Recognizing planned, multi-person action

Stephen S. Intille† and Aaron F. Bobick‡

†Massachusetts Institute of Technology
N51-340, 265 Massachusetts Avenue, Cambridge, MA 02139, U.S.A.
and
‡Georgia Institute of Technology
801 Atlantic Avenue, Atlanta, GA, Zip, 30332, U.S.A.

E-mail: intille@mit.edu; bobick@cc.gatech.edu

Received December, 1999; accepted October, 2000

Multi-person action recognition requires models of structured *interaction* between people and objects in the world. This paper demonstrates how highly structured, multi-person action can be recognized from noisy perceptual data using visually grounded goal-based primitives and low-order temporal relationships that are integrated in a probabilistic framework.

The representation, which is motivated by work in model-based object recognition and probabilistic plan recognition, makes four principal assumptions: (1) the goals of individual agents are natural atomic representational units for specifying the temporal relationships between agents engaged in group activities, (2) a high-level description of temporal structure of the action using a small set of low-order temporal and logical constraints is adequate for representing the relationships between the agent goals for highly structured, multi-agent action recognition, (3) Bayesian networks provide a suitable mechanism for integrating multiple sources of uncertain visual perceptual feature evidence, and (4) an automatically generated Bayesian network can be used to combine uncertain temporal information and compute the likelihood that a set of object trajectory data is a particular multi-agent action.

The recognition method is tested using a database of American football play descriptions and manually-acquired but noisy player trajectories. The strengths and limitations of the system are discussed and compared with other multi-agent recognition algorithms.

Key Words: multi-person action recognition; plan recognition; computer vision; Bayesian networks

1. INTRODUCTION

Computer systems that can reliably observe, recognize, and respond to *action* using visual sensors will enable new computational applications ranging from automatic video annotation and retrieval to interactive environments. Most computer vision research on the visual recognition of action, however, has focused on recognizing the action performed by a single person in isolation from other people and objects. Unfortunately, while these

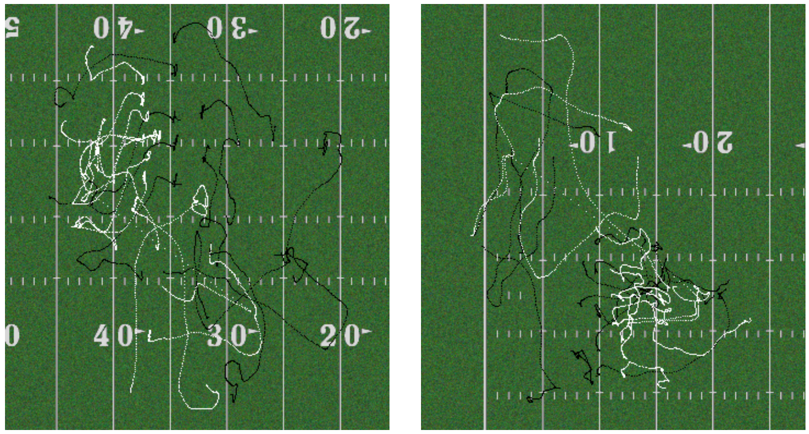


FIG. 1. Two examples of a p51curl play. The lighter trajectories are the offensive players. The data provided to the system consists of trajectories for all the objects including the ball, the approximate orientation of each object at each point along its trajectory, and a position label for each trajectory.

single-person scenarios are easy to construct in the laboratory, they are rarely found in the real world; we live in a world crowded with other people and objects.

This paper refers to any person or thing that moves as an *agent*. People are continuously interacting with other agents in their environments – often with more than one simultaneously – and changing their behavior in response. Application domains where everyday scenes with multiple people are being monitored (e.g. video annotation and automatic surveillance) will benefit from – and one might argue *require* – systems that recognize actions involving the interaction between multiple agents [29, 4].

This work describes a recognition algorithm for identifying well-structured, team-based, collaborative, multi-agent action. While this type of action constitutes only a subset of all multi-agent action, exploring this particular recognition problem identifies several general issues that any multi-agent recognition system must address. We describe the performance, strengths, limitations, and scope of the proposed representation.

Our example domain task is recognizing American football plays. The input for the test system consists of trajectory and upper-body orientation data for each object (i.e. players and the ball) moving in the scene. We construct a system that can identify a given set of trajectories as a particular play from a pre-existing database of play models provided by a domain expert – the football coach. Figure 1 shows 2 different observations of a play called the "p51curl," shown as "chalkboard" trajectory images. These images display the trajectories of all 22 players overlaid on the field. A play, which results from the collaborative interaction of the 11 offensive players, is a temporally-coordinated multi-agent plan where each agent is reacting to both the team goal and the movements of other agents. Using a database of plans, knowledge about football, computation of perceptual features from the trajectories, and propagation of uncertainty, the recognition system developed here can correctly determine that each of the examples shown in Figure 1 is in fact an example of a p51curl play.

The hypothesis of this work can be stated as follows: multi-agent collaborative actions comprised of action components can be represented and recognized from noisy perceptual data using visually-grounded, goal-based primitives probabilistically integrated by low-order temporal and logical relationships. Each collaborative multi-agent action is "compiled down" to a description of collaborative activity based on observation of *coordinated* action detectors that can recognize "what" but not "why."

The remaining sections of this paper are structured as follows. Section 2 discusses general properties of multi-agent action that impact the development of a representation for recognition from perceptual data. Section 3 and Section 4 describe how the representation presented here has been motivated by work in object recognition and plan recognition, respectively. Section 5 describes the football play database used in this work. The recognition algorithm is presented in Section 6. Finally, Section 7 evaluates recognition results for the algorithm and critiques the proposed representation.

2. MULTI-AGENT ACTION

This section examines general characteristics of the multi-agent action recognition task – in particular, the *collaborative* multi-agent action recognition task – that differ from single-agent action recognition task.

2.1. Characteristics of multi-agent action

Fundamentally, multi-agent action recognition differs from single-agent action recognition because the interaction *between* the agents in the environment is typically most characteristic of action. Consequently, popular single-agent recognition strategies cannot be directly applied to multi-agent recognition problems. Some general characteristics of all multi-agent action that differ from those of single-agent recognition are described below.

• *State space size.* Probabilistic state transition diagrams have worked well in practice for modeling a single-agent undergoing changes in state [45]. In multi-agent action, however, the state changes for each agent are not independent of the states of other agents. Therefore, if each object has N different states and there are M agents in the scene, at any given time there are N^M possible states. Consequently, a representation of the action using probabilistic state transition diagrams must use some additional information to reduce the number of possible transitions the system must consider at any given instance.

• *Feature space size.* Single-agent action representations for computer vision have generally used features like an agent's velocity, acceleration, and shape characteristics over time without considering other objects in an environment (e.g. see [48, 40]). The set of all possible perceptual features used for recognition is generally quite small (e.g. less than about 20 parameters). A system observing a multi-agent environment, however, can compute the *relative* states of agents over time and agents with respect to other groups of agents, leading to an explosion in feature space. A representation of action must therefore implicitly or explicitly use high-level contextual information to consider only those features most likely to be relevant to a particular recognition task.

• *Representing time.* When a scene consists of a single object, temporal rescaling techniques like dynamic time warping can be used to match input signals to models even when the input signal does not precisely temporally correspond to a particular model on the temporal axis [39]. Two people interacting in two parallel event streams, however, can result in a more complex temporal structure when there are temporal dependencies linking the two event streams. A representation that simply rescaled other measurements along the temporal axis may not be appropriate for model matching because temporal relationships are powerful matching features [33].

• *Explicit search and object assignment.* In multi-agent domains, an action model may have M agents and the dataset may have N agents, where $N > M$. Consequently, there are $\frac{N!}{(N-M)!}$ possible matchings between objects in the model and objects in the data. In the football domain, where $M = 11$ and (in this case for offensive players only) $N = 11$, there are $11!$ possible model-object to data pairings. Before a model can be compared with the data, a pre-processing step that searches for a "good" data-object to model-object match is required.

• *The complexity of agent interaction.* As more agents are added to a scene, the complexity of multi-agent and single-agent action will change as agents react to one another. A representation needs to model not just probabilistic variation, but probabilistic variation given context established by the interacting agents.

• *Defining temporal primitives.* Actions in the football domain are defined by relative temporal relationships as well as spatial relationships. For example, one player's role may be defined by a coach as follows: "The QB drops back and then throws the ball about the same time that the RFL makes a cutting motion. The RFL catches the ball after it is thrown." The primitives that the temporal detectors such as "before" are comparing must be defined as well as the semantics of the temporal detectors themselves.

• *Intentionality and the difficulty of physical modeling.* In domains with multiple agents, particularly when the agents are sentient and interacting in complex ways, modeling the action using *only* dynamic and kinematic models of physical interaction of agents is often computationally impossible and does not necessarily lead to improved recognition systems. For example, explanation of the movement of football players requires representing the pre-existing plans of the agents and the team. These plans will use the dynamic contexts established by the space-time interaction between objects. The agents are interacting in an *intentional* way and this high-level activity must be represented in the recognition model in some way.

These properties of multi-agent action make it difficult to apply single-agent action recognition methods directly to recognize multi-agent collaborative activity.

2.2. Characteristics of football team action

Using football as an example domain influences the representation that is developed in this work. The following properties of multi-agent action recognition are unique to this domain and do not hold for all multi-agent actions.

• *Infrequent collaborative re-planning.* Agents do adjust their individual plans during the play in reaction to the actions of other agents, but major plan changes accommodating some team goal are infrequent.

• *Agent based goal descriptions.* Football goal actions can be described using goal-like primitive components assigned to particular agents that are partially ordered in time and that obey certain logical constraints.

• *Existing taxonomies.* Coaches have developed semantically meaningful football annotation taxonomies that are used in this work.

• *Reset point.* In the football domain, the context can be initialized at the beginning of each play when the teams line up in their starting formations.

FIG. 2. (a) Two image frames from a video sequence of a football play. (b) Coach's chalkboard diagram for the simple-p51curl example play used in this paper. The play consists of 4 offensive agents (obj1 through obj4), 1 defensive agent (D), and a ball. Also indicated is the line-of-scrimmage (LOS) and some 5-yard marker yardlines.

- *Purposeful behavior.* In the football plays studied in this work, the players are engaged in highly directed behavior – nothing happens in a play without a reason.

These football domain properties simplify the multi-agent recognition problem.

2.3. The s51 football play example

Currently, all professional and many collegiate American football teams manually annotate video for analysis and automatic video retrieval. The football video used by the teams consists of a single camera view filmed from a centrally located press box near the top of a large stadium. The camera operator pans and zooms the camera to keep all of the players in the field of view for the majority of each play. Figure 2a shows two frames from video of a particular play.

Although offensive football plays consist of coordinated actions of 11 offensive players, for clarity a simplified play example will be used here when describing the proposed representation. This example is limited to just 5 players and the ball. Further, the number of actions performed by each agent has been reduced. The diagram for this example, the "simple-p51curl" is shown in Figure 2b and described in more detail in Appendix A.1.

3. OBJECT RECOGNITION IN SPACE-TIME

This work makes the following assumption: some multi-agent action can be identified by a system that can recognize "what things look like" instead of an intentionally based description of "what things are." In other words, recognizing team activity does not *always* require that a system represent and reason about the intentional communication and interaction of the agents being observed. Our strategy is to detect collaborative activity using a representation in which typical interaction has been "compiled down" into coordinated action. This representation is described in Section 6.

The model's tradeoff for computational efficiency is that it does not provide enough power to recognize some intentional collaborative behavior; it can only be used to detect typical coordinated behavior but cannot be used to compute an explanation for "why" some atypical but explainable collaborative behavior has been observed (e.g. explaining how players react to another player who has fallen).

