R}_{\text{ss}}^\top\right] + \\ & \max\left[\mathbf{0}_{n\times n},\ \mathbf{E}_{i_1} \circ \mathbf{R}_{\text{ee}} + \mathbf{E}_{i_2}^\top \circ \mathbf{R}_{\text{ee}}^\top\right] + \\ & \max\left[\mathbf{0}_{n\times n},\ \mathbf{S}_{i_1} \circ \mathbf{R}_{\text{se}} - \mathbf{E}_{i_2}^\top \circ \mathbf{R}_{\text{se}}\right] \cdot 2 \Big\}, \end{aligned} \quad (5)$$

where α is a constant, $\mathbf{0}_{n\times n}$ is an $n \times n$ matrix whose elements are all 0's, and $\circ$ is the Hadamard (elementwise) product. $\mathbf{S}_i$ and $\mathbf{E}_i$ are $n \times n$ matrices whose columns are composed of $\mathbf{s}^{g_k(i)}$ and $\mathbf{e}^{g_k(i)}$, respectively, *i.e.*, $\mathbf{S}_i = [\mathbf{s}^{g_k(i)}, \ldots, \mathbf{s}^{g_k(i)}]$ and $\mathbf{E}_i = [\mathbf{e}^{g_k(i)}, \ldots, \mathbf{e}^{g_k(i)}]$. $\mathbf{R}_{\text{ss}}$, $\mathbf{R}_{\text{ee}}$, and $\mathbf{R}_{\text{se}}$ are $n \times n$ matrices for temporal relationships between roles, which are given respectively by

$$[\mathbf{R}_{\text{ss}}]_{j_1,j_2} = \begin{cases} \infty, & \text{if } j_1 = j_2, \\ 1, & \text{else if } \mathit{Start}(\gamma_{j_1}) < \mathit{Start}(\gamma_{j_2}), \\ -1, & \text{else if } \mathit{Start}(\gamma_{j_1}) > \mathit{Start}(\gamma_{j_2}), \\ 0, & \text{otherwise}, \end{cases} \tag{6}$$

$$[\mathbf{R}_{\text{ee}}]_{j_1,j_2} = \begin{cases} \infty, & \text{if } j_1 = j_2, \\ 1, & \text{else if } \mathit{End}(\gamma_{j_1}) < \mathit{End}(\gamma_{j_2}), \\ -1, & \text{else if } \mathit{End}(\gamma_{j_1}) > \mathit{End}(\gamma_{j_2}), \\ 0, & \text{otherwise}, \end{cases} \tag{7}$$

$$[\mathbf{R}_{\text{se}}]_{j_1,j_2} = \begin{cases} \infty, & \text{if } j_1 = j_2, \\ 1, & \text{else if } \mathit{Start}(\gamma_{j_1}) < \mathit{End}(\gamma_{j_2}), \\ -1, & \text{else if } \mathit{Start}(\gamma_{j_1}) > \mathit{End}(\gamma_{j_2}), \\ 0, & \text{otherwise}, \end{cases} \tag{8}$$

where $\mathit{Start}(\gamma_j)$ is the starting time of the j-th role, and $\mathit{End}(\gamma_j)$ is the ending time of the j-th role. When the j-th role involves multiple primitives in an event, $\mathit{Start}(\gamma_j)$ means the starting time of its earliest primitive and $\mathit{End}(\gamma_j)$ is the ending time of its latest primitive. Note that the values in $\mathbf{R}_{\text{ss}}$, $\mathbf{R}_{\text{ee}}$, and $\mathbf{R}_{\text{se}}$ depend on the scenario and are determined during scenario parsing.

4.4. Event localization through lineup estimation

We identify target event lineups by maximizing the sum of lineup evaluation scores and outsider confidences. This problem involves a joint estimation of group selection and role configuration within the selected groups, which is formulated by an optimization problem with the set-packing constraint [6]. Formally, the objective function of this constrained optimization problem is defined by

$$\begin{aligned} \max_{\mathbf{y},\mathbf{x}_1,\ldots,\mathbf{x}_l} \quad & \mathbf{f}(\mathbf{x}_1, \ldots, \mathbf{x}_l)^\top \mathbf{y} + \mathbf{c}_0^\top (\mathbf{1}_m - \mathbf{A}\mathbf{y}) \\ \text{s.t.} \quad & \mathbf{A}\mathbf{y} \le \mathbf{1}_m, \end{aligned} \tag{9}$$

