Multi-agent event recognition in structured scenarios

Vlad I. Morariu and Larry S. Davis
Institute for Advanced Computer Studies
University of Maryland, College Park, MD 20742
{morariu,lsd}@cs.umd.edu

Abstract

We present a framework for the automatic recognition of complex multi-agent events in settings where structure is imposed by rules that agents must follow while performing activities. Given semantic spatio-temporal descriptions of what generally happens (i.e., rules, event descriptions, physical constraints), and based on video analysis, we determine the events that occurred. Knowledge about spatio-temporal structure is encoded using first-order logic using an approach based on Allen's Interval Logic, and robustness to low-level observation uncertainty is provided by Markov Logic Networks (MLN). Our main contribution is that we integrate interval-based temporal reasoning with probabilistic logical inference, relying on an efficient bottom-up grounding scheme to avoid combinatorial explosion. Applied to one-on-one basketball, our framework detects and tracks players, their hands and feet, and the ball, generates event observations from the resulting trajectories, and performs probabilistic logical inference to determine the most consistent sequence of events. We demonstrate our approach on 1hr (100,000 frames) of outdoor videos.

1. Introduction

The automated analysis of multi-agent activity is difficult due to interactions that lead to large state spaces and complicate the already uncertain low-level processing. Often, activities must satisfy rules that impose a spatio-temporal structure on constituent events. This structure can be leveraged to disambiguate amongst complex activities. For example, in the case of one-on-one basketball, offensive and defensive rebounds are often ambiguous, since both players are near each other as they reach for the ball. However, the rules of half-court basketball can reduce this ambiguity by relating the rebound event to other less ambiguous events; e.g., if the ball were shot shortly after the rebound without any of the players running back to the three-point line, then an offensive rebound must have occurred, since a defensive rebound requires the player to *clear* the ball first (i.e., taken to the three-point line) before taking a shot.

Our goal is to create a framework that, given a semantic description of what generally happens (i.e., rules, meaning of relevant events), determines the events that occurred. We test our framework on one-on-one basketball games, in which two players interact, but event structure is non-trivial, and visual recognition is hampered by players frequently occluding each other. We do not use human annotations such as text, camera movement, shot-changes, or overlaid statistics (unlike [7, 26]), which are typically used to analyze sports videos, as we seek a framework that can analyze broader classes of human/object interactions. The stationary camera simplifies image to court registration, but it also removes important information that a human operated camera provides; e.g, camera movements reveal possession, and shot-changes provide a partial temporal segmentation.

We analyze single camera videos of one-on-one basketball in the context of court annotations (i.e., hoop and points on the court plane), and spatio-temporal relations describing the rules and events of interest. We automatically detect and track players, their hands and feet, and the ball, generating a set of trajectories which are used in conjunction with spatio-temporal relations to generate event observations. Knowledge about spatio-temporal event structure is expressed in first-order logic using a principled and extensible approach based on Allen's Interval Logic [1]. Robustness to low-level observation uncertainty is provided by Markov Logic Networks (MLN) [3], which attach weights to first-order logic formulas and dynamically construct Markov networks representing hypothesized events, spatio-temporal relationships, and low-level observations. Inference on this Markov network determines which high-level events (e.g., *check*, *dribble series*, *shot*) occurred (see figure 1).

Our main contribution is a system that efficiently and robustly recognizes events in structured scenarios from noisy visual observations by combining (1) visual analysis of people and object movements, (2) a powerful and natural event reasoning representation based on Allen's Interval Logic, (3) probabilistic logical inference via MLNs, and (4) efficient bottom-up event hypothesis generation. Although in

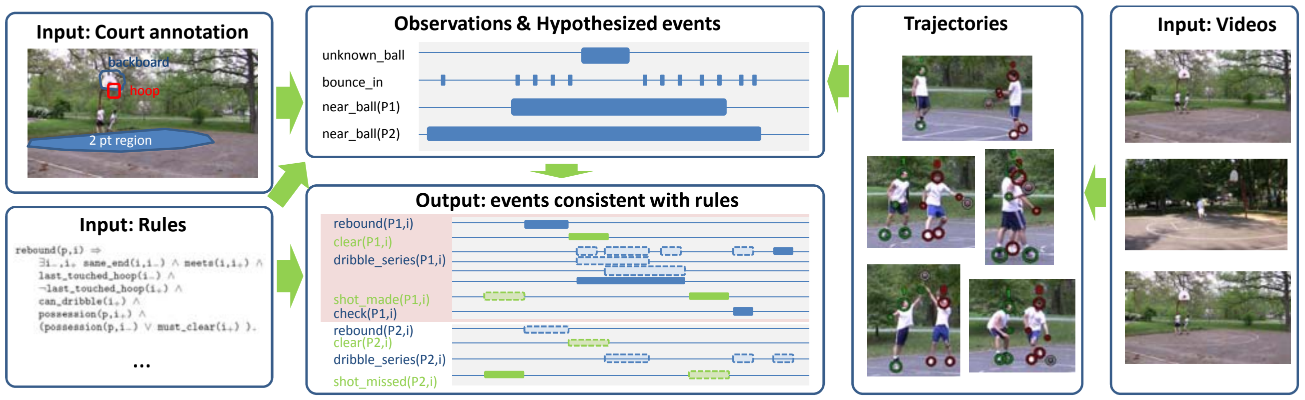

Figure 1. Framework overview. Our system obtains a set of trajectories from the input videos, which are then processed generate observations (e.g., when the ball bounced, was near a player, or was near the hoop). Based on these observations and event definitions, hypothesized event intervals are generated in a bottom-up fashion. Finally, MLN inference determines the set of hypothesized intervals that are most consistent with the observations and the formulas in the knowledge base. In this example, intervals with solid boundaries are the final detected events, while those with dashed boundaries are hypothesized intervals that were discarded by MLN inference.

our experiments we can efficiently perform exact inference on entire sequences at once, we show that our approach also performs well in an online streaming setting.