The question becomes just how to "compile down" a complex intentional activity into visually-detectable components. Our approach is motivated by prior work in object recog-

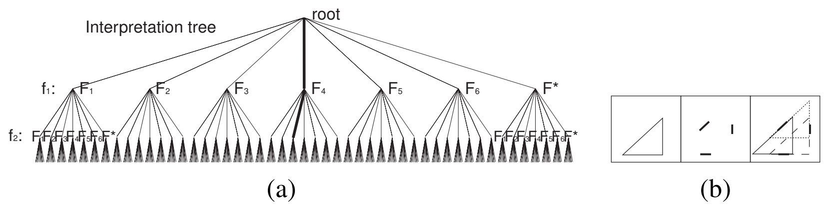


FIG. 3. (a) An illustration (adapted from Grimson[14]), showing the image interpretation tree used to search matches from model image segments to image feature segments. (b) Low-order consistency does not imply global correctness.

nition, which has been well-studied (see [14, 36] for surveys). Grimson has developed an instructive, general framework for analyzing the object recognition problem [14]. The task is to search for correspondence between the known model attributes, F_i (e.g. model edges and faces), and the features extracted from the scene image, f_i (e.g. edgels and region intensities), using some set of allowable transformations. Consider just a single model, $\mathcal{H}$, with attributes, $F_j(1 <= j <= 6)$ and a set of features, $f_i(1 <= i <= 25)$. The search space can then be described by an *interpretation tree* [14], illustrated in Figure 3a.[1]

The search proceeds as follows. At any node, F_j, f_i, which is on the ith layer of the tree, the children of the node are checked for consistency. First all applicable unary constraints are checked; f_{i+1} is compared with $F_1 ... F_6$. If these tests meet the matching criteria, then binary constraints in the current hypothesis path are checked. For example, if the current hypothesis is $\{f_1{:}F_6, f_2{:}F_6\}$ at level 2, then each child would be checked for binary consistency with $\{f_2{:}F_6\}$ and also trinary consistency with $\{f_1{:}F_*, f_2{:}F_6\}$. In this manner, each additional match to a hypothesis can consider new unary, binary, trinary, etc. relations. A poor match will eliminate a huge section of the search tree.

The goal, of course, is to find a good match. This approach finds a *consistent* hypothesis and assumes that consistency implies correctness. As developed in [15], the *order of the consistency* can be varied depending upon computational resources and accuracy requirements, but low-order consistency does not guarantee correctness. For example, Figure 3b, taken from [14], shows a triangle model and some image features. Considering only binary angular comparisons, any pair of features will match as consistent with the model. However, the model can only be put into correspondence with two features simultaneously, as shown in Figure 3b. All binary pairs being consistent does not imply that the match is correct. This appears problematic, because it is computationally intensive to check all n-ary relationships.

Grimson and Lozano-Pérez made a useful observation, however. Although it is mathematically *possible* for an incorrect interpretation to satisfy the all unary and binary relations, but not higher order relations, the probability of an object doing so falls precipitously as object complexity increases. This observation allows them to construct heuristic pruning methods that search for the correct interpretation by only maintaining unary and binary consistency [15].

[1] The $F*$ indicates no good match.

It is this idea, *that massive low order consistency typically implies correctness,* that drives the approach to recognizing complex actions presented in this work. The general principle of checking only unary and binary constraints will be applied to multi-agent team activity in the temporal domain. The principle of using low-order binary relationships will be used by limiting the temporal reasoning to detection of just a few temporal primitives between goals.

4. PRIOR WORK: MULTI-AGENT ACTION RECOGNITION

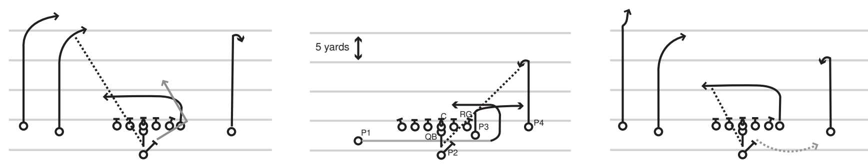
Prior multi-agent plan recognition work can be roughly divided into two methods. Some approaches explicitly represent group intentionality (e.g. [16]), typically using modal logics. Other approaches "compile down" intentional reasoning into procedural components, trading off the ability to reason about complex intentional interaction for computational tractability in domains with noisy evidence detectors.

A football play is a collaborative activity because players will adjust their individual goals in tandem in order to achieve a team goal (see [8] for a discussion differentiating collaboration and coordination). Each player is reasoning about both his own plan and the plans of other agents as he executes a play. Players will infer the plans of other agents and adjust their own plans accordingly; sometimes players will need to infer what another player is able to perceive in order to do so. Finally, at times players explicitly communicate information about their individual plans to other agents to facilitate the achievement of their shared plan of progressing the ball down the field in a particular way (e.g. the ball carrier pointing to the defensive player that he wants his teammate to block).

Formal theories of joint intentionality that explicitly reason about intentionality are difficult to tractably apply to the problem of recognizing collaborative action from noisy *visual* evidence; the methods presuppose the availability of perceptual detectors for cues of collaborative action that subsequently bias a logical inference process towards correct interpretations [22].

Bayesian networks, also known as belief networks, have been used for visual recognition of static objects (e.g. [27] and others) and for visual attention selection (e.g. [43]). Promising work on recognizing single-agent action from trajectory information using transition diagrams and fuzzy reasoning [30] led us to investigate the use of belief networks for multi-agent action recognition, which more explicitly represent knowledge dependencies and are computationally well-understood. Bayesian networks have been used to relax the strict assumptions of plan hierarchy models such as [25]. For example, networks can represent multiple top-level goals where probabilistic priors can be used to rank two equally possible but not equally likely plans [7]. Further, they have been used to integrate "action patterns" and beliefs about an agent's mental state [38].

Previous work in traffic understanding has used an agent-based belief network and agent-centered features for recognition of driving activity from simulated [11] and real data [6, 19]. Unlike that work our task requires that the system must also represent the logical and temporal relationships between multiple agents. Remagnino, Tan, and Baker [41] described a pedestrian and car tracking and surveillance system that models the interaction between any two agents using a small belief network. Dynamic belief networks (DBNs) and hidden Markov models (HMMs) have been used with some success but have not been demonstrated to be appropriate for domains in which multi-agent relationships result in large feature spaces and in which large and complete data sets for training are unavailable.



(a) (b) (c)

FIG. 4. Three New England Patriots' play diagrams for examples used in this work: (a) p52maxpin, (b) p51curl, and (c) p56yunder.

Although some logical backtracking search systems for recognizing multi-agent goals and actions have been proposed [42, 3, 46], noisy visual data requires a representation that can handle uncertainty. Pairwise comparison of features between trajectories can be used to recognize some group military behaviors for large numbers of agents [9]. Huber has shown that simple goal recognition belief networks can be constructed automatically from representations of action used for a plan generation system and then used by a planning agent in a multi-object scene [20]. Our approach builds on Huber's work of automatic construction of networks.

5. DATA AND DATA ACQUISITION

This section describes the football play data used to test the proposed recognition system. We assume the following information can be obtained from a domain expert (i.e. the coach):

1. The ideal paths of individual players.
2. Temporal relationships between the movements of some offensive players.
3. Space-time relationships between nearby offensive and defensive players.
4. The primary and secondary path of the ball.
5. Allowable variations in starting formation positions.
6. Optional moves available to particular players as the play progresses.

This information can be encoded in diagrams like the s51 diagram in Figure 2b.[2] Three play diagrams for some of the example plays used in this work are shown in Figure 4.

The input to the recognition system consists of trajectories of all moving objects given by (x,y,orientation,label) tuples as a function of the frame number, i.e. time. Here, *orientation* denotes the approximate upper-body orientation of a player and *label* is the name of the player's starting position. The player and the ball trajectories are provided.

A multi-object tracking system developed previously can track approximately half the players in a typical play from the football play database using contextual knowledge of "closed-worlds" [23]. Visual trackers, however, have difficulty reliably tracking players colliding in large groups or occluding one another.[3]

[2]Some information that is likely to be encoded in the original playbook is not used in this work such as specialized blocking responsibilities.

[3]Within a few years, electronic tags will be deployed in the National Football League, making it possible to obtain trajectories of all the players with 1 centimeter accuracy [47]. The recognition system described here is capable of using such data.

In this work, manually acquired trajectory data for all players on the field has been used.[4] Unfortunately, tracking a single 9 second play carefully requires several hours of tedious work using a special graphical interface. Currently 27 plays have been fully tracked. The two chalkboard images of a p51curl play in Figure 1 were obtained using this manual tracking system.

The trajectories are noisy due to four types of systematic errors. Placement errors occur when human trackers fail to accurately mark the position of points due to fatigue, leading to field positional errors larger than .5 meters in some instances. Since field points are also tracked and used to transform the trajectories to a field coordinate system, these errors propagate to all tracked objects. Center of mass errors result from the difficulty of estimating an object's center of mass as it projects onto the field.[5] The rapid motion of the players sometimes makes estimation of the feet/ground contact point difficult. When a player is temporally occluded by another player, the person tracking is instructed to smoothly estimate the position of the player until the person is visible again. Estimating upper body orientation is inherently imprecise. Often the torso faces one direction and the head faces another. In these cases, the person tracking will indicate some compromise orientation.

6. THE REPRESENTATION

In this section a representation for recognition of multi-agent action is developed motivated by insights from the object recognition and plan recognition literature. Results of using this representation on a database of football play trajectory data of the type introduced in Section 5 are discussed in Section 7.

6.1. Representational design

The goal of the representation presented in this section is to provide sufficient power to identify some "intentional" action by recognizing "what things look like" instead of reasoning about "what things are." The representation is based upon five assumptions:

1. *Low-order unary and binary temporal relationships are sufficient for recognition of some structured action.* Here only three temporal relationships are used: (1) whether an event A has been *observed* during some time, (2) whether an event A occurs *before* event B, and (3) whether an event A occurs *around* the same time as event B. This representational commitment is weak in comparison to existing temporal reasoning methods that use situation calculus [28, 34], temporal modal logics [44], or temporal interval propagation [2] to enforce complete temporal consistency.

2. *Visually-based goal detectors are useful atomic units in which to specify temporal and logical relationships.* Goals detected using visual cues are the "action components" of team activity. Powerful goal detectors can be created by integrating *local* spatial and temporal visual evidence using a Bayesian graphical network formalism.

3. *Bayesian networks, for reasons discussed in Section 4, are sufficient to integrate uncertainty resulting from noisy data and faulty detectors.*

4. *Local space-time modularity can be used to reduce complexity and simplify knowledge engineering.* Any complex problem that requires specification of domain-specific

[4] We feel that despite using manually-acquired data, the problems described in this work must be addressed by any computer vision system designed to recognize many types of high-level activity useful for video annotation applications, regardless of how the data was obtained.

[5] Only points that are in the plane of the field will be properly transformed to the field coordinate system.

rules by a person will need to allow the knowledge engineer to modularize concepts. Goal detection networks are designed to primarily consider evidence local in space and time.

5. *Deictic, or agent-centered, goal detectors can manage the complexity of multi-agent feature selection.* This assumption has been used successfully in prior work [1, 11, 6]. We use features such as "the closest agent" instead of "agent-5," so that some detectors can be used without an explicit search that matches data to objects.

The remainder of this section describes the details of how these representational assumptions are used to recognize multi-agent action.