where $\mathbf{y}$ is an l-dimensional binary vector reflecting the group selection, and $\mathbf{x}_k$ is a role assignment variable for the k-th group. $\mathbf{f}(\mathbf{x}_1, \ldots, \mathbf{x}_l)^\top$ is a vector function, which returns a vector of the evaluation scores of the estimated lineups, *i.e.*, $[f_1(\mathbf{x}_1), \ldots, f_l(\mathbf{x}_l)]^\top$. $\mathbf{1}_m$ is an m-dimensional vector whose values are all 1's, so $(\mathbf{1}_m - \mathbf{A}\mathbf{y})$ is a binary vector indicating the estimated outsiders.

Solving the joint optimization Eq. (9) is time-consuming as its search space is prohibitively large, $O(2^{l \cdot n!})$. Instead, we employ a two-step optimization technique for lineup estimation, which is equivalent to the original problem.[2] In the first step, we find the best role configuration per group; the best role assignment to the members of the k-th group can be obtained by maximizing Eq. (3) as

$$\widehat{\mathbf{x}}_k = \operatorname*{argmax}_{\mathbf{x}_k} \; f_k(\mathbf{x}_k) \qquad \text{s.t. } \mathbf{U}\mathbf{x}_k = \mathbf{1}, \;\; \mathbf{V}\mathbf{x}_k = \mathbf{1}, \tag{10}$$

where $\mathbf{U}$ is an $n \times n^2$ binary matrix constraining each member to play only one role and $\mathbf{V}$ is an $n \times n^2$ binary matrix constraining each role to be assigned to only one member, which are given respectively by

$$[\mathbf{U}]_{i,j} = \begin{cases} 1, & \text{if } n(i-1)+1 \le j \le ni, \\ 0, & \text{otherwise}, \end{cases} \tag{11}$$

$$[\mathbf{V}]_{i,j} = \begin{cases} 1, & \text{if } i \equiv j \bmod n, \\ 0, & \text{otherwise}. \end{cases} \tag{12}$$

The second step is for group selection given the best role configurations of all groups, which is given by

$$\max_{\mathbf{y}} \; \widehat{\mathbf{f}}^\top \mathbf{y} + \mathbf{c}_0^\top (\mathbf{1}_m - \mathbf{A}\mathbf{y}) \qquad \text{s.t. } \mathbf{A}\mathbf{y} \le \mathbf{1}_m, \tag{13}$$

where $\widehat{\mathbf{f}}$ is the collection of the best lineup evaluation scores that are obtained from the objective function in Eq. (10), *i.e.*, $\widehat{\mathbf{f}} = [f_1(\widehat{\mathbf{x}}_1), \ldots, f_l(\widehat{\mathbf{x}}_l)]^\top$.

In summary, for joint event localization and role assignment, we first generate groups, identify the best lineup candidate per group by quadratic programming, and then finally select the best lineup candidates by linear programming. The optimization procedures are modeled and solved by YALMIP [13].

4.5. Multi-agent event detection

In our framework, role confidences of each agent are estimated by marginalizing event structure over roles as described in Sec. 4.2. Such marginalization is useful to avoid a combinatorial explosion of the complexity, but it ignores interactions among roles. This may affect the accuracy of event detection, particularly increasing false alarms. Therefore, we finally verify the identified lineups obtained from Eq. (9) by applying the event detector in Sec. 3 to them. Each identified lineup is validated by counting how many *hallucinations* exist in its video interpretation, where a hallucination is a time interval whose length is too short; the lineup will be rejected if the number of hallucinations in its interpretation is too big. Note that hallucinations found in the verification process come from unnatural role configuration, which is not observed in the optimization step because role confidences of an agent are marginalized.

5. Experiments

We demonstrate our framework on *surveillance* sequence from [10], two sequences from VIRAT dataset [16], and five *fieldgoal* sequences downloaded from *YouTube*. All

[2]The proof about the equivalence is given in the supplementary file.

tested sequences contain many agents and unknown number of target events whose starting and ending times are arbitrary. Our framework accurately and efficiently localizes and detects target events in the sequences.

5.1. Surveillance sequence

In *surveillance* sequence, there are 16 agents with no outsiders and 8 Transaction events, which consist of two roles, customer and cashier.[3] We first grouped 2 agents whose occurrences temporally overlap and obtained 8 groups; the number of lineup candidates is 16 because 2 role assignments are available per group. Our framework correctly identifies 8 lineups out of the 16 candidates and detects 8 true target events.