2. Related work

Hidden Markov Models (HMMs) [17] have been successfully applied to action recognition tasks, but their performance degrades as the state size increases (much more data is needed to train an accurate model); this is a problem, since multi-agent interaction models generally require a large state space. To deal with this complexity in highly coupled T'ai Chi hand movements, Brand *et al.* [2] presented coupled HMMs, which factorize the joint transition table into two smaller transition tables. Shi *et al.* [21] used Propagation Nets (P-Net), an extension of Dynamic Bayesian Networks (DBNs) that models duration and can represent complex activities including concurrent events, but requires manual specification of state connectivity. Unfortunately, HMM and DBN extensions generally assume fixed number of actors and objects, do not handle missing observations well, and require large training sets to learn structure that a human could easily describe.

Expert domain knowledge can be leveraged to create models of multi-agent activities. Intille and Bobick [9] recognize football plays by using temporal constraints to dynamically construct complex action Bayes nets from smaller manually specified Bayes nets that relate agent goals to visual evidence. Perse *et al.* [16] analyze team activities in basketball games by transforming trajectories into a sequence of semantically meaningful symbols and comparing them to templates provided by domain experts. Ryoo and Aggarwal [19] model two person interactions by a context-free grammar (CFG), where high-level interactions are defined hierarchically using logical spatial and temporal predicates on sub-actions. Their atomic actions are detected using HMMs, but CFG parsing is not probabilistic and can be sensitive to low-level failures. Similarly, Store Totally/Partially Recognized Scenario (STRS/SPRS) [5, 24] approaches efficiently recognize multi-agent scenarios, but are symbolic and do not account for low level uncertainty. To introduce robustness to inconsistent first-order logic knowledge-bases (e.g., due to low-level errors, or imperfect rules), Tran and Davis [25] used Markov Logic Networks (MLN) [3] to analyze simple person-person and person-vehicle interactions. Similarly, Sadilek and Kautz [20] analyzed multi-agent interactions from GPS data, using MLNs to jointly denoise low-level data and incorporate temporally distant events. Their rules focus on a single event, *capture*, in the game of capture the flag.

Multi-agent activities have also been analyzed with little or no supervision. Gupta *et al.* [7] use label data loosely associated with videos during training to automatically learn the spatio-temporal structure of baseball plays. Siracusa and Fisher [22] infer the interaction dependency structure in basketball games using a directed temporal interaction model and a latent variable to allow interaction dependency structures to change over time. While the latent variable state sequence is sampled by Markov Chain Monte Carlo (MCMC), given a state sequence, the posterior over dependency structures is obtained efficiently by exact inference. Sridhar *et al.* [23] perform unsupervised learning of events by modeling interactions between tracks as a relational graph structure that captures spatio-temporal relationships, clustering events by MCMC. These methods can be useful in learning multi-agent event patterns, but require large training sets to learn constraints.

Our approach is similar to [25, 20], as it leverages expert domain knowledge, expressed in first-order logic, and

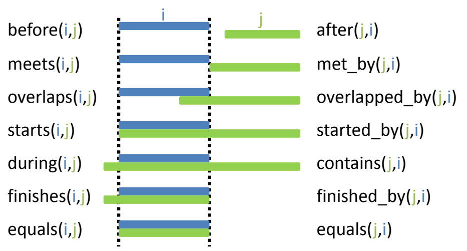

Figure 2. The base interval relations in Allen's Interval Logic.

performs logical inference probabilistically using MLNs for robustness to noisy observations and knowledge base inconsistencies. Unlike [25, 20], our approach also incorporates a powerful and natural representation based on Allen's Interval Logic to reason about complex spatio-temporal relationships between multiple properties, events, and observations. Because of the expressiveness of our approach, we do not require large training sets; in fact, in our experiments, all knowledge is provided manually via rules, though probabilistic observations can be incorporated as in [25].

3. Event reasoning

To describe our event reasoning framework, we continue to use the one-on-one basketball scenario introduced in section 1 as the target application. Given the ball and player trajectories, court homography, and hoop location, our framework uses knowledge about basketball rules, events, and physical constraints to generate and evaluate hypothesized events. This knowledge is expressed using first-order logic, following the example of Allen *et al.* [1], where events are defined by their interactions with properties of the world (see table 1). Observations computed from the trajectory and court annotations are incorporated into the knowledge base using a set of soft rules. To avoid computational complexity, instead of considering all $O(T^2)$ possible intervals for each event, our system uses the rules themselves to generate bottom-up event hypotheses from observations, aiming for a high recall ratio, while avoiding events that are unlikely given our rules (e.g., if the rules say that for a shot to occur, the ball must be in the air, then a hypothetical `shot` event is generated only when the ball is observed in the air). This bottom-up process may not always generate events that are consistent with the rules (the ball being in the air does not necessarily mean that a shot was attempted), so we use probabilistic inference to determine which set of hypothesized event candidates most likely occurred, given the observations and the rules. Figure 1 illustrates this process.