6.2. Representational framework overview

The approach consists of the following representational elements:

1. *Single-agent goals* are the primitive action recognition atoms used for building up multi-agent play descriptions from visual evidence. Examples are *goal:catchpass(receiverPlayer)* and *goal:blockForQB(TE-player)*.

2. A *temporal structure description* of the global behavior, in this case a football play, is defined. The basic elements of this structure represent individual agent goals that must be detected. The relations coded in the structure are temporal and logical constraints to be verified.

3. For each basic element of the temporal structure a *visual network* that probabilistically detects the occurrence of the individual goal at a given time is defined. These networks encapsulate the knowledge required to address the individual decisions. The networks may refer to other such networks, but the more they do the less they decouple the individual goals. The evidence nodes of these graphs are direct perceptual sensors, such as the relative position of two players.

4. *Temporal analysis functions* are defined which evaluate the validity of a particular temporal relationship. For example, if there is a visual network detecting a quarterback throwing the ball and another detecting a receiver catching it, the temporal analysis functions can evaluate whether one goal happened *before* the other.

5. A multi-agent *temporal goal belief network* is automatically constructed for each group activity to be recognized that reflects the temporal and logical constraints of the temporal structure description. The nodes are the individual goals and the specified temporal relations. The links enforce conditional probabilities and associated logical relations such as *xor*.

Figure 5 shows an overview diagram of the system. Assume the data consists of trajectory information for each player from frame 0 to frame N obtained from video input where each player has been tracked. A domain expert (i.e. a coach) has provided prototypical temporal structure descriptions of all known team plays. The system is designed to operate causally, computing, at each time, the likelihood the observed data is a given play in the play database. At each time, visual networks are applied to each object in the trajectory data and output likelihoods of observing agent goals such as *dropback (QB)* and *throwPass (RSE)*. Over the course of the play, those networks produce likelihood curves as a function of time. Temporal analysis functions use the visual network likelihood curves and heuristic functions to compute new likelihood curves detecting temporal relationships between agent goals (e.g. dropback (QB) is observed *before* throwPass (RSE)). At initialization of the algorithm, each play's temporal structure description is automatically converted into

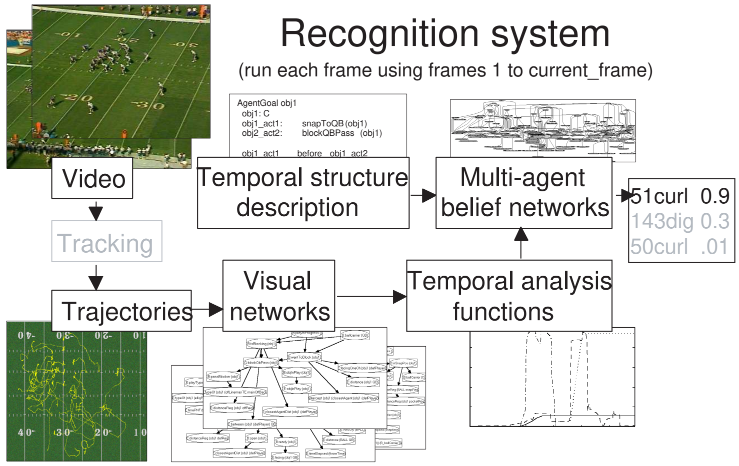


FIG. 5. The recognition system diagram.

a multi-agent belief network used to compute the likelihood of observed data being that play. The multi-agent belief network uses evidence from the temporal analysis function likelihood curves to compute the likelihood of having observed a particular play at time i. Likelihoods are computed for each model and rank-ordered. The maximum likelihood play model indicates the system's best play recognition hypothesis at frame i.

The remainder of this section describes the representational components in more detail.

6.3. The temporal structure description

The temporal structure description is the format in which a domain expert (e.g. a coach) encodes typical examples of team activity. It represents the prototypical scenario of the described action and is comprised of fundamental behavior elements connected by temporal and logical constraints. In designing such a structure the assumption is made that the structured actions being represented have such a prototype. Furthermore, since the description needs to be generated manually, the description needs to be simple to express.

6.3.1. Agent-based goals as atoms

Within the current framework, individual agent *goals* are used as the basis for the descriptive structure. Actions are viewed as a partially ordered set of goal directed behaviors on the part of interacting agents. Agent-based goals are selected as atomic units because they are a natural and convenient way to describe ongoing human activity. Furthermore, when an agent is situated in the midst of other agents engaged in some activity, the main goal of each agent will usually have visual consequences.

Goals are *defined* by their visual and probabilistic characteristic behaviors, building on work in probabilistic plan recognition [7]. The perceptual evidence is evaluated to determine whether an agent has a particular goal at a particular time. For example, player A can have the goal of running between players B and C. To determine if indeed he has such a goal, a recognition system must evaluate the visual evidence, particularly the positions of B and C and the direction of motion of A.

6.3.2. Criteria for selecting goals

Unlike methods based upon computational simulation of intentional communication [35, 16], here there is no principled basis by which to define what can constitute a individual agent goal. However, several desirable heuristic design criteria are used when selecting individual goals for modeling. First, if a goal or behavior is causally related to other goals (e.g. catchPass and throwPass) the temporal relationships are easy to define. Second, the goals need to have visually-detectable components. For example, a goal to "win the game" may have many subgoals that have visual components, but it is difficult to define visual cues associated directly with the goal "win the game" itself. Third, if the behavior involves a single primary agent then detectors for these behaviors are easier to construct, and the application of these detectors is easier to control.

6.3.3. Linking goals with constraints

Three temporal relationships are defined for the temporal description: observed, before, and around. Given two agent goals, A and B, the following temporal detectors are available in this framework for describing activity: whether A (and B) has been *observed*[6], whether A is *before* B, whether B is *before* A, and whether A is *around* B (and therefore, since around will be implemented symmetrically[7], B is *around* A). By assumption, the goals of an agent are active during temporal intervals of finite duration. The detectors therefore compute the relationships between goals that extend for intervals of time.

The justification for using only low-order temporal relationships is discussed in Section 4. Systems that reason about the temporal relationships between intervals typically permit Allen's 7 possible temporal relations, not counting inverses: *before*, *meet*, *during*, *start*, *finish*, *equals*, and *overlaps* [2]. Allen's detectors require temporal precision – single-frame start and end points. In this work, the intervals are defined by likelihood curves computed using Bayesian networks and noisy evidence. The curves rarely have sharp activation and deactivation transitions. Some of this imprecision is due to noise, but much of it results from actions not having clear start and stop points (e.g. precisely when a cutting or throwing action begins and ends is not clear).

The *before* detector is common to both Allen's framework and this work. The remaining relations are ill-defined when the start and end points of temporal intervals are uncertain. For example, in the framework presented here the temporal relation of simultaneity is expressed as *around* which can be interpreted as "about the same time as" or as a combination of Allen's *during* and *overlaps* relations. Without explicit interval borders, the relation is not transitive and cannot support the hard chaining used by Allen.

Due to this imprecision, transitive closure is *not* applied to the temporal relations. Rather, only relations manually constructed by the knowledge engineer designing the action description are exploited. The only temporal relations in the temporal structure graph are the *before* and *around* relations explicitly coded in the temporal structure description. Further, only *observed*, *before*, and *around* relationships are used for action recognition due to the ill-defined boundaries of the intervals being temporally compared.

[6] *Observed* is a unary temporal relationships that sums up evidence for a single goal over a window of time.

[7] Since an *around* detector could can take into account the percentage of an interval that overlaps with another interval, it is possible to generate a non-symmetric around detector for cases where interval A is shorter than interval B and a occurs entirely *during* B.

```
(goalTeam s51 "Team goal for simple-p51curl (s51) play."

  (agentGoal obj1 (agent (obj1 (C)))   ; Obj1 is always the Center (C)
     (goal obj1_act1 "snapToQB (obj1)")
     (goal obj2_act2 "blockQBPass (obj1)")
     (before obj1_act1 obj1_act2))

  (agentGoal obj2 (agent (obj2 (QB)))  ;Obj2 is always the Quarterback (QB)
     (goal obj1_act1 "dropback (obj2 5)")
     (goal obj2_act2 "throwPass (obj2)")
     (before obj2_act1 obj2_act2))

  (agentGoal obj3                      ;The Right Wing Back (RWB)
     (agent (obj3 (RWB RTE RHB HB FB TB LWB LSB)))
     (goal obj3_act1 "passPatStreaking (obj3 4 45 defReg nearRightSidelineReg 0)")
     (goal obj3_act2 "passPatCutting (obj3 70 offSidelineR-
ightReg freeBlockingZoneReg)")
     (goal obj3_act3 "runBehind (obj3 obj4)")
     (goal obj3_act4 "passPatParaLos (obj3 3 defReg offSidelineRightReg 4)")
     (goal obj3_act5 "catchPass (obj3)")
     (before obj3_act1 obj3_act2)
     (before obj3_act2 obj3_act4))

  (agentGoal obj4                      ;The Right Flanker (RFL)
     (agent (obj4 (RFL RWB RSB LFL LSB LWB)))
     (goal obj4_act1 "passPatStreaking (obj4 4 50 defReg offEndZoneReg 0)")
     (goal obj4_act2 "passPatCutting (obj4 70 offSidelineLeft-
Reg freeBlockingZoneReg)")
     (goal obj4_act3 "passPatParaLos (obj4 3 defReg offCenterLineReg 4)")
     (goal obj4_act4 "catchPass (obj4)") (before obj4_act1 obj4_act2)
     (before obj4_act2 obj4_act3))

  (around obj3_act2 obj4_act2) (xor obj3_act5 obj4_act4))
```

FIG. 6. A temporal structure description for the *s51* play example with only some actions and temporal relationships specified.

Two logical relationships can be used in the temporal structure description: *xor* and *or*. The semantics are standard. Implicit in the temporal structure description is the logical relationship *and* because all the elements listed in a play are presumed to happen unless overridden by an explicit *xor* or *or*. These logical relations will be converted into biases on conditional probabilities when the issue of uncertainty is addressed.

The small set of logical relationships will permit nested logical options to be modeled. However, as discussed in Section 2, modeling some intentional team action can require reasoning about agent interaction where single agents make decisions that cause other agents to change their plans. This type of plan recognition requires a reasoning process that backward chains logical decisions; the framework proposed here cannot model that type of plan.

6.3.4. Example

Figure 6 shows the temporal description for the s51 example play, diagrammed in Figure 2b. The description contains four agents: obj1, obj2, obj3, and obj4. Each agent has an *agentGoal* definition. The first slot in the definition, called *agent*, is a rank-ordered list of player positions that can possibly map to the object definition.[8]

Following the agent slot, one or more goal slots are specified. The example indicates that in an s51 play, obj1 (which will match the trajectory labeled C), should have a goal to

[8]This slot is necessary because the same play definition can apply to multiple starting position configurations. Space constraints limit further discussion here; see [22].

snapToQB (snap – hand the ball to the quarterback) and blockQBPass (block for the QB as the QB passes the ball). Each goal has a label, such as obj1_act1 (short for object1's action1). The s51 example has been limited to just six goal types: snapToQB, blockQB-Pass, passPatStreaking, passPatCutting, passPatParaLos, catchPass. For each goal type, there exists a detector which receives a list of parameters. The detectors are implemented as Bayesian networks that integrate direct visual measurements. Two of these detectors and their parameters are described below:

- *blockQBPass (obj) Obj* blocks for the QB as the QB tries to pass the ball.
- *passPatCutting (obj angle toReg inReg) Obj*, which must be an eligible receiver, runs a pass pattern segment making a sharp (e.g. about *angle* degrees) change in motion in *inReg* after which *obj* is moving in towards the *toReg*.

