Multi-Agent Event Detection: Localization and Role Assignment —–Supplementary Material—–

Suha Kwak Bohyung Han Joon Hee Han
Department of Computer Science and Engineering, POSTECH, Korea

1. Overview

This supplementary material presents contents omitted in our paper due to space limit. Section 2 describes the scenarios of the three target events. Section 3 introduces the scenario parsing algorithm that discovers temporal relationships between roles and constructs the three temporal relationship matrices (*i.e.*, $\mathbf{R}_{\text{ss}}$, $\mathbf{R}_{\text{ee}}$, and $\mathbf{R}_{\text{se}}$) automatically. Section 4 presents the proof omitted in Section 4.4 of the main paper. Finally, additional results in *fieldgoal* sequences are illustrated in Section 5.

Our event detection results are also demonstrated in a video. The video, '102.avi' is encoded with MPEG-4 Xvid codec, which can be found at `http://www.xvid.org`.

2. Scenarios of the target events

See Table 1 for the scenarios of the target events Transaction, Delivery, and Fieldgoal. Note that the scenario Transaction is described hierarchically due to its complexity.

3. How to build temporal relationship matrices

Our method penalizes a lineup candidate if its role-specific time intervals violate scenario constraints. However, a scenario describes constraints for relationships between primitives, not between roles. We need to specify scenario constraints in a role level to measure the fidelity of role-specific time intervals to the scenario constraints. For the purpose, we design three matrices that capture temporal relationships between roles: $\mathbf{R}_{\text{ss}}$ for relationships between starting times, $\mathbf{R}_{\text{ee}}$ for relationships between ending times, and $\mathbf{R}_{\text{se}}$ for relationships between starting and ending times (Eq. (6–8) of the main paper).

In this section, we illustrate how to construct the above three matrices from a given scenario automatically. We first discover temporal order of starting and ending times of primitives. The time interval of a role is specified by the first and last primitives of the role. Temporal order between a pair of roles is then determined by comparing starting and ending times of their first and last primitives.

3.1. Discovering temporal order of primitives

Let us assume that the given scenario consists of n primitives. We represent temporal orders between the primitives by two $n \times n$ matrices, $\mathbf{D}_{\text{s}}$ and $\mathbf{D}_{\text{e}}$, which indicate order of starting times and order of ending times of the primitives, respectively. Formally, $[\mathbf{D}_{\text{s}}]_{i,j}$ is r if the j-th primitive starts its interval earlier than the i-th primitive with respect to a binary temporal relationship operator r, and 0 otherwise. Similarly, $[\mathbf{D}_{\text{e}}]_{i,j}$ is r if the j-th primitive ends earlier than the i-th primitive with respect to r, and 0 otherwise. Note that $r \in \{<, \wedge, \sim\}$ by our description method.

$\mathbf{D}_{\text{s}}$ and $\mathbf{D}_{\text{e}}$ are automatically generated by parsing the given scenario. First, for each binary temporal relationship in the scenario, its backward and forward scopes are determined by adjacent parentheses and logical relationships (Figure 1(a)). Then, the parser determines temporal order between the backward and forward scopes by the definition of the binary temporal relationship (see Section 3.1 of the main paper), and the order is recorded in $\mathbf{D}_{\text{s}}$ and $\mathbf{D}_{\text{e}}$ as illustrated in Figure 1(b) and 1(c). Once $\mathbf{D}_{\text{s}}$ and $\mathbf{D}_{\text{e}}$ are constructed, we accumulate temporal predecessors per primitive; indirect predecessors of a primitive (*e.g.*, predecessors of predecessors of the primitive) are also considered as its predecessors in the matrices. The accumulations are done by

$$[\mathbf{D}_{\text{s}}]_{i,k} = \begin{cases} `<', & \text{if } [\mathbf{D}_{\text{s}}]_{i,j} = `<' \text{ and } [\mathbf{D}_{\text{s}}]_{j,k} \neq 0 \\ [\mathbf{D}_{\text{s}}]_{j,k}, & \text{else if } [\mathbf{D}_{\text{s}}]_{i,j} \neq 0 \\ [\mathbf{D}_{\text{s}}]_{i,k}, & \text{otherwise} \end{cases} , \quad (1)$$

$$[\mathbf{D}_{\text{e}}]_{i,k} = \begin{cases} `<', & \text{if } [\mathbf{D}_{\text{e}}]_{i,j} = `<' \text{ and } [\mathbf{D}_{\text{e}}]_{j,k} \neq 0 \\ [\mathbf{D}_{\text{e}}]_{j,k}, & \text{else if } [\mathbf{D}_{\text{e}}]_{i,j} \neq 0 \\ [\mathbf{D}_{\text{e}}]_{i,k}, & \text{otherwise} \end{cases} . \quad (2)$$