3.1. Interval logic representation

The rules of basketball are non-trivial, even for the one-on-one case, so we need a principled approach to representing the rules and how they relate both to the state of the

Properties	
possession(p, i)	player p has possession during interval i; mutually exclusive and exhaustive between players
last_touched_hoop(i)	no player touched the ball since it last hit the hoop
can_dribble(i)	once dribble series ends, ball cannot be dribbled again until after a rebound, check, or steal
must_clear(i)	after a defensive rebound, ball handler must clear the ball by taking it to three-point line
must_check(i)	ball must be checked to player who has possession after a shot is made or ball goes out of bounds
Events	
shot_{*}(p, i)	the shot events, shot_made and shot_missed
dribble_series(p, i)	complete series of continuous ball bounces performed by player p during interval i
check(p, i)	sequence of passes to/from offensive player p, who is outside three-point line; last pass to p resumes play
rebound(p, i)	begins when ball falls from hoop (after a missed shot), and ends when player p obtains ball
clear(p, i)	after a defensive rebound player p clears ball by taking it to the three-point line
steal(p, i)	player p steals ball from other player, not by a rebound or out_of_bounds event
out_of_bounds(p, i)	starts when ball is out of bounds and ends when the ball is brought back on the court
Observations	
obs_in_air(i)	ball is in the air; implies that someone took a shot
obs_possession(p, i)	implies possession(p, i) with weight prop. to # frames ball is nearest p and p is farthest from hoop
obs_shot_made(p, i)	ball seen in air, obs_possession(p, i_-) is true before shot, obs_near_hoop(i_+) is true at end of shot
obs_shot_missed(p, i)	ball seen in air, obs_possession(p, i_-) is true before shot, obs_near_hoop(i_+) is not true at end of shot
obs_check(p, i)	sequence of passes with ball ending near p (pass observed by switches in obs_nearest_ball(p, i))
obs_dribble(p, i)	at least one bounce near p was observed

Table 1. Property and event predicates are used as queries. Observation predicates used as evidence in observation rules are shown; others such as `obs_near_hoop`, `obs_near_ball`, etc., are not listed.

game and to visual observations. For this purpose, we adopt a framework similar to that proposed by Allen *et al.* [1], where predicates are grouped into three categories: properties, events, and actions. Temporal relationships between these predicates, which are defined on time intervals, are expressed using the following base binary relations and their inverses: `before`, `meets`, `overlaps`, `starts`, `during`, `finishes`, and `equals`. These relations are illustrated in figure 2. Properties describe the relevant parts of the state of the world; events change these properties when they occur, as long as prerequisite properties hold before the event's occurrence; finally, actions are *programs* that an agent (such as a robot) executes in order to cause events to occur. This last category makes use of a *Try* predicate which indicates that an agent attempts to perform an action, which if successful, brings about one or more events. In our case, the system is a passive observer, so it cannot perform actions to bring about changes; thus, we ignore the *action* category described in [1]. Instead, we explicitly model observations with rules that generally hold true (but not always, due to mistakes in visual analysis, or because these rules are *rules-of-thumb*). Below we describe the categories of predicates and related axioms.

Properties. Properties describe the state of the world. In one-on-one basketball, the relevant properties are $\texttt{possession}(p, i)$, $\texttt{last\_touched\_hoop}(i)$, $\texttt{can\_dribble}(i)$, $\texttt{must\_clear}(i)$, and $\texttt{must\_check}(i)$ (see table 1 for descriptions).

Events. An event is defined by the prerequisite values of relevant properties prior to, during, and after its occurrence. The occurrence of an event could also imply that a related event occurred or that other events could not have occurred. Allen *et al.* [1] group event related axioms into *event definition*, *event generation*, *action definition*, and *event explanation closure* categories. We adopt these categories, excluding *action definition*, and add another category, *event mutual exclusion*. The *event definition* axioms are of the form $\texttt{event}(i) \wedge \phi \Rightarrow \psi$, where ϕ and ψ are expressions that contain temporal constraints between the event and relevant properties. For the rebound event,

$$\begin{aligned}&\texttt{rebound}(p, i) \Rightarrow\\ &\quad \exists i_-, i_+ \texttt{same\_end}(i, i_-) \wedge \texttt{meets}(i, i_+) \wedge\\ &\quad \texttt{last\_touched\_hoop}(i_-) \wedge \neg\texttt{last\_touched\_hoop}(i_+) \wedge\\ &\quad \texttt{can\_dribble}(i_+) \wedge \texttt{possession}(p, i_+) \wedge\\ &\quad (\texttt{possession}(p, i_-) \vee \texttt{must\_clear}(i_+))\end{aligned}$$