To test robustness of our algorithm, we performed a simulation by gradually increasing the number of virtual agents—increasing observation noise for lineup estimation. Virtual agents were generated by sampling role confidences and time interval lengths from a Dirichlet distribution and a Gaussian distribution, respectively. Both distributions were estimated from the 16 real agents. Also, starting times of virtual agents were uniformly distributed over the entire sequence. Performance in the presence of the virtual agents is illustrated in Fig. 3. According to the simulation, the recall rates are consistently high regardless of the number of virtual agents (Fig. 3(a)) while only a small number of probable lineups are maintained among an exponentially increasing number of candidates (Fig. 3(c)); it is because infeasible lineup candidates are penalized by the marginalized scenario constraints (Eq. (6–8)) and ignored by the set-packing constraint in Eq. (9).

5.2. VIRAT sequences

We validate our framework with two sequences, *VIRAT1* (*S_000101*) and *VIRAT2* (*S_000102*), from VIRAT dataset; the target event in the sequences is Delivery, whose scenario is described in Sec. 3.1. Because fragments of tracks, which we call tracklets, are given for the sequences, we track moving objects by associating the tracklets [7]. Each agent group in the sequences is made up of 2 agents that are spatially adjacent at a sufficiently long time interval. The primitives of Delivery are detected with respect to each agent by measuring spatial relationships between the agent and other objects (*e.g.*, cars and boxes).

Then, our framework is applied to automatically localize and detect the target event; there are three Delivery events in *VIRAT1* and one Delivery is in *VIRAT2*. Also, our algorithm is compared with a naïve approach, which attempts to detect the event from all lineup candidates. The performance is summarized in Table 2. Note that there are a large

	Agents	Groups	Events (Ours)	Events (Naïve)
VIRAT1	21	23	3 (**1.0**/1.0)	4 (**0.75**/1.0)
VIRAT2	47	97	1 (1.0/1.0)	1 (1.0/1.0)

Table 2. Results in *VIRAT* sequences. Precision and recall of event detections are given in parentheses (precision/recall). Performance by the naïve approach is also given for comparison.

		Role anal.	Lineup est.	Event det.	Overall
VIRAT1	Ours	50.6	6.6	14.5	**71.8**
	Naïve	-	-	507.7	507.8
VIRAT2	Ours	150.2	9.0	136.7	**295.9**
	Naïve	-	-	6939.5	6939.5

Table 3. Comparison of execution times in seconds in *VIRAT1* and *VIRAT2* sequences; the execution times for agent grouping are omitted since they are too short (≤ 0.1 sec).

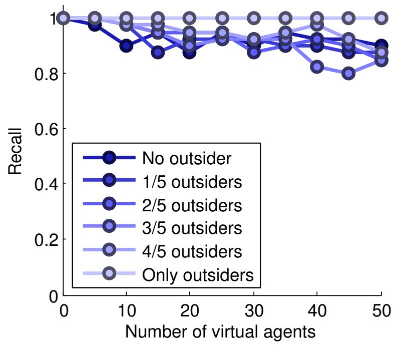
number of lineup candidates and some outsiders act like receivers near cars in both sequences; it is not straightforward to detect and localize true target events with correct role assignments. Our framework identifies all true lineups, and misidentified lineups are successfully removed by the validation step (see Fig. 5); the final event detection results are illustrated in Fig. 4. Note that our framework achieves perfect accuracy whereas the naïve approach has a false alarm in *VIRAT1* sequence. This is because our algorithm can prevent conflicting lineups (*e.g.*, lineups sharing at least one agent) by using a single objective function while the naïve approach evaluates each lineup candidate independently.

Our framework is also significantly faster than the naïve method; according to our experiments, it is 7 and 23 times faster than the naïve approach in *VIRAT1* and *VIRAT2*, respectively. Note that *VIRAT2* has more agent groups; the relative speed is highly dependent on the number of groups in the sequence. Table 3 presents the execution times of the individual components in both algorithms. The computational complexity of the naïve approach is $O(l \cdot n! \cdot |\mathcal{L}|)$, where $|\mathcal{L}|$ is the number of edges in the constraint flow, while our complexity is $O(m \cdot n \cdot |\mathcal{L}'| + C + \hat{l} \cdot |\mathcal{L}|)$, where $|\mathcal{L}'|$ is the average number of edges in the role-specific constraint flows, C indicates the cost of lineup estimation, and $\hat{l}$ means the number of the estimated lineups. It is obvious that $O(l \cdot n! \cdot |\mathcal{L}|) \gg O(m \cdot n \cdot |\mathcal{L}'| + \hat{l} \cdot |\mathcal{L}|)$.