The remaining slots in the temporal structure description indicate the temporal and logical relationships between agent goals. Two temporal primitives are available: *before* and *around*. For example, "(before obj1_act1 obj1_act2)" indicates that goal obj1_act1 occurs before obj1_act2, where obj1_act1 is the label for "snapToQB (obj1)" and obj2_act2 is the label for "blockQBPass (obj1)". Similarly, "(xor obj3_act5 obj4_act4)" indicates that object3's catchPass goal or object4's catchPass goal should be observed, but not both.

6.3.5. Compound goals

A *compound goal* (cgoal) is simply a goal comprised of multiple primitive goals. Temporal and logical relations can exist between the cgoal and other primitive goals, as well as between the primitive goals of the cgoal itself.

Since cgoals could be expanded by their definition in terms of their primitive goals, one might wonder why use them at all. The main reason is that cgoals represent causally linked individual goals that are tightly coupled. Therefore, it is possible to specify temporal and logical relationships between cgoals. Within each cgoal, subgoals are listed with appropriate temporal relationships specified. However, a logical constraint is also specified between the two cgoals (e.g. *xor*) indicating that either all the sub-goals of one cgoal are observed or all the subgoals of the other cgoal are observed, but not both.

6.4. Perceptual features

The input to the system is the labeled player trajectory data. Some of the features are computed directly from the trajectories of an object or group of objects without directly comparing attributes of multiple objects. These detectors can be roughly grouped into the following categories:

- *Movement of objects* (e.g. *velocity(offensePlayers)*).
- *Trajectory curvature* (e.g. *curvatureTraj(obj)*).
- *Categorization and visibility* (e.g. *typeOf(obj,offense)*).

Although information computed directly from the agent trajectories can be used by the system, in multi-agent domains the most salient features are sometimes the relationships *between* the trajectories of individual and groups of agents and other agents or regions in the scene. These detectors can be roughly grouped into the following categories:

- *Distances between objects* (e.g. closestAgent(obj,defenseObjs)).
- *Relative orientation of objects* (e.g. *facingOneOf(obj1,defense)*).

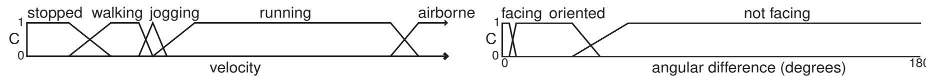


FIG. 7. Two examples of the piecewise-linear functions used to compute a degree of membership value by perceptual detector functions computing object motion and object-to-object facing angle.

- *Measurement of change* (e.g. *compareDistanceMoved (obj timePT distance)* checks if *obj* has moved *distance* since *timePT*).
- *trajectory properties* (e.g. *enteredReg(obj decTime region)* checks if *obj* entered *region* within *decTime*)).
- *spatial relationships between objects* (e.g. *behind(obj,linemen)*).
- *spatial relationships during intervals of time* (e.g. *closestApproach (obj, QB, decTime)*).
- *path interception estimation* (e.g. *intercept (obj1 obj2 incTime)*).
- *Temporal extent checks* (e.g. *timeElapsed(throwTime)*).

Each feature detector takes a set of arguments, typically groups of objects and/or regions. The detectors can be applied at any time and are causal, only using data from frame 0 to (just after) the current time.

Each detector quantizes its output into a small set of output states. For example, the states for the "velocity" (`velocity (objs)`) detector are {stopped walking jogging running airborne}; the states for the "facing each other" detector (`facing (objs1 objs2)`) are {facing oriented notFacing}. The output states are used for convenience when simplifying the specification of conditional probabilities in the visual goal networks, to be described in Section 6.5. To avoid hard cutoffs in continuous-valued concepts (e.g. velocity, distance), the functions actually compute a *degree of membership* in each state [49] using overlapping piecewise linear functions.[9] Figure 7 shows the piecewise discretization functions for the *velocity* and *facing* evidence functions. A value of 1 indicates that the detector output is definitely a member of the state and, conversely, a value of 0 indicates the detector output is not a member of the state. In this work, these functions are manually specified by the domain expert.[10] The interpretation of the membership curve values as relative likelihoods and how they are used by the visual networks is described next.

6.5. Visual networks

Previous work, discussed in Section 4, demonstrates that agent goals can be represented in a probabilistic framework using Bayesian belief networks [7, 21, 38]. The Bayesian networks use the output of the visual feature detectors, so they will be referred to as *visual networks*. The networks are used as building blocks for recognizing multi-agent activity

[9] The degree of memberships are interpreted as probabilities. This is just one of the semantic modeling tradeoffs required to make Bayesian networks work in a large, practical system.

[10] With a large dataset of recorded actions, where the states have been manually labeled by multiple users, it would be possible to estimate degree of membership curves. In this work, such a labeled dataset could not be obtained.

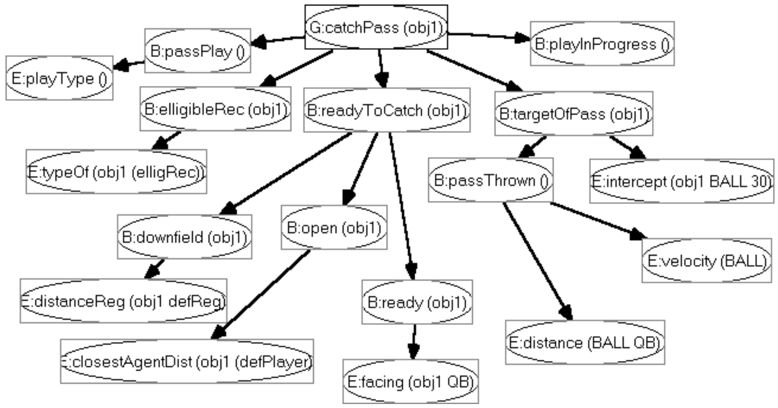


FIG. 8. The *catchPass* goal network.

by using their outputs to check for temporal consistencies (e.g. goal A is observed *before* goal B).

6.5.1. Network structure and evaluation

A single belief network is used to detect each goal in this framework. Each network can be instantiated at any time during a play in order to compute the likelihood the goal has been observed at the given time. The networks typically contain only between 15 and 25 nodes because they are designed to integrate information in a relatively small, *local* spatial-temporal window.

The random variables (i.e. nodes) and structure of each network are manually specified. The networks are generally sparsely connected, having only two or three connections per node. Sparse connectivity is possible because the knowledge engineer hierarchically clusters concepts within the networks. Propagation of uncertainty in sparsely connected networks with less than a about 100 nodes can be performed using exact probabilistic propagation techniques [24] in interactive times (i.e. < 1 second of compute time on a 500 MHz Digital AlphaStation). Currently the priors are also manually assigned, however some priors could be obtained from analyzing the evidence and the performance of particular feature detectors.

Figure 8 shows the *catchPass* network. Although this network is singly-connected, other visual networks are not. However, the knowledge engineer must use standard modeling tricks to avoid cycles in the network so that fast belief propagation algorithms can be employed [24].[11] Each network consists of two types of nodes: unobservable belief and observable evidence.

- *Unobservable belief nodes.* A belief node has two states, *true* and *false* (each with a corresponding probability value), and represents an internal state of the agent or some

[11] In graph theory literature, it is common to indicate dependence using arrows such at if A is dependent on B then B has a link pointing to A. However, when the networks are used for belief propagation, Bayesian propagation rules are maintained - information propagates in both directions.

external state in the world at the time when the network is evaluated. An example is the *B:readyToCatch* node.

- *Observable evidence nodes.* An evidence node's states and state values are directly dependent upon the data. Some nodes are binary (e.g. *observed*, *notObserved*), most are trinary, (e.g. *observed*, *maybeObserved*, *notObserved*), and the remainder have specialized states that quantize a particular feature detector output (e.g. the result of the *distance* detector is quantized into states *inContact*, *nextTo*, *near*, *inVicinity*, *far*, *distant*). When the data is inconclusive, the function can return a *NIL* value, and the evidence node will not be instantiated with any value in the network.

Evidence nodes, which use the feature functions described in the previous section, can take objects, groups of objects, object categories, regions, or times. For instance, the *distanceReg ((offEnd) LOSReg)* node in the *playInProgress* network will compute the distance from any offensive end in the play (of which there are several possible types) to the line-of-scrimmage (LOS) region. Most evidence functions will work with groups of agents. *velocity (LT RT LG)* will return the average velocity of the three objects. Some regions and functions (e.g. *LOSReg* and *snapTime()*) are dependent upon variables that have been previously set by other networks.

By nesting functions, references to most agents in the local spatial-temporal window can be made from within a network. For example, to compute the distance from the agent to the closest defensive agent, the detector *distance (closestAgent (defense))* is applied.

References to other agents are generally deictic – relative to the position of the current agent – (e.g. closest agent, second closest agent, agents within some distance, etc.) Using deictic feature references reduces the number of features that need to be considered by a network; the networks primarily include features computed locally in space and time.

The main node of each visual network can accept parameters set by the caller of the network at run-time. For example, goal node *catchPass (obj1)* accepts one argument, a specific agent. Each network is designed so that it can be applied to any world object and return a reasonable result. Child nodes of the main node inherit the top-level parameters and can refer to additional objects. In the *catchPass* example, if the network is evaluated with *obj1* bound to *WR*, then the *E:intercept (obj1 BALL 30)* evidence node will return either *observed*, *maybeObserved*, *notObserved*, or *NIL* depending on whether it appears that the ball's current estimated trajectory will intercept with the *WR*'s estimated trajectory within 30 frames.

6.5.2. Approximations to the closed-world principle

The task for the knowledge engineer is to estimate the probabilistic causality between an agent's particular goal leading to the agent having a specific set of beliefs which will lead to *visual* evidence of particular actions. In doing so, a small space of the domain's knowledge space is carved out and within that subset of knowledge rich interdependencies between the data and a goal can be modeled. By assumption in Bayesian networks, dependencies between evidence and beliefs for a particular goal are either fully modeled or not considered at all. The networks are intended to be closed in time and knowledge – by considering information local to the given time and given agent position, the amount of domain knowledge that must be considered remains manageable.

Networks are designed to use primarily evidence detected from a local space-time window. Note, however, that some goal networks make use of dynamic state variables (e.g.

throwTime used in the *catchPass* network of Figure 8), and some networks use the output of other goal networks as evidence (e.g. *catchPass* uses the result of the *playInProgress* network). Therefore, the networks are not entirely "closed." External knowledge can impact the network, which violates the belief network assumption that all dependencies are modeled via explicit conditional probabilities. This approximation is acceptable because the networks themselves are simplified approximations to the actual dependency structure: partitioning actions into small networks simplifies the job of the knowledge engineer.

The evidence from the output of the *playInProgress* visual network is entered into the *catchPass* network in Figure 8 as continuous values using "virtual" likelihood evidence (see [32]). This technique permits continuous valued information to be entered into evidence nodes with discrete states. The likelihood is obtained using the degree of membership values obtained for each of the node's states (see Section 6.4 and Figure 7).

In practice, nested visual networks were required to manage the complexity of knowledge engineering the visual networks for the play recognition system. Implementing the football-specific detectors without use of smaller, generic network components that could be inserted as evidence proved impractical. Using virtual evidence insertion in this way assumes, incorrectly, that all variables in the generic network are independent of the variables used in the football-specific network. This is just one of the many "non-Bayesian" assumptions being made by the system in order to allow practical construction of the system.