is an *event definition* axiom which states that for the rebound event to occur over interval i, person p first touches the ball at the end of the rebound event, and can then dribble the ball; p has possession after the rebound, and if p did not initially have possession, then the ball must be cleared. The *event generation* axioms are of the form $\texttt{event}(i) \wedge \phi \Rightarrow \exists i' \texttt{event}'(i') \wedge \psi$. In the basketball scenario, such a rule might say that if a shot event occurs, either a jump-shot, layup, or set-shot occurs. The *event explanation closure* axioms encode the assumption that only known events change properties, so if a property changed, an event affecting this property must have occurred. For example,

$$\begin{aligned}&\texttt{can\_dribble}(i') \wedge \neg\texttt{can\_dribble}(i) \wedge \texttt{meets}(i', i) \Rightarrow\\ &\quad \exists p, i'' \texttt{dribble\_series}(p, i'') \wedge \texttt{meets}(i'', i)\end{aligned}$$

states that for can_dribble to change from true to false, a dribble event must have occurred, after which the property transitions from true to false. Finally, *event mutual exclusion* axioms (not explicitly included in [1]) encode the constraint that some events cannot occur simultaneously.

$$\begin{aligned}&\texttt{intersects}(i_1, i_2) \wedge \texttt{dribble\_series}(p_1, i_1) \wedge\\ &\quad (i_1 \neq i_2 \vee p_1 \neq p_2) \Rightarrow \neg\texttt{dribble\_series}(p_2, i_2)\end{aligned}$$

This axiom states that a person can only take part in one dribble_series event at one time, and only one person at a time can dribble. The temporal relation intersects is a disjunction of a subset of the base temporal constraints and their inverses that is true if the intersection of the intervals results in a time interval of some positive length.

Observations. Observations about the world could imply certain events happened, or that certain values of properties are more likely than others. These were not included explicitly in Allen *et al.* [1], but we include them since they determine the likelihoods of events that occurred.

$$\begin{aligned}&\texttt{obs\_nearest\_ball}(p, i) \wedge \neg\texttt{obs\_nearest\_hoop}(p, i) \Rightarrow\\ &\quad \texttt{possession}(p, i)\end{aligned}$$

These rules may be inconsistent for two reasons: 1) observations are generated by video processing, which may include mistakes, and 2) some observation rules encode *common sense* knowledge that generally holds, but may at times lead to an inconsistent knowledge base. In the example above, when the ball is nearest the player who is farthest from the hoop, it generally means that player has possession of the ball, but this may not always be true (e.g., immediately after a rebound, or after a successful drive to the hoop). These potential inconsistencies are dealt with by our inference approach by allowing these rules to be treated as soft rules (i.e., a truth assignment can break these rules and incur a relatively low cost compared to, say, violating the rules of basketball).

3.2. Bottom-up event hypothesis generation

To avoid the computational cost of considering all $O(T^2)$ intervals for each event and property (assuming T frames), we generate a set of intervals which is small but has a high recall rate. To achieve this, we consider only event intervals that are implied by observations, using the logic rules themselves to generate candidate intervals for events and properties. For example, since a dribble_series event is related by the observation rules to predicates such as $\texttt{obs\_bounce\_in}(i)$ and $\texttt{obs\_ball\_near}(p, i)$, we can use these predicates to generate hypothesized start and end times for a dribble series. Figure 3 depicts this process for the dribble_series event. Given the observation predicates, a small set of start times and end times is created, and from these two sets, a small set of intervals is created by pairing start and end times that are consistent with each other (end time is after start time, etc). A similar process is performed for all of the events shown in table 1: observed start and end times for observation predicates are used to create hypothesized start and end times for event predicates. We employ the *closed world assumption* (CWA) and assume that predicates grounded over intervals that are not hypothesized are false. For this reason, it is important that recall is as high as is computationally practical, since missed intervals will not be detected by further inference due to the CWA, and may cause related formulas to be violated or related events be incorrectly missed/detected. Conversely, if too many false positives are introduced in order to achieve high recall, then the number of ground atoms and inference complexity will increase.

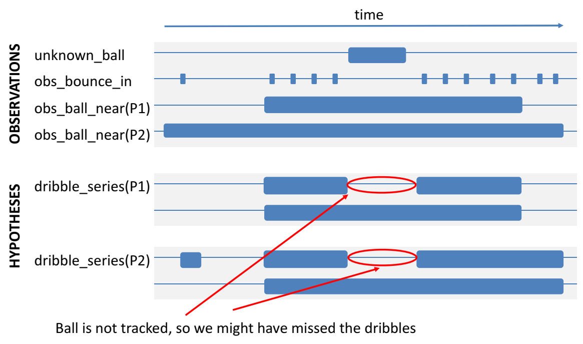

Figure 3. The bottom-up event generation module generates hypothesized event intervals from low-level observations, with the goal of achieving a high recall rate with a reasonably small set of intervals. Here, seven intervals are generated from observations for which $\texttt{dribble\_series}(p, i)$ is open world (could be true).