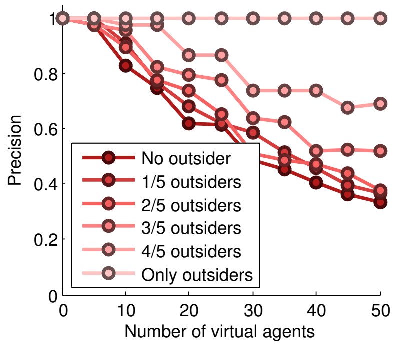
5.3. Fieldgoal sequences

Each of the five *fieldgoal* sequences contains 20 or more agents and one Fieldgoal event, which requires three roles: holder, snapper, and kicker. Due to severe occlusions and rapid appearance changes, agents in the sequences are tracked by annotating key frames [23]. The pose of an agent is described by a pyramid of histograms of oriented gradients [3] and classified by a linear support vector ma-

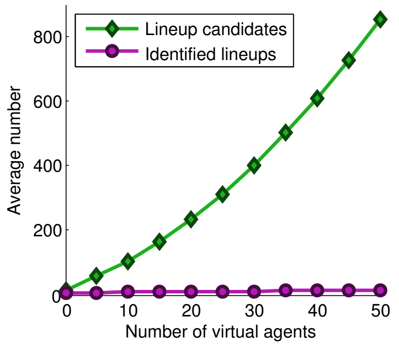
[3]We used the dataset, scenario, and code available at http://cv.postech.ac.kr/research/constflow/.