Some agent goals have an additional property: "achievement" of the goals can set global states. A goal is achieved when its likelihood value surpasses a fixed threshold. A *throwPass* detector, for example, will return a high likelihood value (i.e. close to 1.0) once the ball and QB player have undergone appropriate motion in the appropriate context and the ball has left the QB player at a high speed. At this time, the threshold value is typically surpassed. Once a *throwPass* is executed, some goals are no longer likely (e.g. *tackle(QB)*), so some achieved events set global state indicators that other visual networks can use as evidence.[12]

6.6. Temporal analysis functions

Temporal analysis functions are functions which evaluate the validity of a particular temporal relationship. For example, if there is a visual network detecting a quarterback throwing the ball and another detecting a receiver catching it, the temporal analysis functions determine whether one event happened *before* the other.

The output of a visual goal network at each frame for a given object results in a likelihood curve over time. Two curves returned by the networks "dropback (QB 5)" and "catchPass (RSE)" are shown in Figure 9. Temporal relationship evidence detectors use these curves as input. The functions compute a certainty value for the *observed*, *before*, and *around* tests at each time frame using heuristic functions that compare the activation levels of each goal over time, the characteristics of each input curve, the temporal distance between features of the curves, the amount of overlap between the curves, and a minimal activation time for each goal (see Appendix A.2). Figure 9 shows the certainty values for the *before*

[12] One performance problem is the potential for circularity in the visual goal networks. Network A uses the output from network B, network B uses the output of network C, but network C uses the output of A. Here loops are handled by having the evidence computation algorithm avoid recomputing any networks that have already been computed for the current time.

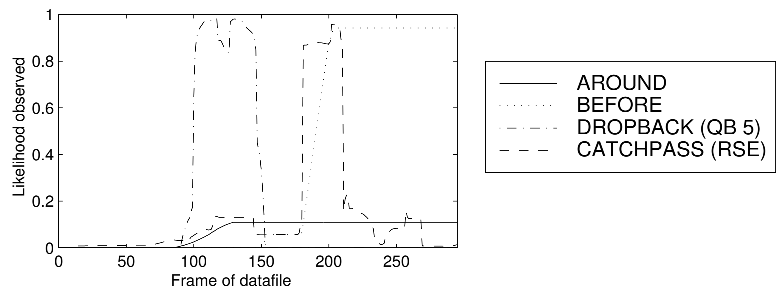


FIG. 9. Goal likelihood curves returned by the networks "dropback (QB 5)" and "catchPass (RSE)" superimposed with the corresponding temporal curves for "dropback (QB 5) before catchPass (RSE)" and "dropback (QB 5) around catchPass (RSE)".

and *around* detectors corresponding to "dropback (QB 5) before catchPass (RSE)" and "dropback (QB 5) around catchPass (RSE)".

The semantics of (*before* A B), where B is a compound goal are interpreted as A is *before all* the subgoals of B. Similarly, for *around*, A is *around* all the subgoals of B. In practice, if compound goal B consists of subgoals $\{B_0, \ldots, B_i\}$ then the certainty value for B at each time t is simply $argmax(B_0(t), \ldots, B_i(t))$. This new certainty curve is then compared with the curve for A using the heuristic functions.

6.7. Multi-agent networks

The fundamental hypothesis of this work is that visual networks and the temporal structure description can, when integrated so as to propagate uncertainty, be used to recognize structured team goals in some interesting multi-agent domains without explicit reasoning about group intentionality and communication. This integration is accomplished by an extension of visual networks to *multi-agent networks*. Each multi-agent network represents a "compiled-down" description of the collaborative action of a particular play.

6.7.1. Multi-agent network structure

Figure 10 shows an example fragment of a multi-agent network that recognizes *part* of the action description shown in Figure 6 for the s51 play example.

All nodes in the multi-agent networks represent beliefs or evidence observed over all the play data seen from the start of the play until the current time. All nodes have the state $(observed, notObserved)$. The main node in the example is *B:s51 (obj1 obj2 obj3 obj4)*. To apply this network, four arguments matching the objects in the action description to objects in the data are required. Below this node are nodes representing:

- *Compound goals* (e.g. *B:s51 (obj1)*). A compound goal represents several related subgoals (or temporal relations between subgoals). In the network, each component of a compound goal is causally related to some compound goal node. Each agent in the scene has one top compound goal node that is conditioned on the main play node.

- *Binary temporal relationships between goals* (e.g. *B:obj1_act1 before obj1_act2*), which represents the belief that *snapToQB (obj1)* is *before blockQBPass (obj1)*). These nodes encode the belief that a particular temporal ordering of agent goals has been ob-

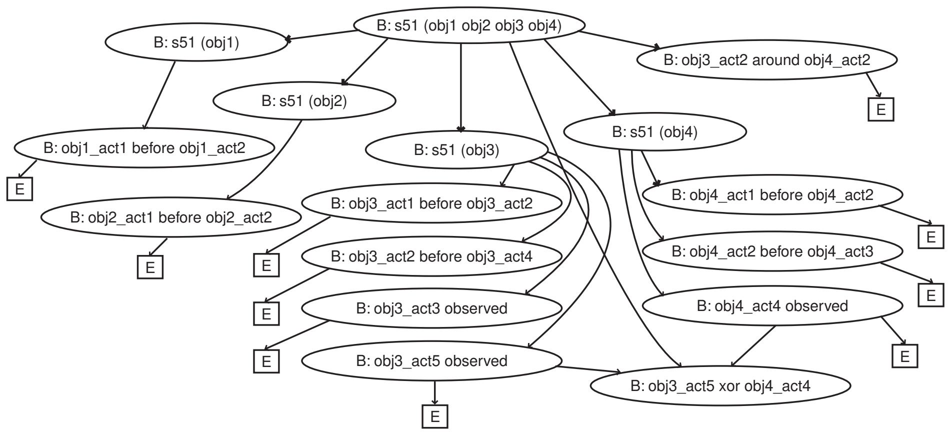


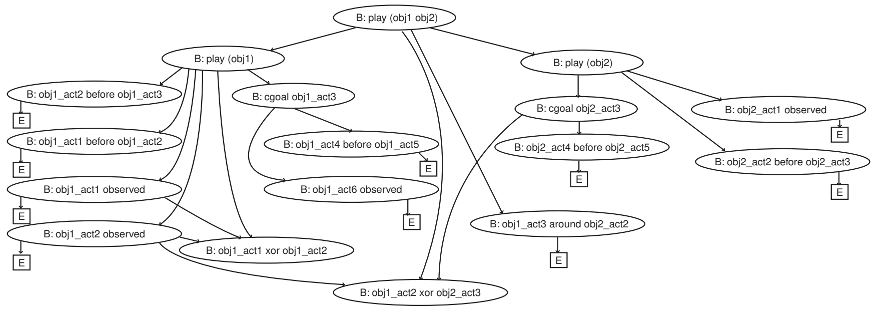
FIG. 10. The multi-agent network structure for the s51 play temporal structure description in Figure 6.

served during the action sequence. They are assumed to be causally related to a parent compound goal node.

- *Observation of goals* (e.g. *B:obj4_act4 observed*). When the temporal structure description does not specify any temporal relationship for a goal (e.g. obj4_act4 in Figure 6), the goal is inserted into the network using an observation node. The node is assumed to be causally related to a parent compound goal node.
- *Evidence for temporal relationships between goals or observations of goals* (e.g. the boxed E nodes). By tuning the conditional probability table, $P(evidence|belief)$, sensor uncertainty is modeled.
- *Logical relationships between goals or compound goals* (e.g. *B:obj3_act5 xor obj4_act4*). The two logical relations, *XOR* and *OR* are represented using special nodes. For each operand of the relation, an observed node is included in the graph. The logical relationship node is then a child of the observed nodes and the first compound goal node common to the goal actions being compared; in the example, this node is the main goal node (because the xor relationship spans two agents).

Figure 11 shows an example network with the corresponding play description. This example demonstrates some additional multi-agent network properties. If a goal does not have corresponding temporal relationships, it is inserted into the network using an observed relation (e.g. obj2_act1). Further, any goal with a logical constraint is inserted into the network as an observed relation, even if a temporal relationship is already in the graph (e.g. obj1_act2 has both a temporal relation node, “before obj1_act2 obj1_act3” and an observed node). Logical relationships between compound goal nodes are treated exactly as relations for single goal nodes (e.g. “xor obj1_act2 obj2_act3”). Finally, temporal relationships between a compound goal node and single goal node are implemented as shown by the “before obj2_act1 obj2_act2” node and “around obj1_act3 obj2_act2” node.

There are two important observations. The first is that the graph node and linking structure is clearly an approximation to the true dependencies in the world. For example, the node “B:obj1_act3 around obj2_act2” is certainly causally related to “B:cgoal obj1_act3”, but this dependency is not modeled in the graph in order to reduce the number of links




```
(goalTeam play "Play example"

    (agentGoal obj1 (agent (obj1 (C)))
        (goal obj1_act1)
        (goal obj1_act2)
        (cgoal obj1_act3
            (goal obj1_act4)
            (goal obj1_act5)
            (goal obj1_act6)
            (before obj1_act4 obj1_act5)
        (before obj1_act2 obj1_act3))
        (xor obj1_act1 obj1_act2))

    (agentGoal obj2 (agent (obj2 (LT)))
        (goal obj2_act1)
        (goal obj2_act2)
        (cgoal obj2_act3
            (goal obj2_act4)
            (goal obj2_act5)
            (before obj2_act4 obj2_act5))
        (before obj2_act2 obj2_act3))

    (around obj1_act3 obj2_act2)
    (xor obj1_act2 obj2_act3))
```

FIG. 11. A temporal structure description and the multi-agent network structure generated.

required in the graph for a typical multi-agent action. The second observation is that the graphs are using all observed evidence from frame 0 to the currentTime. Therefore, the integration of evidence over time is encoded within the nodes of the network, not within transitions between the links. These two observations will be discussed further in Section 7.

The network shown in Figure 10 is for the simple s51 example, which has only a few agents with a reduced set of goals and temporal relations. The temporal structure descriptions normally describe the actions of all 11 offensive agents with about 5 goals per agent, 5 temporal and logical goals per agent, and 3 between-agent temporal/logical goals. Hence, a typical multi-agent network contains about 188 nodes (1 main node, 11 compound player nodes, 55 temporal relation nodes, 55 temporal relation evidence nodes, 33 between-agent temporal relation nodes, and 33 between-agent temporal evidence nodes). Propagation by exact algorithms in interactive times remains feasible because the networks are relatively shallow, sparsely connected, and consist of binary nodes.

6.7.2. Generating networks automatically

Multi-agent networks are generated directly from the temporal structure descriptions provided by the domain expert given a set of construction rules and conditional probability templates. The rules specify how to insert and connect nodes for a temporal structure description so that the node-link structure described in the previous section are obtained. Next, conditional and prior probabilities for the network are inserted automatically using node-link structure templates that match to specific situations and apply pre-designed tables. This process of template-based insertion of probabilities is similar to the method used by Huber [20], although with networks of a different structure.

The automatic structure creation and probability assignment is possible in practice only because the networks are limited to just 6 node types (before, around, observed, cgoal, evidence, and main play) and a small set of possible links (e.g. evidence nodes will always only have one incoming link from either a before, around, or observed node). Therefore, the priors and conditional probabilities for each possible network structure can be manually estimated by the knowledge engineer.

Others have used the idea of automatic construction of networks using rule-based expert systems [18, 31], first-order logic [13], Horn clauses with associated conditional probability tables [5], and sensitivity analysis pruning [5, 37, 17]. All these methods rely on pre-specified conditional probability tables and (to make the techniques practical) a relatively small set of primitives that can be combined.