When combined with the event explanation closure axioms presented earlier, the hypothesized event intervals lead to a set of times when properties *might* change value. For example, if all properties change values either at the beginning or end of an event, then all unique hypothesized event start and end times can be collected to discretize time (non-uniformly). The intervals considered for properties are those with start and end times that are consecutive in the ordered list of event interval start and end times (these time periods are called *moments*); thus, if there are M unique times that appear as start or end times in hypothesized event intervals, there will be $M+1$ moments over which property predicates can be grounded. See figure 4 for an illustration of moments and general intervals once time is discretized. Moments are exhaustive and mutually exclusive, so often only a single moment will satisfy an existential quantifier. Consequently, we can remove some existential quantifiers, which can be computationally expensive. For example, the first two lines of the rebound event definition in section 3.1 can be replaced by the following:

$$\texttt{rebound}(p, i) \wedge$$
$$\texttt{started\_by\_moment}(i, i_-) \wedge \texttt{meets\_moment}(i, i_+) \Rightarrow$$

Once intervals are hypothesized, computational complexity may be further reduced by converting remaining existential quantifiers to disjunctions of conjunctive clauses, removing clauses that are always false given evidence such as known temporal relations and observation predicates.

3.3. Probabilistic inference using Markov Logic Networks

The knowledge base is likely to contain inconsistencies, either due to noisy or missed observations, or due to imperfect rules that occasionally do not hold. For this reason, we relax these rules and perform queries probabilistically using Markov Logic Networks (MLN) [3]. Markov

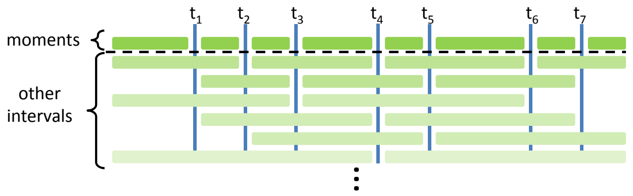

Figure 4. The space of all intervals once time is discretized. Property predicates are grounded over all moments. Event predicates are grounded over general intervals (which include moments), but only over those that are hypothesized. In our approach, time is discretized by the start and end times of hypothesized events and the times at which properties could change, given the hypotheses.

Logic Networks relax first-order logic by attaching a weight to each formula, such that when a world violates a formula, that world becomes less probable instead of becoming impossible. More formally, Domingos *et al.* [3] define an MLN as follows. An MLN consists of a set of first-order logic formulas F_i, associated real weights w_i and a finite set of constants $C = \{c_1, c_2, \ldots, c_{|C|}\}$. An MLN can then be viewed as a *template* for dynamically constructing a Markov network, given a set of constants. For a given set of constants, C, the network is constructed by creating one binary node for each grounding of each predicate, which takes value 1 if that ground predicate is true and 0 if it is false. Each possible grounding of each formula F_i will then have an associated feature, which will have a value of 1 if that formula is satisfied, and 0 if it is not. Each feature will have an associated weight w_i. In the factor graph representation of the Markov network, ground predicates become nodes and formulas become factors defined over these nodes. The probability distribution of a world x is then given by $P(X = x) = \frac{1}{Z} \exp\left(\sum_{i=1}^{F} w_i n_i(x)\right)$, where $n_i(x)$ is the number of true groundings of formula F_i in x, F is the total number of formulas, and Z is a normalizing constant. Inference on this Markov network can then be performed using standard techniques. A theoretically desirable property of MLNs is that they can represent (and augment) formalisms such as BNs, HMMs, DBNs, and CFGs.

Predicates and formulas in our application contain three types of variables – moment, interval, and person – so the set of constants will include the $M+1$ unique moments, the I intervals, and the two players. Although weights can be learned for each formula, we manually set the weights using intuitive values; for example, formulas or axioms that describe constraints imposed by basketball rules have high weight, but *common sense* or observation formulas have lower weight, as there is a larger chance that they could cause the knowledge base to be inconsistent. We use Alchemy[1] [11] to generate ground MLNs, and AND/OR Branch-and-Bound [13] to perform exact inference.

[1]http://www.cs.washington.edu/ai/alchemy

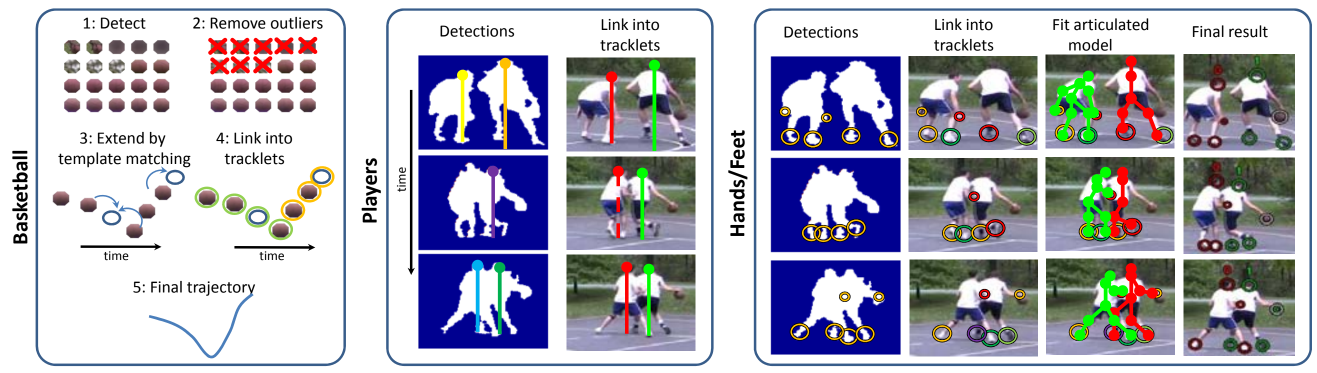

Figure 5. Tracking overview. Basketball, head, hand, and foot candidates are first detected and are then linked into tracklets.