3.2. Construction of $\mathbf{R}_{\text{ss}}$

To determine the order of starting times between a pair of roles, we compare the first primitives of the roles. Let $\hat{\rho}_i$ and $\hat{\rho}_j$ be the first primitives of the i-th role and j-th role, respectively. If $\hat{\rho}_i$ starts its interval earlier than $\hat{\rho}_j$ (*i.e.*, $start(\hat{\rho}_i) < start(\hat{\rho}_j)$), the starting time of the i-th role is obviously earlier then that of the j-th role (*i.e.*, $Start(\gamma_i) <$

Transaction		
TakeItem$[\gamma]$	$\Rightarrow$	MoveDesk$[\gamma]$ $<$ MoveOut$[\gamma]$ $\wedge$ WithItem$[\gamma]$
BringItem$[\gamma]$	$\Rightarrow$	(WithItem$[\gamma]$ $\sim$ MoveDesk$[\gamma]$) $<$ MoveOut$[\gamma]$
TakeMoney$[\gamma]$	$\Rightarrow$	WithoutMoney$[\gamma]$ $<$ WithMoney$[\gamma]$ $\wedge$ MoveOut$[\gamma]$
BringMoney$[\gamma]$	$\Rightarrow$	WithMoney$[\gamma]$ $\wedge$ MoveDesk$[\gamma]$) $<$ WithoutMoney$[\gamma]$
ScanItem$[\gamma]$	$\Rightarrow$	WithItem$[\gamma]$ $\wedge$ (MoveScanner$[\gamma]$ $<$ MoveDesk$[\gamma]$)
Payment$[\gamma_{\text{ctm}}, \gamma_{\text{csh}}]$	$\Rightarrow$	(BringMoney$[\gamma_{\text{ctm}}]$ $\sim$ TakeMoney$[\gamma_{\text{csh}}]$) $<$ # | (BringMoney$[\gamma_{\text{csh}}]$ $\sim$ TakeMoney$[\gamma_{\text{ctm}}]$)
Transaction$[\gamma_{\text{ctm}}, \gamma_{\text{csh}}]$	$\Rightarrow$	(BringItem$[\gamma_{\text{ctm}}]^+$ $\sim$ ScanItem$[\gamma_{\text{csh}}]^+$) $<$ Payment$[\gamma_{\text{ctm}}, \gamma_{\text{csh}}]$ $<$ TakeItem$[\gamma_{\text{ctm}}]^+$

Delivery		
Delivery$[\gamma_{\text{del}}, \gamma_{\text{rec}}]$	$\Rightarrow$	GetOff$[\gamma_{\text{del}}]$ $<$ ComeClose$[\gamma_{\text{rec}}]$ & HoldObj$[\gamma_{\text{del}}]$ $<$ (GetAway$[\gamma_{\text{rec}}]$ $\wedge$ HoldObj$[\gamma_{\text{rec}}]$) & GetInto$[\gamma_{\text{del}}]$

Fieldgoal		
Fieldgoal$[\gamma_{\text{rec}}, \gamma_{\text{snp}}, \gamma_{\text{kck}}]$	$\Rightarrow$	Sit$[\gamma_{\text{rec}}]$ $\wedge$ (GetBall$[\gamma_{\text{snp}}]$ $<$ ThrowBall$[\gamma_{\text{snp}}]$ $<$ GetBall$[\gamma_{\text{rec}}]$ $\wedge$ (Kick$[\gamma_{\text{kck}}]$ $\wedge$ GetBall$[\gamma_{\text{kicker}}]$))

Table 1. Scenarios of the target events.