6.8. Applying the representation

6.8.1. *Trajectory to model consistency matching*

So far, this section has described how the system can compute a likelihood that a particular play is being observed. However, the description has assumed that the matching between object trajectories and objects in the temporal structure description is known. In the worst case, given n objects and n trajectories, there are $n!$ possible pairings. However, the football domain can be used to limit valid combinations. The trajectories in the football play dataset are labeled using standard football position notations (e.g. QB, C, RFL). These labels provide information about the role of the object for the given trajectory. Given such labels, the number of valid object to trajectory pairings can be reduced by specifying preference information in the temporal structure description. In the example described in Figure 6, the *agent* slot of the *agentGoal obj3* description indicates that object *obj3* can possibly match with a trajectory if the trajectory has the label "RWB" In reality, this slot has an ordered list of labels: *(RWB RTE RHB HB FB TB LWB LSB)*. *obj3* will most often be the RWB, then the RTE, and so on. Object *obj2* in the example always matches with the trajectory labeled QB since there is a QB in every play. Given the preference orders, a *consistent* assignment of trajectory data to the object description must be made. Here, a rule-based algorithm finds the single most consistent interpretation using preference assignments, the constraint that all trajectories must be assigned to an object in the temporal structure description, and a scoring function. In the example, the most preferred assignment is *(obj1 = C, obj2 = QB, obj3 = RWB, obj4 = RFL)* when there are C, QB, RWB, and RFL trajectories in the data. If the data does not contain a trajectory labeled RWB, the next most preferred assignment is *(obj1 = C, obj2 = QB, obj3 = RTE, obj4 = RFL)*. On the dataset used in this work, this matching algorithm finds the correct trajectory-to-object assignment for the examples, so in the recognition system described here, only the best assignment is used for recognition. However, most likely the top several best matches would need to be used for recognition with a larger dataset where the matching algorithm cannot reliably find the best match. This situation has not been tested.

6.8.2. *Network propagation performance*

The computational efficiency of the multi-agent networks depends entirely upon on the number of logical relations enforced. Without logical relations, the networks are singly-connected trees and evidence can be propagated in time linear to the number of nodes

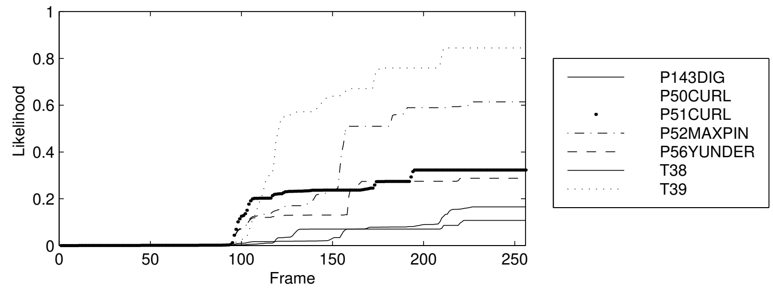


FIG. 12. Result of running 7 play detectors on a t39 play example. Shown is the likelihood of each play having been observed at frame t considering all evidence from frames $0 - t$.

[32] because the size of the largest clique is just 2 regardless of the number of nodes. This tree structure is obtained because the relationships between features are captured *within* nodes. This is a representational compromise because by encapsulating relationships within nodes, some dependencies between states in the network have not been modeled. For example, in Figure 10 the *B:obj3_act2 around obj4_act2* node is not linked directly to *B:obj3_act2 before obj3_act4* node even though the nodes *are* related because they both depend upon *B:obj3_act2*. Fully modeling all the real-world dependencies, however, would result in a heavily linked (and therefore computationally impractical) graph.

One multi-agent network exists for each play in the database, and the networks are evaluated simultaneously as data is observed. Figure 12 shows an example of output curves for the football play detection system, showing the likelihood values for different plays over time. As time progresses and the play begins, the likelihood of the correct play rises above the other play types shortly after motion begins in frame 90. Currently the system is computing the likelihood at each time for each possible play for the entire play. After some preliminary feature computation that takes a few minutes for each play instance, the system requires approximately 1 second of computation per frame per tested play on a 500 MHz Digital AlphaStation. Since each multi-agent network is evaluated independently of the others (other than sharing cached feature computation results), the computation can be parallelized. A control structure could be used to improve efficiency by pruning plays from consideration during processing or pursuing the most rapidly rising likelihood values first, but control strategies are not investigated in this work.

7. RECOGNITION RESULTS AND EVALUATION

This section describes the recognition performance of the system and discusses the strengths and weaknesses of the representation.

7.1. System components and knowledge engineering

The representation for action recognition developed in Section 6 has been applied to a data set of 29 plays consisting of 14 example play types. In order to apply the representation, each play needed to be encoded as a temporal structure description. To describe the action in a football play, general and football-specific knowledge had to be encoded. Sixty player types (e.g. *quarterback*, *receiver*) and *ISA* relationships between player types (e.g. *wide-receiver ISA receiver*) were defined. More than 60 evidence detectors (e.g. *distance(closestAgent))* and their corresponding functions have been defined. Some of these evidence detectors, which are applied to the trajectory data and produce degree of membership values for small sets (e.g. *inContact* $= 0.3$, *nextTo* $= 0.7$), were described in Section 6. Football-specific regions of the field were specified. Some regions are static (e.g. center-of-field, near-sideline-region) and some are defined with respect to the starting position of the ball in each play (e.g. line-of-scrimmage, post-region, defensive region). Over 60 visual goal networks have been implemented.

Although they have desirable properties as a representation for uncertainty, Bayesian networks are laborious to construct. It was not atypical for a thirty node network – assuming all evidence detectors were implemented and working properly – to require 3-5 hours of work to construct and much longer to properly test and debug. Unfortunately, a data set labeled with "ideal" visual goal network outputs is not available or easy to obtain for testing. Therefore, networks were tested by running each network for each player in a few example plays and inspecting the results using a special GUI to check that the detector generated reasonable outputs when applied to any player object. Constructing the visual networks is the most time-consuming and challenging knowledge-engineering task required to implement the action recognition algorithm. Although some networks are generic and could be applied to other domains (e.g. stayingBetween), about half of the networks are specific to the football domain (e.g. catchPass).

The time-intensive knowledge engineering task is adding new visual networks, not adding new temporal structure descriptions. A new temporal structure description can be encoded quickly given a play example, if a sufficiently rich set of football-related visual networks are already available.

7.2. Recognition performance

Section 5 described the data acquisition process. The system has been evaluated on the example set of 29 tracked plays using a database of 10 temporal play descriptions (see [22] for diagrams of each play type). Figure 13 is a confusion matrix showing the final likelihood value obtained for each temporal play description when run on the 29 example plays. A "-" value indicates a play where no good object-to-trajectory consistency match could be found (see Section 6.8.1 and [22]). The examples below the line (i.e. p58 through s35) do not have fully-implemented temporal play descriptions because not all the visual networks required have been implemented. The highest likelihood value obtained on each data file (each row) is marked in bold.

Considering only the top portion of the table, the maximum likelihood value along each row selects the correct play for 21 of the 25 play instances. For each of these examples, the intended play is within the top two ranked options. 3 of the 4 errors are caused by p56yunder examples being misclassified as p52maxpin plays. This error is discussed in the next section.

The bottom section of the table are the likelihoods produced when applying the system to instances of plays for which there is (as yet) no action network due to required but missing visual networks. The discouraging result here is that false positives have values comparable to the correct positives above. That is, while the system is capable of selecting the correct play description, it cannot yet determine when a play does not belong to one of its known categories. This problem can be addressed by strengthening the models of play action by including more action components and temporal relations per agent. The

Name	p143 pdig	p50 curl	p51 curl	p52 maxpin	p54max cross	p56 yunder	p63 up	p63 upa	t38	t39
p143dig (file 1)	**.75**	.49	-	-	.37	.33	-	.24	.53	-
p143dig (file 2)	**.98**	.63	-	-	.75	.71	-	.57	.65	-
p143dig (file 3)	**.93**	.45	-	-	.57	.63	-	.32	.39	-
p143dig (file 4)	**.87**	.35	-	-	.53	.49	-	.27	.30	-
p143dig (file 5)	**.91**	.42	-	-	.50	.36	-	.60	.41	-
p143dig (file 6)	**.86**	.42	-	-	.43	.41	-	.70	.43	-
p143dig (file 7)	**.98**	.58	-	-	.85	.65	-	.57	.36	-
p50curl (file 8)	.19	**.87**	-	-	-	.44	-	.62	.58	.27
p51curl (file 9	-	.21	**.69**	-	-	.27	.35	.34	-	.58
p51curl (file 10)	-	.54	**.95**	-	-	-	-	.55	-	.66
p51curl (file 12)	-	-	**.98**	-	-	-	-	.82	.09	.68
p52maxpin (file 13)	-	-	.37	**.93**	-	.66	.88	-	-	-
p54maxcross (file 14)	.55	.55	.37	.57	**.97**	.48	-	.77	-	-
p56yunder (file 15)	-	-	.47	-	-	**.63**	-	-	-	-
p56yunder (file 16)	-	-	.24	.51	-	**.69**	.39	-	-	-
p56yunder (file 17)	-	-	.75	**.88**	-	.83	.72	-	-	-
p56yunder (file 18)	.61	.26	-	-	-	**.80**	-	.73	.41	.47
p56yunder (file 19)	.38	-	.38	**.78**	-	.62	.57	-	-	-
p56yunder **(file 20)**	-	-	.54	**.76**	-	.64	.34	-	-	-
p63up (file 21)	-	.41	.56	-	-	-	**.87**	-	-	-
p63up (file 22)	-	.61	.79	-	-	-	**.95**	-	-	-
p63up (file 23)	-	.35	.43	-	-	-	**.89**	-	-	-
p63upa (file 24)	-	-	.52	-	-	-	-	.73	.12	**.76**
t38 (file 25)	-	-	-	-	-	-	-	.27	**.83**	.51
t39 (file 26)	-	.25	.39	-	-	.27	.55	.30	-	**.83**
p58 (file 27)	-	-	-	.30	-	-	**.57**	-	-	-
r34wham (file 28)	.35	.62	-	-	.42	.43	-	**.65**	.56	-
s14wham (file 29)	-	-	-	-	-	.27	-	.57	**.72**	.53
s35 (file 30)	.16	.45	.22	**.64**	-	.31	.40	-	-	-

FIG. 13. Likelihood values when the recognition system is run for each of 10 play models over a data set of 29 examples.

weaker the model, the more easily it is matched by some incorrect instance. More detailed models, requiring the construction of more visual networks, should improve the ability of the system to determine that a play is "none of the above."

The next section analyzes some of these results in more detail.

7.3. Recognition case study

The multi-agent data in Figure 13 is difficult to interpret because a football play is not "one action." It is the simultaneous action of 11 people when, even in a "perfect" example, agents are permitted to vary their motion. In this data set, the offensive players are adjusting their motion based on the defense. Therefore, there is no obvious measure of similarity to compare two plays. The examples in the table must be examined individually to understand why the algorithm performs as it does. We believe this is a problem computer vision researchers will increasingly encounter as they attempt to develop systems to identify complex, multi-agent activities.

The "file 20" example in Figure 13 (highlighted) is one of the erroneous matches of a p56yunder example being misclassified as a p52maxpin play. The system returned likelihoods of .64 for p56yunder and .76 for p52maxpin. Figure 14 has a p52maxpin and p56yunder play diagrams overlaid. In addition, the player movement in "file 20" example is overlaid as well. The diagram demonstrates why the system has difficulty classifying the example. The two plays, when executed perfectly, are similar except for the "optional action" of one player. The only large observed difference between the plays is for the rightmost player, who follows a trajectory different from both the p56yunder and the p52maxpin. The temporal structure descriptions currently do not include optional actions. Reasons for this are discussed in the next section. If the optional action were to be modeled, it would contribute evidence to the desired p56yunder classification.