`shot_{*}`(p, i)	Hypothesized when `obs_in_air`(i) is true; as with dribble series (Fig 3), multiple hypothesized shot events may be merged to create new hypotheses if the ball location is briefly unknown while in flight.
`dribble_series`(p, i)	In-bounds bounces, `obs_bounce_in`(i), that occur during the same `obs_ball_near`(p, i) interval, are grouped to generate hypotheses, which may be merged as in Fig 3 to account for lost tracks.
`check`(p, i)	Alternating passes between players are detected by switches in `obs_nearest_ball`(p, i) and are grouped together to generate a hypothesized check interval if the final pass is to player p outside the three-point region `obs_inside_3pt`(p, i).
`rebound`(p, i)	An interval is hypothesized for each player by pairing the end of an `obs_in_air`(i) interval with the start of the next encountered `obs_ball_near`(p, i) interval.
`clear`(p, i)	Hypotheses are generated by pairing the end times of hypothesized rebound intervals and the first encountered `obs_inside_3pt`(p, i) interval start time after the end of the rebound.
`steal`(p, i)	Hypothesized when switches in observed possession, determined from `obs_nearest_ball`(p, i) and $\neg$`obs_nearest_hoop`(p, i), are not explained by `out_of_bounds` or `missed_shot` interval hypotheses.
`out_of_bounds`(p, i)	Hypothesized intervals start when the ball or player are first observed out of bounds (via `obs_bounce_out`(i) and `obs_person_out_of_bounds`(p, i)) and end when the player is back in bounds. Additional hypothesized intervals are added if a possession switch and out-of-bounds bounce are observed near each other.

Table 2. Bottom-up event hypothesis generation from trajectory-based observations.

4. Experiments

We demonstrate our approach on a dataset consisting of 7 outdoor sequences of one-on-one basketball (roughly 100,000 frames at 30fps, or 1hr of video), with varying camera positions, and 7 unique players. These videos contain varying illumination conditions (two are collected right before sunset and contain strong shadows), and 5 out of the 7 sequences contain full games to 11 points. The static annotation provided by the user includes hoop and backboard polygons and the homography from camera view to court plane (using 5-7 pairs of points on the court).

4.1. Tracking and hypothesis generation

The low-level part of our system detects and tracks the basketball, players, and their hands and feet, providing their trajectories to the high-level component (see figure 5). Videos are preprocessed by first computing optical flow [15] and then detecting moving foreground pixels using background subtraction [10]. To handle slow outdoor lighting changes caused by clouds and changes in relative position of the sun, we split videos into smaller segments and use the same segment for training and testing, using flow to mask out moving pixels during the training phase. Once foreground pixels are obtained, we use silhouette features to detect ball, head, hand, and foot candidates (based on roughly circular blobs for the ball and high contour curvatures for the head, hands, and feet). We link detections into tracklets using a data association approach based on [8]. Before linking, ball outliers are removed by k-center clustering [6], which generally places outliers in small isolated clusters. After linking, at most two hand and foot tracklets are chosen for each player person by an approach based on pictorial structures [4], with additional constraints added to account for interactions between players due to occlusion [14]. Our framework does not yet include formulas to handle player identity switches, so additional human input is needed after player tracking to merge/split tracklets. Head location tracklets are shown to the user in an X-T plot, so that as many as 1600 frames can be inspected at a time. The user does not add or modify detections, but provides only identities of tracklets where necessary to prevent merges/splits.

These trajectories are then processed in the context of the court annotations to yield the set of deterministic observations. These observations include whether or not the ball is in the air (`obs_in_air`(i)), the ball is near or nearest one of the two players (`obs_ball_near`(p, i), `obs_nearest_ball`(p, i)), a player is inside the three-point region (`obs_inside_3pt`(p, i)), a player is nearest the hoop (`obs_nearest_hoop`(p, i)), the ball bounced in or out of bounds (`obs_bounce_in`(i), `obs_bounce_out`(i)), a player stepped out of bounds (`obs_person_out_of_bounds`(p, i)), or the ball is near the hoop (`near_hoop`(i), implies a shot was made). Hand locations are useful for determining the time intervals dur-

	TP	FP	FN	P	R	F1
Hand tracks	2095	475	1255	.82	.61	.70
Foot tracks	3600	339	720	.91	.82	.86
Ball tracks	1059	76	87	.93	.92	.93
Ball bounces	183	2	65	.99	.74	.85

Table 3. Tracking performance.

	TP	FP	FN	P	R	F1
Check	103	20	11	.84	.90	.87
Clear	122	282	15	.30	.89	.45
Dribble	226	436	27	.34	.89	.49
OutOfBounds	26	61	6	.30	.81	.44
Rebound	153	426	6	.26	.96	.41
ShotMade	75	559	2	.12	.97	.21
ShotMissed	166	468	5	.26	.97	.41
Steal	2	50	2	.04	.50	.07
Overall	**873**	**2302**	**74**	**.27**	**.92**	**.42**

Table 4. Event hypothesis performance.