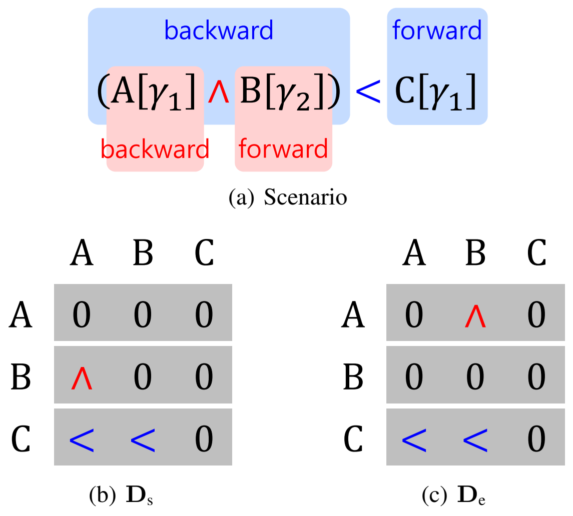


Figure 1. An example of $\mathbf{D}_s$ and $\mathbf{D}_e$ (best viewed in color). (a) The two relationship operators in the scenario, '$\wedge$' and '$<$' are denoted by red and blue, respectively. (b) Each binary relationship determines the order of starting times between its backward and forward scopes. In this example, '$<$' constrains its backward scope (A$[\gamma_1]$ and B$[\gamma_2]$) to start earlier than its forward scope (C$[\gamma_1]$); the corresponding elements in $\mathbf{D}_s$ are denoted by blue. (c) $\mathbf{D}_e$ represents the order of ending times in the same manner.

$Start(\gamma_j)$), and consequently $[\mathbf{R}_{\text{ss}}]_{i,j} = 1$ and $[\mathbf{R}_{\text{ss}}]_{j,i} = -1$. So, we focus on how to find the first primitives of each role and how to determine their order in starting times. Both of the above problems are solved simply by searching $\mathbf{D}_s$.

Let $\mathcal{P}_i$ denote the set of primitives involved in the i-th role. $\hat{\rho}_i$ is the first primitive of the i-th role if $\hat{\rho}_i \in \mathcal{P}_i$ and $[\mathbf{D}_s]_{\hat{\rho}_i,\rho_i} = 0$ for all $\rho_i \in \mathcal{P}_i$ because

$$\begin{aligned} &[\mathbf{D}_s]_{\hat{\rho}_i,\rho_i} = 0, \ \ \forall \rho_i \in \mathcal{P}_i \\ \Leftrightarrow\ & start(\hat{\rho}_i) \leq start(\rho_i), \ \ \forall \rho_i \in \mathcal{P}_i. \end{aligned} \tag{3}$$

Given $\hat{\rho}_i$ and $\hat{\rho}_j$, the corresponding element of $\mathbf{R}_{\text{ss}}$ is determined by comparing their order in starting times as follow:

$$[\mathbf{R}_{\text{ss}}]_{i,j} = \begin{cases} \infty, & \text{if } i = j, \\ 1, & \text{else if } [\mathbf{D}_s]_{\hat{\rho}_j,\hat{\rho}_i} \neq 0, \\ -1, & \text{else if } [\mathbf{D}_s]_{\hat{\rho}_i,\hat{\rho}_j} \neq 0, \\ 0, & \text{otherwise.} \end{cases} \tag{4}$$

3.3. Construction of $\mathbf{R}_{\text{ee}}$

We determine the order of ending times between a pair of roles by comparing the last primitives of the roles. Let $\check{\rho}_i$ and $\check{\rho}_j$ be the last primitives of the i-th role and j-th role, respectively. Note that $\check{\rho}_i$ is obtained by searching $\mathbf{D}_e$ in a similar manner to Section 3.2. Given $\check{\rho}_i$ and $\check{\rho}_j$, $[\mathbf{R}_{\text{ee}}]_{i,j}$ is determined by comparing their order in ending times as follow:

$$[\mathbf{R}_{\text{ee}}]_{i,j} = \begin{cases} \infty, & \text{if } i = j, \\ 1, & \text{else if } [\mathbf{D}_e]_{\check{\rho}_j,\check{\rho}_i} \neq 0, \\ -1, & \text{else if } [\mathbf{D}_e]_{\check{\rho}_i,\check{\rho}_j} \neq 0, \\ 0, & \text{otherwise.} \end{cases} \tag{5}$$