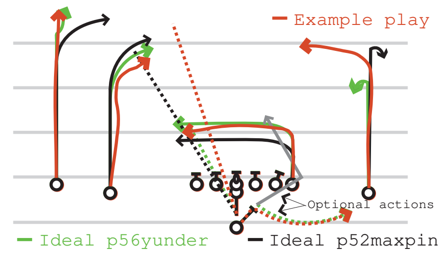


FIG. 14. P56yunder and p52maxpin play diagrams with one p56under example play approximately overlaid. The system returned likelihoods of .64 for p56yunder and .76 for p52maxin.

The algorithm is performing well on a real-world, multi-agent recognition task. However, because of limited data, we were limited to testing with the same plays used to design the visual networks. Limitations of the representation are discussed in Section 7.5.

7.4. Evaluating the multi-agent network structure

What are the multi-agent networks actually doing? Most fundamentally, the networks act as softly-weighted, probabilistic voting mechanisms for uncertain features of the multi-agent action. The conditional probabilities specify the relative weights of action feature components. For instance, consider only the main node, *B:s51 (obj1 obj2 obj3 obj4)* and the compound agent action nodes (e.g. *B:s51 (obj1)*) of Figure 10. These nodes are structured as a naive Bayesian network [10]. The network represents a soft, probabilistic summary of a class (in this case the play), where the class is detected via a set of attributes (in this case combined evidence for an agent's goals) that are assumed to be independent given the class. The network designer specifies $P(attribute|class)$ and $P(class)$. Each compound agent action node is also structured like a naive Bayesian network (e.g. see *B:s51 (obj4)* and children nodes).

The network, however, is not quite a hierarchical naive Bayesian network. For example, binary connections between some nodes model logical relationships. Further, most nodes represent a *relationship* between two agent goals, capturing attribute dependencies.

7.4.1. Augmenting naive Bayes classifiers

The naive Bayesian classifier can be substantially improved by reintroducing a small subset of dependencies between pairs of variables. In particular, one method uses Tree Augmented Naive Bayesian (TAN) networks, where each attribute has as parents the class variable and *at most one other attribute* [12]. TAN classifiers improve the performance of Bayes, and on standard machine learning data sets they are equivalent to or better than some of the best performing (and more complex) methods, such as the C4.5 decision tree classifier [12, 26].

The TAN networks are naive Bayesian classifiers where binary dependencies have been encoded between attributes using links. The multi-agent networks developed here are structured as naive classifiers where binary temporal relations between attributes (i.e. goals) have been encoded *within* nodes. The multi-agent networks contain some additional explicit dependencies between variables due to logical relationships. Both network structures generally assume class-conditional independence but attempt to model the most significant dependencies. For instance, as discussed in Section 6.2, the multi-agent networks do not compute closure on temporal relationships (i.e. if A around B and B around C, A around C is not used unless explicitly encoded by the domain expert). The excellent performance of the TAN networks suggests that the idea of assuming general independence but specifying some important low-order dependencies is a promising approach for modeling real-world domains.

7.5. Representational strengths and weaknesses

The algorithm is now discussed with respect to issues raised in previous sections and the implications of the network structure.

State space size. The state space of multi-agent problems is exponential. The proposed representation circumvents this problem by using many low-order relationships between agents to probabilistically recognize collaboration by observing coordination. The representation is allowing minor variation in individual actions to take place and observing major temporal changes between those actions. These major changes can be encoded in a moderately-sized network. For example, the multi-agent representation uses the temporal analysis functions to condense information from the entire observation interval into a single evidence detector that is then incorporated into the multi-agent network.

Feature space size. The size of the feature space is large because features of most importance *relate* two object trajectories. Using deictic features in visual networks limits feature computation significantly. Using these agent-centered features is reasonable because most agent-goal action behavior can be recognized by analyzing an agent's interactions locally in space and time. The feature space size is further controlled by hierarchically using recognized agent goals as input features for recognition of multi-agent action. In the football domain, a given agent usually only has 3-8 goals per play.

Representing time. The representation for time in this work is simply a set of low-order relationships between intervals. The interval boundaries are fuzzy, determined only by likelihood curves. This temporal representation is weak because it cannot be used for propagating temporal constraints for *reasoning* about temporal activity. The representation, however, is adequate for *recognizing* some temporal behavior.

Scalability in classes. The algorithm must scale in the number of multi-agent actions that can be differentiated. The current algorithm will suffer when two actions are differentiated by only a small subset of motions. For example, if two plays are identical except for one motion of one player, there may be only one or two temporal relationships using the differentiating motion; therefore, noise in the other relations or small variations in agent behavior may prevent the correct play from being selected. The algorithm also fails to scale when actions that require *intentional differentiation* are introduced. For example, if two plays are identical except that in one play the primary receiver is A and in the other play the primary receiver is B, the quarterback's thinking process that led to the change in receivers must be represented (e.g. "A is primary but covered, so I'll throw to B"). The system would need to infer the quarterback's logic by detecting that one player is covered and thereby unavailable to receive a pass (a subtle distinction in football) and then weight that against the cost of throwing to a secondary player, who is less sufficiently covered.

Representational uncertainty. The visual networks and multi-agent networks adequately encode uncertainty from noisy trajectory data. The multi-agent networks softly weight components, thereby making the representation robust to small variations in observed multi-agent action.

Encoding domain rules. The temporal structure descriptions are easy to encode. The components of the descriptions, the visual networks, however, are difficult knowledge engineering challenges. Experience encoding rules on this project suggests that although rule based systems can be problematic to extend due to hidden dependencies between rules, Bayesian networks are difficult and time consuming to design as well. The rule-based system problem of inexplicit rule dependency would appear to be solved by using Bayesian networks, but to make the system possible to construct, multiple network modularity was required that violates independence assumptions.

Comparing likelihoods. Each multi-agent network is independent. Therefore, the likelihood values of networks should ideally not be directly compared. However, as with the visual networks, here the assumption is made that the larger the likelihood difference output by two networks, the more likely that the likelihood value difference is significant. There is one implication of this assumption – each of the multi-agent play networks need to include approximately the same number of attributes of approximately the same sensitivity. If one network contains two relation attributes that are easy to detect and another contains 40 relation attributes, each difficult to detect, comparing the likelihoods is of little value. Therefore, although temporal structure descriptions can be added in without modifying other descriptions, an effort must be made to keep the descriptions at approximately the same level of descriptive detail to avoid unwanted biases.

Modeling optional actions. As discussed in Section 7.3, the algorithm does not currently model optional actions well. The reason for this is that the multi-agent networks are softly-weighting attributes (i.e. evidence) for the class (i.e. the play) in a voting scheme. Therefore, an optional action that is added as an attribute will result in a decrease in computed likelihood of the action if the optional action is not observed. The weighting factor assigned to this optional attribute can be small, but then the optional attribute may not provide sufficient information to differentiate two actions in a noisy environment. An exclusive-or action can be more easily modeled because one attribute must be observed, otherwise the likelihood for the action should legitimately be reduced.

Using hierarchical information. The multi-agent networks implicitly encode the relative importance of the attributes. For example, in Figure 10, the relative importance of attribute *B:s51 (obj2)*, which is an agent goal node, and the logical relationship *B:obj3_act5 xor obj4_act4* is determined by the conditional probability values that are assigned. Different types of evidence can be weighted differently using the conditional probability assignments that are automatically inserted into the multi-agent network structure. For example, $P(xorNode|agentGoal)$ and $P(beforeNode|agentGoal)$ can be set so that evidence from *before* nodes is weighted more in the classification decision than evidence from *xor* nodes. Currently, these conditional probabilities are set by the knowledge engineer during system construction. Ideally, the relative weights of attributes should be specified by the domain expert in the temporal structure description by marking “important,” or “necessary” actions. The relative weights of these actions could then be adjusted when the conditional probabilities are entered into the multi-agent network.

8. CONTRIBUTIONS AND FUTURE DIRECTIONS

The main contributions of this work are:

1. Using an analogy between static object recognition and multi-agent action recognition to motivate the idea that a large set of simple, binary comparisons can be used to recognize planned collaborative actions.

2. Demonstrating how collaborative, highly structured action can be "compiled down" into soft (i.e. probabilistically weighted) collections of visual features detecting spatially and temporally *coordinated* activity; the representation proposed here can be used to recognize "what coordinated actions look like" not "what collaborative actions are."

3. Demonstrating how a multi-agent action description consisting only of temporal and logical relationships between action components, when action components are individual agent goals, can be used to design a computationally tractable system for recognizing collaborative activity using modular probabilistic networks.

Future work should focus on extending the idea of low-order temporal relationships to represent multi-agent actions. A logical inference framework may be needed to extend the temporal representation. In addition, to make the framework practical for many domains, it needs to be improved using learning methods that operate on sparsely labeled datasets to simplify or eliminate some manual knowledge engineering. Two especially important directions are (1) extending the framework to detect "none of the above" actions where the example does not match any of the known action models and (2) extending the temporal representation by allowing temporal logic propagation.

In conclusion, recognition of multi-agent action offers new challenges to the computer vision community. Agents are everywhere in the world, and many of the most useful computer vision tasks will require that recognition systems be capable of identifying multi-agent coordination and collaboration. Representations for single-agent recognition do not straightforwardly extend to the multi-agent domain. This work describes one experimental framework for recognition of highly structured, multi-agent action from perceptual data and highlights the many representational tradeoffs that were required to make the representational practical. We hope this work motivates other researchers to propose new representations that can represent and identify the interaction of agents in everyday scenes.

REFERENCES

1. P. Agre and D. Chapman. Pengi: an implementation of a theory of activity. In *Int'l Joint Conf. on Artificial Intelligence*, pages 268–272, 1987.
2. J.F. Allen. Towards a general theory of action and time. *Artificial Intelligence*, 23:123–154, 1984.
3. J. Azarewicz, G. Fala, and C. Heithecker. Template-based multi-agent plan recognition for tactical situation assessment. In *Proc. of the Sixth Conference on Artificial Intelligence Applications*, pages 248–254, 1989.
4. A.F. Bobick. Movement, activity, and action: the role of knowledge in the perception of motion. *Phil. Trans. Royal Society London B*, 352:1257–1265, 1997.
5. J. Breese. Construction of belief and decision networks. *Computational Intelligence*, 8(4), 1992.
6. H. Buxton and S. Gong. Advanced visual surveillance using Bayesian networks. In *Proc. of the Workshop on Context-Based Vision*, pages 111–123, Cambridge, MA, June 1995. IEEE Computer Society Press.
7. E. Charniak and R. P. Goldman. A Bayesian model of plan recognition. *Artificial Intelligence*, 64:53–79, 1993.
8. P.R. Cohen and H.J. Levesque. Teamwork. *Nous*, 25(4):487–512, 1991.
9. M. Devaney and A. Ram. Needles in a haystack: Plan recognition in large spatial domains involving multiple agents. In *Proc. Fifteenth Nat. Conf. on Artificial Intelligence*, pages 942–947, Madison, WI, July 1998.
10. R.O. Duda and P.E. Hart. *Pattern classification and scene analysis*. John Wiley & Sons, New York, 1973.
11. J. Forbes, T. Huang, K. Kanazawa, and S. Russell. The BATmobile: towards a Bayesian automated taxi. In *Int'l Joint Conf. on Artificial Intelligence*, volume 14, pages 1878–1885, 1995.