	TP	FP	FN	P	R	F1
Check	102	19	12	.84	.89	.87
Clear	83	13	54	.86	.61	.71
Dribble	189	45	64	.81	.75	.78
OutOfBounds	21	3	11	.88	.66	.75
Rebound	115	71	44	.62	.72	.67
ShotMade	66	37	11	.64	.86	.73
ShotMissed	135	67	36	.67	.79	.72
Steal	2	24	2	.08	.50	.13
Overall	**713**	**279**	**234**	**.72**	**.75**	**.74**

Table 5. MLN recognition performance.

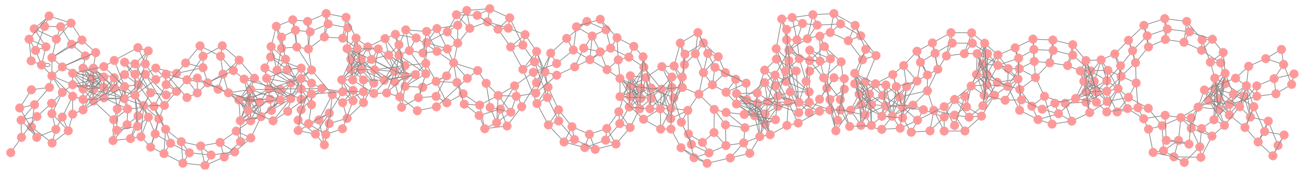
Figure 6. Ground MLN graph for 2910 frames, with 667 nodes, 2351 factors, and treewidth of 12. Nodes are ground predicates, and edges link nodes of ground predicates that appear in same formula.

ing which players are near the ball ($\texttt{obs\_ball\_near}(p, i)$, $\texttt{obs\_nearest\_ball}(p, i)$), and foot locations are useful for determining the location of players with respect to the court (e.g., $\texttt{obs\_nearest\_hoop}(p, i)$). Based on these observed relationships, events are hypothesized using a set of simple rules (see table 2). Although these rules are manually provided in our experiments, in general, hypothesized intervals can be obtained by conventional action recognition algorithms based on low-level image features [12].

For evaluation, ground truth is provided manually and includes locations of visible hands, feet, and ball every 100th frame, and start/end times of the events of interest. For hand, foot, and ball tracklets we first use the Hungarian algorithm to associate one ground truth location to one detected location, subject to some maximum distance, and then count the number of true/false positive and false negative matches. We use a threshold of $.1h$ for hands and feet (h is the height of the person), and $.5r$ for the ball (r is the ground truth radius of the ball). To evaluate events, we represent ground truth and detected events by intervals and use the Hungarian algorithm to match intervals to each other, minimizing the sum of the absolute difference between their start/end times. Matches are disallowed if the gap between two intervals is too long (60 frames/2 sec).

Table 3 shows the overall performance of the tracking module: feet and ball tracks are most accurate, since hands are subject to large amounts of self-occlusion and fast motion. Event generation is evaluated in table 4 by assuming all hypothesized event intervals are true; as expected, the recall ratio is high (.92), but some events are still missed, and a large number of false positives are present.

4.2. High-level inference results

High-level inference should be able to discard most false positives (increasing precision), but false negatives (missed intervals) are more problematic, since high-level inference only assigns truth values to hypothesized intervals; thus, final recall is strictly bounded by the recall of the event generation module. Table 5 shows the overall performance of our framework. As expected, most false positive hypotheses were removed (from 2302 to 279), but recall was reduced since some true positive hypotheses became false negatives after being labeled *false* by the MLN inference, likely due to observation errors or missing nearby hypotheses that are required by the axioms. Table 6 shows the final event recognition performance, given tracking and hypothesis module performance, in order to analyze the sensitivity of our final result to varying input performance. Performance is relatively stable, except for sequences 1 and 7, which have much better ball tracks, and thus have the highest F1 scores for event recognition; this is not surprising since the ball is the most important object in the game.

Our formulas relate only events that are nearby in time, leaving the long-term temporal relations, `before` and `after` in figure 2, to be implicitly enforced through properties, so the treewidth of the resulting ground MLN is relatively small (see figure 6), enabling exact inference on the whole video at once. The largest treewidth we encounter is 21, for a 25,287 frame sequence whose ground MLN contains 4,963 nodes and 18,440 factors, requiring 1.9 seconds for exact inference (not including tracking and network generation) on a 2.5 GHz Core 2 Quad CPU.

We also explore the effects of performing inference in an online streaming fashion by considering only observations and hypothesized events whose intervals end inside a sliding time window. Figure 7 shows our performance for varying window sizes. The performance approaches that of using the whole graph (table 5) once the window size is above 30sec (900 frames), implying that our approach will perform well even when we cannot solve the whole graph at once. Performance *does* degrade for smaller window sizes, implying that temporal relationships are valuable

	Hands	Feet	Ball	Hyp.	MLN
Sequence 1	.68	.88	.96	.55	.99
Sequence 2	.50	.84	.79	.41	.60
Sequence 3	.59	.82	.79	.47	.76
Sequence 4	.70	.82	.74	.40	.74
Sequence 5	.76	.88	.80	.40	.67
Sequence 6	.76	.91	.81	.43	.69
Sequence 7	.76	.89	.88	.41	.89

Table 6. F1 scores of tracking, hypothesis generation (**Hyp.**), and MLN inference (**MLN**)

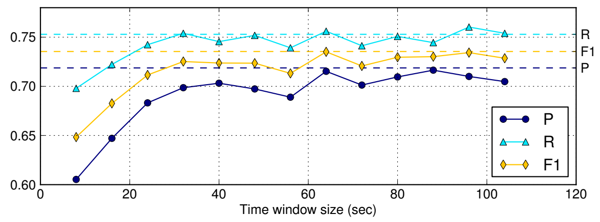

Figure 7. Performance of MLN inference with sliding windows. Dashed lines indicate full inference results from table 5.

in our problem.