3.4. Construction of $\mathbf{R}_{\text{se}}$

An element of $\mathbf{R}_{\text{se}}$ indicates temporal order between the starting time of a role and the ending time of another role. Let $\hat{\rho}_i$ and $\check{\rho}_j$ be the first primitive of the i-th role and the last primitive of the j-th role, respectively. Then $[\mathbf{R}_{\text{se}}]_{i,j}$ is determined by comparing temporal order between $\hat{\rho}_i$ and $\check{\rho}_j$; it is more complicated than constructing $\mathbf{R}_{\text{ss}}$ and $\mathbf{R}_{\text{ee}}$, but also done by searching $\mathbf{D}_s$ and $\mathbf{D}_e$:

$$[\mathbf{R}_{\text{se}}]_{i,j} = \begin{cases} \infty, & \text{if } i = j, \\ 1, & \text{else if } [\mathbf{D}_s]_{\check{\rho}_j,\hat{\rho}_i} \neq 0, \\ 1, & \text{else if } [\mathbf{D}_e]_{\check{\rho}_j,\hat{\rho}_i} \neq 0, \\ -1, & \text{else if } [\mathbf{D}_e]_{\hat{\rho}_i,\check{\rho}_j} = <, \\ 0, & \text{otherwise.} \end{cases} \tag{6}$$

4. The proof for two-step optimization

When estimating true lineups, we substitute the two-step optimization (Eq. (10) and Eq. (13) in the main paper) for the original joint optimization (Eq. (9) in the main paper) to reduce search space significantly. The solutions of the two optimization problems are identical, which is proved by the following proposition.

Proposition 1. *The optimization problem in Eq. (9) of the main paper is equivalent to the joint optimization problem given by Eq. (10) and Eq. (13) of the main paper.*

Proof. For notational simplicity, let

$$\mathbf{h}(\mathbf{x}_1, \dots, \mathbf{x}_l) = \mathbf{f}(\mathbf{x}_1, \dots, \mathbf{x}_l) - \mathbf{A}^\top \mathbf{c}_0, \quad (7)$$

$$\widehat{\mathbf{h}} = \widehat{\mathbf{f}} - \mathbf{A}^\top \mathbf{c}_0. \quad (8)$$

Then

$$\widehat{\mathbf{h}}^\top \mathbf{y} \geq \mathbf{h}(\mathbf{x}_1, \dots, \mathbf{x}_l)^\top \mathbf{y}, \quad \forall \mathbf{x}_1, \dots, \mathbf{x}_l, \mathbf{y} \quad (9)$$

because $\mathbf{A}^\top \mathbf{c}_0$ is a constant vector, $\mathbf{y}$ is a binary vector, and $f_k(\widehat{\mathbf{x}}_k) \geq f_k(\mathbf{x}_k)$ for all $\mathbf{x}_k$ of all $k = 1, \dots, l$. The optimization problems in Eq. (9) and Eq. (13) of the main paper can be rewritten respectively by

$$\max_{\mathbf{y}, \mathbf{x}_1, \dots, \mathbf{x}_l} \quad \mathbf{h}(\mathbf{x}_1, \dots, \mathbf{x}_l)^\top \mathbf{y} \qquad \text{s.t.} \ \mathbf{A}\mathbf{y} \leq \mathbf{1}_m, \quad (10)$$

$$\max_{\mathbf{y}} \quad \widehat{\mathbf{h}}^\top \mathbf{y} \qquad \text{s.t.} \ \mathbf{A}\mathbf{y} \leq \mathbf{1}_m, \quad (11)$$

where the constant $\mathbf{c}_0^\top \mathbf{1}_m$ is dropped. Let $\mathbb{Y}$ be the set of feasible $\mathbf{y}$'s, *i.e.*, $\mathbb{Y} = \{\mathbf{y} | \mathbf{A}\mathbf{y} \leq \mathbf{1}_m, \mathbf{y} \in \{0, 1\}^l\}$, and $\widehat{\mathbf{y}}$ denote the solution of Eq. (11). By the inequality in Eq. (9),

$$\widehat{\mathbf{h}}^\top \widehat{\mathbf{y}} \geq \widehat{\mathbf{h}}^\top \mathbf{y} \geq \mathbf{h}(\mathbf{x}_1, \dots, \mathbf{x}_l)^\top \mathbf{y}, \quad \forall \mathbf{y} \in \mathbb{Y}. \quad (12)$$

Therefore, the maximum value of the optimization Eq. (11) is same with the maximum value of the optimization Eq. (10), which indicates the equivalence of the optimization problems in Eq. (13) and Eq. (9) of the main paper. □

5. Results in *fieldgoal* sequences

In this section, we present event detection results in all five *fieldgoal* sequences. Each *fieldgoal* sequence contains 20 or more agents and one Fieldgoal event. Because all agents are spatially close to others in the sequences, we counted all possible agent groups. Despite of a large number of groups (see Table 2), our method successfully localizes and detects the true target events with no misidentified lineups; the detection results are illustrated in Figure 2.