(a) Recall

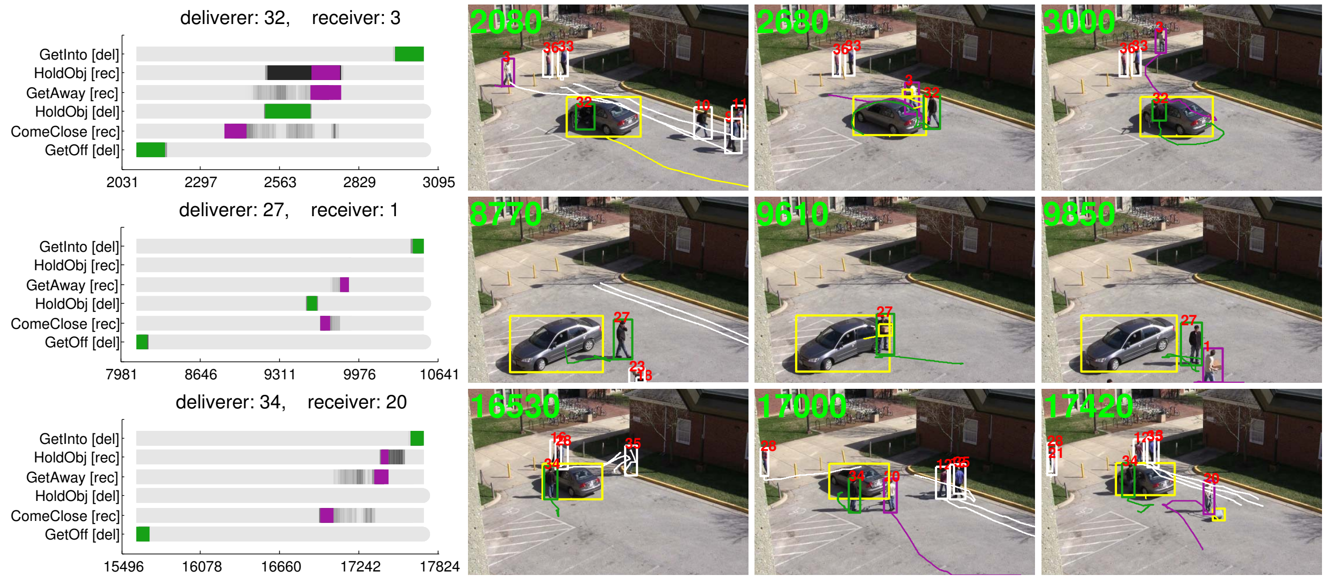


(b) Precision

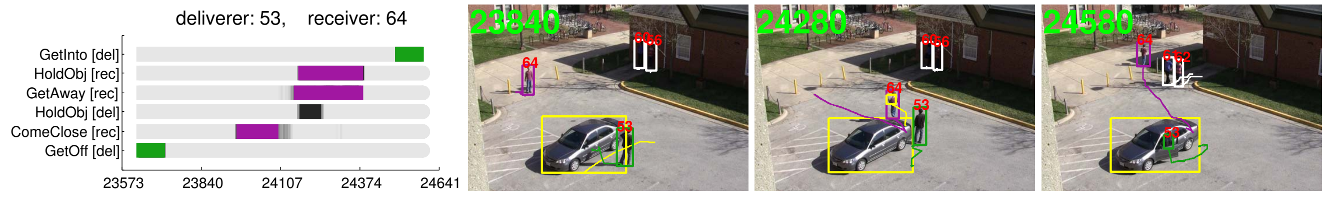


(c) Number of identified lineups

Figure 3. Simulation results by gradually adding up to 50 virtual agents. We controlled the ratio of virtual outsiders, whose outsider confidences are larger than all other role confidences. (a) Our method shows good recalls; most of the true lineups were retrieved despite a large number of virtual agents. (b) Precision gracefully degrades by increasing virtual agents due to lineups organized by virtual agents only. (c) Our method identified few probable lineups while the number of lineup candidates exponentially increases.



(a) Three Delivery events detected from *VIRAT1* sequence



(b) One Delivery event detected from *VIRAT2* sequence

Figure 4. Event detection results in (a) *VIRAT1* and (b) *VIRAT2*. In each graph, the horizontal axis represents time indices and vertical axis enumerates primitives of the target event. Primitive detection responses are represented by gray, where darker shade means higher response. Deliverers and receivers are denoted by green and purple, respectively, for both of the intervals on the graphs and bounding boxes in images; outsiders are indicated by white boxes in the images. In the sequences, outsiders often act like receivers near cars (*e.g.*, 2080, 16530, and 17420 of (a)), or they are spatially correlated with true deliverers and receivers (*e.g.*, 3000 of (a) and 24580 of (b)).

chine. Also, we extract motion context descriptors [21] from agents and recognize their actions by a nearest neighbor classifier. Then the primitives of Fieldgoal event are detected from the pose, action and relative positions of each agent. Despite a large number of groups, 1140 or more in total, and imperfect primitive detections, we successfully localize and detect the true target events in all the sequences as illustrated in Fig. 6.

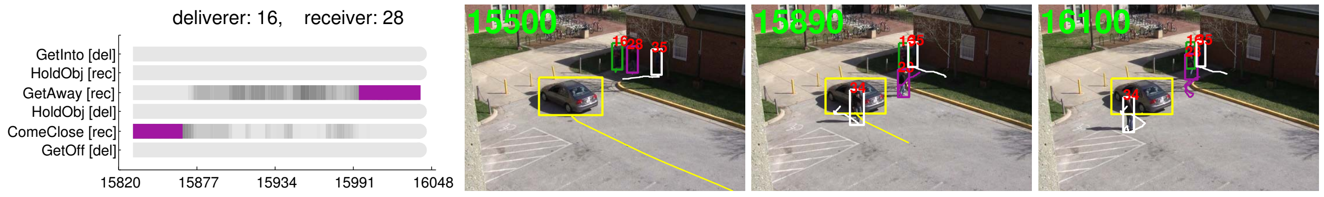


Figure 5. A misidentified lineup in *VIRAT1* sequence. Note that agent 28 acted like a receiver. This lineup was removed effectively by our verification process.