5. Conclusion

We presented a framework which, given a semantic description of what generally happens in a scenario, uses video analysis and mixed probabilistic and logical inference to annotate the events that occurred. We demonstrated our approach on one-on-one basketball videos, recognizing complex events without additional cues such as text, camera movement, shot-changes, or overlaid time or score statistics. Because of the flexibility of the logical knowledge representation and relatively few restrictions on problem type (concurrent events are allowed, number of actors can vary, etc.), we believe that our framework can be extended to more difficult scenarios or other problem domains. Interesting future work includes using approaches such as Probabilistic Inductive Logic Programming (PILP) [18] to automatically learn additional rules given a background theory.

Acknowledgments

This research was funded by ONR surveillance grant N00014-09-10044. We also acknowledge Ryan Farrell, Aniruddha Kembhavi, and Brandyn White for their useful comments.

References

[1] J. F. Allen and G. Ferguson. Actions and Events in Interval Temporal Logic. *Journal of Logic and Computation*, 1994.

[2] M. Brand, N. Oliver, and A. Pentland. Coupled hidden Markov models for complex action recognition. In *CVPR*, 1997.

[3] P. Domingos, S. Kok, H. Poon, M. Richardson, and P. Singla. Unifying Logical and Statistical AI. In *AAAI*, 2006.

[4] P. F. Felzenszwalb and D. P. Huttenlocher. Pictorial structures for object recognition. *IJCV*, 2005.

[5] M. Ghallab. On Chronicles: Representation, On-line Recognition and Learning. In *KR*, 1996.

[6] T. F. Gonzalez. Clustering to minimize the maximum inter–cluster distance. In *Journal of Theoretical Computer Science*, 1985.

[7] A. Gupta, P. Srinivasan, J. Shi, and L. S. Davis. Understanding videos, constructing plots learning a visually grounded storyline model from annotated videos. In *CVPR*, 2009.

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

[9] S. S. Intille and A. F. Bobick. A framework for recognizing multi-agent action from visual evidence. In *AAAI*, 1999.

[10] K. Kim, T. H. Chalidabhongse, D. Harwood, and L. S. Davis. Real-time foreground-background segmentation using codebook model. *Real-Time Imaging*, 2005.

[11] S. Kok, P. Singla, M. Richardson, and P. Domingos. The Alchemy System for Statistical Relational AI. Technical report, Dept. of Computer Science and Engineering, University of Washington, Seattle, WA, 2005.

[12] Z. Lin, Z. Jiang, and L. S. Davis. Recognizing actions by shape-motion prototype trees. In *ICCV*, 2009.

[13] R. Marinescu and R. Dechter. AND/OR Branch-and-Bound search for combinatorial optimization in graphical models. *Artif. Intell.*, 2009.

[14] V. I. Morariu, D. Harwood, , and L. S. Davis. Tracking people's hands and feet using mixed network AND/OR search. *PAMI*, 2010, submitted.

[15] A. S. Ogale and Y. Aloimonos. A roadmap to the integration of early visual modules. *IJCV*, 2007.

[16] M. Perse, M. Kristan, S. Kovacic, G. Vuckovic, and J. Pers. A trajectory-based analysis of coordinated team activity in a basketball game. *CVIU*, 2009.

[17] L. R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Readings in speech recognition*, 1990.

[18] L. D. Raedt and K. Kersting. Probabilistic Inductive Logic Programming. In *ALT*, 2004.

[19] M. S. Ryoo and J. K. Aggarwal. Recognition of Composite Human Activities through Context-Free Grammar Based Representation. In *CVPR*, 2006.

[20] A. Sadilek and H. Kautz. Recognizing Multi-Agent Activities from GPS Data. In *AAAI*, 2010.

[21] Y. Shi, A. Bobick, and I. Essa. Learning Temporal Sequence Model from Partially Labeled Data. In *CVPR*, 2006.

[22] M. Siracusa and J.W.Fisher III. Tractable bayesian inference of time-series dependence structure. In *AISTATS*, 2009.

[23] M. Sridhar, A. G. Cohn, and D. C. Hogg. Unsupervised Learning of Event Classes from Video. In *AAAI*, 2010.

[24] V. thinh Vu, F. Bremond, and M. Thonnat. Automatic video interpretation: A novel algorithm for temporal scenario recognition. In *IJCAI*, 2003.

[25] S. D. Tran and L. S. Davis. Event Modeling and Recognition Using Markov Logic Networks. In *ECCV*, 2008.

[26] G. Zhu, Q. Huang, C. Xu, Y. Rui, S. Jiang, W. Gao, and H. Yao. Trajectory based event tactics analysis in broadcast sports video. In *MULTIMEDIA*, 2007.