We compare our results with the naïve approach in terms of detection accuracy (Table 2) and execution time (Table 3). The naïve approach applies the event detection algorithm in Section 3 of the main paper to all possible lineup candidates, and rejects candidates if the numbers of hallucinations in their interpretations are larger than a threshold θ. Note that our method also verifies the identified lineups by the event detection algorithm in the same manner.

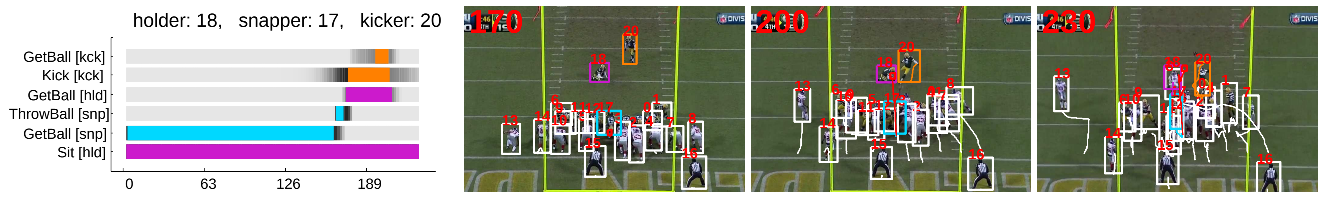
Our method achieves perfect accuracy in all the experiments except only one while the naïve approach typically has many false alarms. This is because our method can prevent conflicting lineups by using a single objective function while the naïve approach evaluates each lineup candidate independently. Imperfect primitive detection is another reason for the false alarms. For example, the primitives associated with the snapper role γ_{snp} (GetBall and ThrowBall) are also detected with high confidences from agents near the true snapper, and the action detector for the Kick primitive often fail to distinguish kicking and running. Our method is also 2$\sim$3 times faster than the naïve approach even though there are a large number of lineup candidates and the cost of event detection is relatively small.

		Role anal.	Lineup est.	Event det.	Overall
fieldgoal1	Ours	10.9	165.7	0.1	**176.6**
	Naïve	-	-	464.8	464.9
fieldgoal2	Ours	13.2	303.2	0.1	**316.4**
	Naïve	-	-	826.2	826.4
fieldgoal3	Ours	16.7	284.2	0.1	**301.0**
	Naïve	-	-	1063.5	1063.7
fieldgoal4	Ours	14.6	343.6	0.1	**358.3**
	Naïve	-	-	899.5	899.7
fieldgoal5	Ours	14.3	224.8	0.1	**239.2**
	Naïve	-	-	839.1	839.3

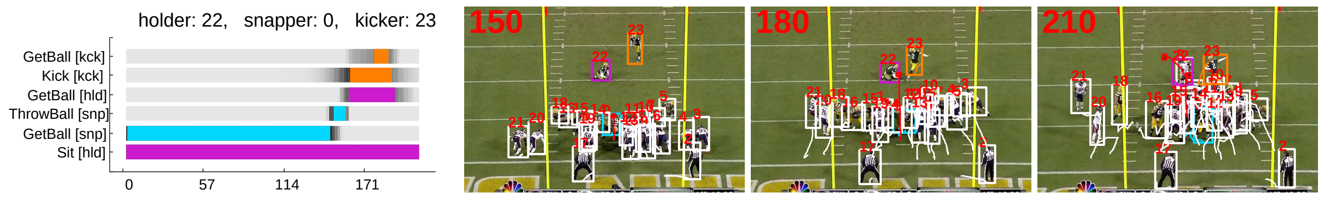
Table 3. Execution times in seconds in *fieldgoal* sequences.