12. J. Friedman, D. Geiger, and M. Goldszmidt. Bayesian network classifiers. *Machine Learning*, 29:131–163, 1997.
13. R. Goldman and E. Charniak. Dynamic construction of belief networks. In P.P. Bonissone, M. Henrion, L.N. Kanal, and J.F. Lemmer, editors, *Uncertainty in Artificial Intelligence*, volume 6. Elsevier Science Publishers B.V., 1990.
14. W.E.L. Grimson. *Object recognition by computer: the role of geometric constraints*. MIT Press, Cambridge, MA, 1990.
15. W.E.L. Grimson and T. Lozano-Pérez. Localizing overlapping parts by searching the interpretation tree. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 9(4):469–482, 1987.
16. B.J. Grosz and S. Kraus. Collaborative plans for complex group action. *Artificial Intelligence*, 86(2):269–357, 1996.
17. P. Haddawy, J.W. Helwig, L. Ngo, and R.A. Krieger. Clinical simulation using context-sensitive temporal probability models. In *Proc. of the 19th Annual Symposium on Computer Applications in Medical Care (SCAMC '95)*, 1995.
18. S. Holtzman. *Intelligent decision systems*. Addison-Wesley, Reading, MA, 1988.
19. T. Huang, D. Koller, J. Malik, G. Ogasawara, B. Rao, S. Russell, and J. Weber. Automatic symbolic traffic scene analysis using belief networks. In *Proc. Nat. Conf. on Artificial Intelligence*, pages 966–972. AAAI Press, 1994.
20. M.J. Huber. *Plan-based plan recognition models for the effective coordination of agents through observation*. PhD thesis, University of Michigan, 1996.
21. M.J. Huber, E.H. Durfee, and M.P. Wellman. The automated mapping of plans for plan recognition. In *Int'l Conference on Uncertainty in Artificial Intelligence*, pages 344–351, July 1994.
22. S.S. Intille. *Visual Recognition of Multi-Agent Action*. PhD thesis, Massachusetts Institute of Technology, September 1999.
23. S.S. Intille and A.F. Bobick. Closed-world tracking. In *Proceedings of the Fifth International Conference on Computer Vision*, pages 672–678, Los Alamitos, June 1995.
24. F.V. Jensen. *An introduction to Bayesian networks*. Springer-Verlag, 1996.
25. H.A. Kautz and J.F. Allen. Generalized plan recognition. In *Proc. Nat. Conf. on Artificial Intelligence*, pages 32–37, August 1986.
26. E. Keogh and M. Pazzani. Learning augmented Bayesian classifiers: A comparison of distribution-based and classification-based approaches. In *Uncertainty 99, 7th. Int'l Workshop on AI and Statistics*, pages 225–230, Ft. Lauderdale, FL, 1999.
27. T.S. Levitt, J.M. Agosta, and T.O. Binford. Model-based influence diagrams for machine vision. In M. Henrion, R.D. Shachter, L.N. Kanal, and J.F. Lemmer, editors, *Uncertainty in Artificial Intelligence*, volume 5, pages 371–388. Elsevier Science Publishers B.V. (North Holland), 1990.
28. J. McCarthy and P.J. Hayes. Some philosophical problems from the standpoint of artificial intelligence. In B. Meltzer, D. Michie, and M. Swann, editors, *Machine Intelligence 4*, pages 463–502. Edinburgh University Press, Edinburgh, Scotland, 1969.
29. H.-H. Nagel. From image sequences towards conceptual descriptions. *Image and Vision Comp.*, 6(2):59–74, 1988.
30. H.-H. Nagel, H. Kollnig, M. Haag, and H. Damm. Association of situation graphs with temporal variations in image sequences. In *Computational Models for Integrating Language and Vision*, pages 1–8, November 1995.
31. L. Ngo, P. Haddawy, and J. Helwig. A theoretical framework for context-sensitive temporal probability model with application to plan projection. In P. Besnard and S. Hanks, editors, *Int'l Conference on Uncertainty in Artificial Intelligence*, volume 11, San Francisco, CA, August 1995. Morgan Kaufmann Publishers.
32. J. Pearl. *Probabilistic Reasoning in Intelligent Systems : Networks of Plausible Inference*. Morgan Kaufmann Publishers, San Mateo, CA, 1988.
33. C.S. Pinhanez. *Representation and recognition of action in interactive spaces*. PhD thesis, Massachusetts Institute of Technology, June 1999.
34. J. Pinto and R. Reiter. Reasoning about time in the situation calculus. *Annals of Mathematics and Artificial Intelligence*, 1995.
35. M.E. Pollack. Plans as complex mental attitudes. In P.R. Cohen, J.L. Morgan, and M.E. Pollack, editors, *Intentions In Communication*, pages 77–103. MIT Press, 1990.

36. A.R. Pope. Model-based object recognition: a survey of recent research. Technical report, University of British Columbia, 1994.

37. G.M. Provan and J.R. Clarke. Dynamic network construction and updating techniques for the diagnosis of acute abdominal pain. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 15(3):299–307, March 1993.

38. D.V. Pynadath and M.P. Wellman. Accounting for context in plan recognition with application to traffic monitoring. In P. Besnard and S. Hanks, editors, *Int'l Conference on Uncertainty in Artificial Intelligence*, volume 11, 1995.

39. L.R. Rabiner and B.-H. Juang. *Fundamentals of speech recognition*. Prentice Hall, 1993.

40. J. Rehg and T. Kanade. Visual tracking of high DOF articulated structures: an application to human hand tracking. In *Proc. European Conf. Computer Vision*, pages 35–46. Springer-Verlag, May 1994.

41. P. Remagnino, T. Tan, and K. Baker. Agent orientated annotation in model based visual surveillance. In *Proc. Int'l Conf. Computer Vision*, volume 6. IEEE Computer Society, Narosa Publishing House, January 1998.

42. G. Retz-Schmidt. A REPLAI of SOCCER: recognizing intentions in the domain of soccer games. In *Proc. European Conf. AI*, volume 8, pages 455–457, 1988.

43. R.D. Rimey and C.M. Brown. Where to look next using a Bayes net: incorporating geometric relations. In G. Sandini, editor, *Proc. European Conf. Computer Vision*, pages 542–550. Springer-Verlag, 1992.

44. Y. Shoham. *Reasoning about change: time and causation from the standpoint of artificial intelligence*. MIT Press, Cambridge, MA, 1988.

45. P. Smyth, D. Heckerman, and M. Jordan. Probabilistic independence networks for hidden Markov probability models. Technical report, MIT, 1996.

46. M. Tambe. Tracking dynamic team activity. In *Proc. Nat. Conf. on Artificial Intelligence*, pages 80–87, August 1996.

47. Trakus, 1999. A company producing football trajectory tracking devices to be used in the NFL. http://www.trakus.com.

48. A. D. Wilson and A. F. Bobick. Learning visual behavior for gesture analysis. In *Proc. IEEE Int'l. Symp. on Computer Vision*, Coral Gables, Florida, Nov. 1995.

49. L.A. Zadeh. Fuzzy sets as a basis for a theory of possibility. *Fuzzy Sets and Systems*, 1:3–28, 1978.

APPENDIX

A.1. DESCRIPTION OF EXAMPLE PLAY

The example play will be called the *simple-p51curl* (or *s51* for short) because it is a simplified example of the real p51curl play. The s51 is represented in the diagram in Figure 2b. The example comprises four offensive players – obj1, obj2, obj3, and obj4 – as well as one defensive player – D. The horizontal lines indicate the yardlines of a playing field, which are 5 yards apart. The s51 starts with obj1 holding the BALL and obj2 next to obj1. All four offensive players get set in special positions relative to one another and do not move. The obj3 and obj4 both are in positions that are a yard back from the *line of scrimmage*, or *LOS*, which is the line parallel to the yardlines centered at the starting BALL position. Obj1 hands the BALL to obj2. At that time, obj3 and obj4 each begin running *pass patterns* that are coordinated in space and time. Obj4 tries to run straight down the field for 5 yards, then *cuts*, turning quickly towards the inside of the field, and runs parallel to the yardlines. Obj3 tries to run straight down field for 5 yards, then cuts towards the outside and runs parallel to the yardlines. Obj4 and obj3 cut at the same time, and when they do, obj4 tries to run just behind obj3 with respect to the LOS. As obj3 and obj4 run, obj2, who has the BALL, performs a *dropback* action, moving backwards a few steps. Obj1 *blocks* by trying to stay between obj2 and the D. Obj3 and obj4 cut just before obj2 *passes* the BALL. Obj2 is most likely to throw the BALL to obj3 (the heavy dotted line indicates the BALL trajectory), but in some cases obj2 may throw the BALL to the obj4.

Although many players are *eligible* to throw a pass, in reality it is nearly always obj2. In this example, only obj3 and obj4 are both *eligible receivers* of a pass. The D is trying to prevent obj2 from throwing the BALL. The play ends if the BALL is not caught by a player or if a player with the

BALL is *tackled* by a defensive player. If a receiver catches the BALL, the player will try to continue running with the BALL downfield until tackled.

A.2. TEMPORAL FUNCTION HEURISTICS

This appendix describes the heuristic functions used to compute the temporal functions given goal likelihood curves.

ObservedNow. Instantaneous value indicating certainty that some goal is being observed at the given time. Assume: t (currentTime), g (Goal); Likelihood value of goal at t, $l_g(t)$, where $l_g(t) = 0$ when $g(t) = NULL$. Evidence function minimal width threshold, w_g; Sum, $S = \sum_{\tau=t}^{t-w_g} l_g(\tau)$; ObservedNow$(g)_t = \frac{S}{w_g}$.

Observed. The maximum value for ObservedNow for frames $0 - t$.
Observed$(g)_t = argmax($ObservedNow$(g)_0, \ldots,$ObservedNow$(g)_t)$.

BeforeNow. Instantaneous value indicating certainty that some goal, $g1$ is observed as before some other goal, $g2$ at the given time. If t (currentTime), $g1$ (Goal1), $g2$ (Goal2), and w (Minimum before-gap-width) = min(w_{g1}, w_{g2}); Difference between goal observed values at t, $D_t = max(0, g2{-}g1)$; Maximum difference between two goal values at any time, D_{max} =argmax$(D_0, \ldots, D_t)$, where t_{max} indicates the time frame of D_{max}. Scaling factor, based on width of temporal interval from current time to time indicating the maximum difference between two goals, $s = \frac{t-t_{max}}{w}$; BeforeNow$(g)_t = s *$ min(D_t, D_{max}).

Before. The maximum value for BeforeNow for frames $0 - t$
BeforeNow$(g)_t = argmax($BeforeNow$(g)_0, \ldots,$BeforeNow$(g)_t)$.

AroundNow. Instantaneous value indicating certainty that some goal, $g1$ is observed as around some other goal, $g2$ at the given time. If t (currentTime), $g1$ (Goal 1), and $g2$ (Goal2) then Minimum overlap width, w = min(w_{g1}, w_{g2});
$S = \sum_{\tau=t-\frac{w}{2}}^{t+\frac{w}{2}}($ObservedNow$(g1)_\tau *$ ObservedNow$(g2)_\tau)$; AroundNow$(g)_t = \frac{S}{w}$.

Around. The maximum value for AroundNow for frames $0 - t$
AroundNow$(g)_t = argmax($AroundNow$(g)_0, \ldots,$AroundNow$(g)_t)$.

ACKNOWLEDGMENT

The authors thank the New England Patriots for providing video of the team; S. Srinivas and J. Breese for making their IDEAL package publicly available; the Microsoft Research Decision Theory Group for making MSBN available; and AT&T, for providing the GraphViz software. This work was supported in part by DARPA Contract DAAL01-97-K0103 and ORD Contract 94-F133400-000.