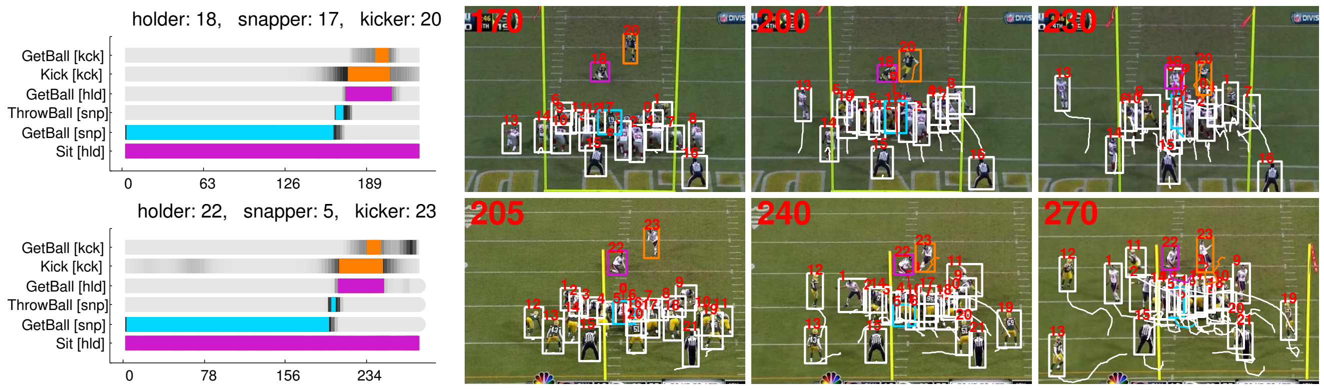


Figure 6. Results in two *fieldgoal* sequences. Holder, snapper, and kicker are denoted by purple, cyan, and orange, respectively while outsiders are denoted by white boxes. Additional results in other *fieldgoal* sequences are included in the supplementary file.

6. Conclusion

We presented a novel framework to automatically and efficiently detect unknown number of target events by identifying agents involved in events while simultaneously assigning roles to these agents. Our framework is formulated by a constrained optimization problem, which is solved by quadratic programming followed by linear programming. We demonstrated the performance of our framework on several challenging sequences in complex environments, and illustrated that it successfully localized and detected target events even when many agents including outsiders and multiple target events occur in videos.

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] A. Bobick and A. Wilson. A state-based approach to the representation and recognition of gesture. *TPAMI*, 1997. 2

[3] A. Bosch, A. Zisserman, and X. Munoz. Representing shape with a spatial pyramid kernel. In *CIVR*, pages 401–408, 2007. 6

[4] W. Brendel, A. Fern, and S. Todorovic. Probabilistic event logic for interval-based event recognition. In *CVPR*, 2011. 2

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

[6] K. L. Hoffman and M. Padberg. Set covering, packing and partitioning problems. In *Encyclopedia of Optimization*. 2009. 5

[7] C. Huang, B. Wu, and R. Nevatia. Robust object tracking by hierarchical association of detection responses. In *ECCV*, 2008. 6

[8] Y. A. Ivanov and A. F. Bobick. Recognition of visual activities and interactions by stochastic parsing. *TPAMI*, 2000. 2

[9] 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. 1

[10] S. Kwak, B. Han, and J. H. Han. Scenario-based video event recognition by constraint flow. In *CVPR*, 2011. 1, 2, 3, 5

[11] T. Lan, L. Sigal, and G. Mori. Social roles in hierarchical models for human activity recognition. In *CVPR*, 2012. 2

[12] 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. 2

[13] J. Löfberg. YALMIP : A toolbox for modeling and optimization in MATLAB. In *CACSD*, pages 284 –289, 2004. 5

[14] D. Minnen, I. Essa, and T. Starner. Expectation grammars: leveraging high-level expectations for activity recognition. In *CVPR*, volume 2, pages 626–632, 2003. 2

[15] V. I. Morariu and L. S. Davis. Multi-agent event recognition in structured scenarios. In *CVPR*, pages 3289–3296, 2011. 2

[16] S. Oh and et al. A large-scale benchmark dataset for event recognition in surveillance video. In *CVPR*, june 2011. 5

[17] M. S. Ryoo and J. K. Aggarwal. Semantic representation and recognition of continued and recursive human activities. *IJCV*, 2009. 2

[18] M. S. Ryoo and J. K. Aggarwal. Stochastic representation and recognition of high-level group activities. *IJCV*, 93(2):183–200, 2011. 2

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

[20] J. M. Siskind. Grounding the lexical semantics of verbs in visual perception using force dynamics and event logic. *JAIR*, 2001. 2

[21] D. Tran, A. Sorokin, and D. Forsyth. Human activity recognition with metric learning. In *ECCV*, pages 548–561, 2008. 1, 7

[22] S. D. Tran and L. S. Davis. Event modeling and recognition using markov logic networks. In *ECCV*, pages 610–623, 2008. 2

[23] C. Vondrick, D. Patterson, and D. Ramanan. Efficiently scaling up crowdsourced video annotation. *IJCV*, 2012. 6

[24] Z. Zhang, T. Tan, and K. Huang. An extended grammar system for learning and recognizing complex visual events. *TPAMI*, 2011. 2