			Number of detected events (precision/recall)					
			$\theta = 2$		$\theta = 1$		$\theta = 0$	
	Agents	Groups	Ours	Naïve	Ours	Naïve	Ours	Naïve
fieldgoal1	20	1140	1 (**1.00**/1.00)	56 (0.02/1.00)	1 (**1.00**/1.00)	6 (0.17/1.00)	1 (**1.00**/1.00)	2 (0.50/1.00)
fieldgoal2	24	2024	1 (**1.00**/1.00)	132 (0.01/1.00)	1 (**1.00**/1.00)	9 (0.11/1.00)	1 (**1.00**/1.00)	4 (0.25/1.00)
fieldgoal3	24	2024	1 (**1.00**/1.00)	107 (0.01/1.00)	1 (**1.00**/1.00)	5 (0.20/1.00)	1 (**1.00**/1.00)	3 (0.33/1.00)
fieldgoal4	24	2024	1 (**1.00**/1.00)	131 (0.01/1.00)	1 (**1.00**/1.00)	4 (0.25/1.00)	0 (0.00/0.00)	0 (0.00/0.00)
fieldgoal5	23	1771	1 (**1.00**/1.00)	301 (0.00/1.00)	1 (**1.00**/1.00)	9 (0.11/1.00)	1 (**1.00**/1.00)	3 (0.33/1.00)

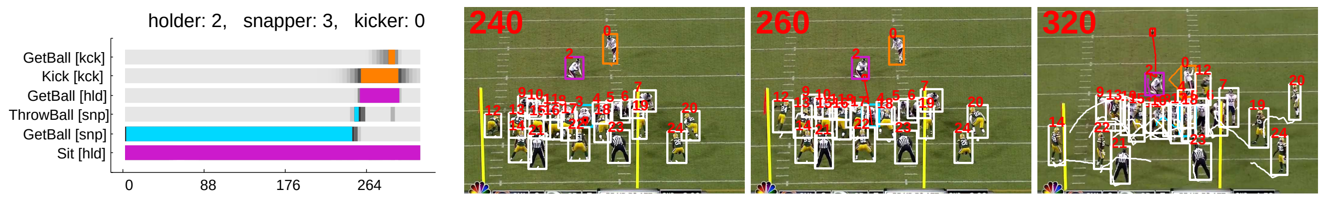
Table 2. Event detection results by varying the threshold for the number of hallucinations (θ) in *fieldgoal* sequences.



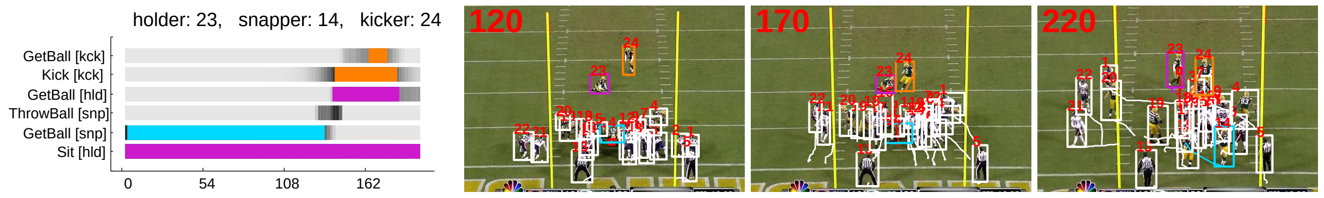
(a) Detection result in *fieldgoal1* sequence



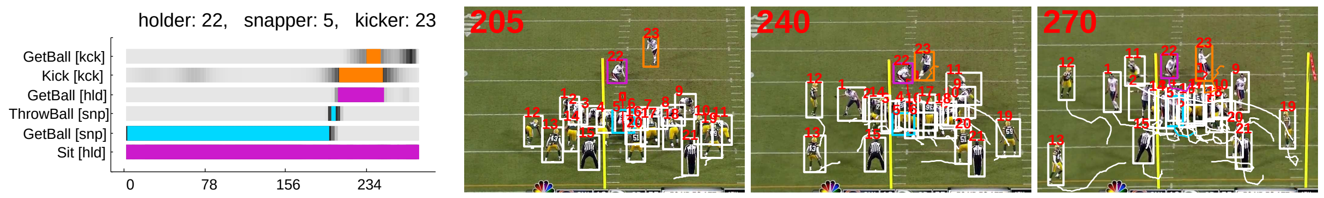
(b) Detection result in *fieldgoal2* sequence



(c) Detection result in *fieldgoal3* sequence



(d) Detection result in *fieldgoal4* sequence



(e) Detection result in *fieldgoal5* sequence

Figure 2. Results in the five *fieldgoal* sequences; only (a) and (e) are given in the main paper. Holder, snapper, and kicker are denoted by purple, cyan, and orange, respectively while outsiders are denoted by white boxes.