Visual Recognition of Multi-Agent Action

by

Stephen Sean Intille

B.S.E., University of Pennsylvania (1992)
S.M., Massachusetts Institute of Technology (1994)

Submitted to the Program in Media Arts and Sciences,
School of Architecture and Planning,
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 1999

Author . . . .
Program in Media Arts and Sciences
August 6, 1999

Certified by . . . .
Aaron F. Bobick
Associate Professor of Computational Vision
Massachusetts Institute of Technology, USA
Thesis Supervisor

Accepted by . . . .
Stephen A. Benton
Chairman, Departmental Committee on Graduate Students
Program in Media Arts and Sciences

Visual Recognition of Multi-Agent Action

by

Stephen Sean Intille

Submitted to the Program in Media Arts and Sciences,
School of Architecture and Planning,
on August 6, 1999, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

Abstract

Developing computer vision sensing systems that work robustly in everyday environments will require that the systems can recognize structured *interaction* between people and objects in the world. This document presents a new theory for the representation and recognition of *coordinated multi-agent action* from noisy perceptual data.

The thesis of this work is as follows: highly structured, multi-agent 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 theory is developed and evaluated by examining general characteristics of multi-agent action, analyzing tradeoffs involved when selecting a representation for multi-agent action recognition, and constructing a system to recognize multi-agent action for a real task from noisy data.

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 algorithm is tested using a database of American football play descriptions. A system is described that can recognize single-agent and multi-agent actions in this domain given noisy trajectories of object movements. The strengths and limitations of the recognition system are discussed and compared with other multi-agent recognition algorithms.

Thesis Supervisor: Aaron F. Bobick
Title: Associate Professor of Computational Vision
Massachusetts Institute of Technology, USA

Doctoral Committee

Thesis Advisor . . . .

Aaron F. Bobick
Associate Professor of Computational Vision
Massachusetts Institute of Technology, USA

Thesis Reader . . . .

W. Eric L. Grimson
Professor of Computer Science and Engineering
Bernard Gordon Chair of Medical Engineering
Massachusetts Institute of Technology, USA

Thesis Reader . . . .

Hans-Hellmut Nagel
Professor für Informatik
Universität Karlsruhe (T.H.), Germany

Contents

List of Figures

Acknowledgments

Thanks to: my advisor, Aaron Bobick, for too many things to mention; my readers, Eric Grimson and Hans-Hellmut Nagel, for the insightful questions and comments; undergraduate researchers Ann Bui, Qian Wang, Nick Lesica, Kamal Mokeddem, and Anthony Hui, for many hours of sometimes tedious work; the HLV group, for years of great discussion, brainstorming, and fun times; everyone in Vismod present and past, for creating a vibrant intellectual atmosphere; my friends long out of school, for keeping straight faces when they asked how much longer I had to go; my friends still in school, for making me feel much better; my family, for being the terrific people that they are; and my wonderful, generous wife Amy, for her companionship, support, and adventurous spirit.

Additional thanks to: the New England Patriots, former coach Bill Parcells, and the video staff, for providing video of the team; S. Srinivas and J. Breese for making their IDEAL package available; the Microsoft Research Decision Theory Group for making MSBN available; and AT&T, for making GraphViz available.

Chapter 1

Introduction

Computer systems that can reliably observe, recognize, and respond to *action* using visual sensors will ultimately enable new computational applications ranging from automatic video annotation and retrieval to life-enriching interactive environments. Visual recognition of action, however, is a difficult problem, and most computer vision research has focused on recognizing the action performed by a single person in isolation from other objects (e.g. a person waving, pointing, or exercising in an empty room). Unfortunately, while these single-person scenarios are easy to construct in the laboratory, they are rarely found in the real world because we live in a world crowded with other people and objects.

This work refers to any person or thing that moves as an *agent*. Our world is filled with agents. People are continuously interacting with other agents in their environments – often with more than one simultaneously – and changing their behavior in response. Therefore, it is not surprising that people recounting scenes commonly describe the static and temporal relationships *between* people and nearby objects. Consider this description of an everyday scene of a parking lot, as annotated by an observer:

> A red car just pulled into the parking lot entrance and drove next to the big sign. The driver got out of the car, walked around the car, and then headed toward the store. A green car waited for the person to get out of the parking lot before proceeding to drive by the front of the store. About 20 seconds later, two people exit the store and walk together to their car. The driver gets in the car and unlocks the passenger door, and then the second person opens the door and gets into the car.

The behaviors of the people and vehicles in this scene are clearly coordinated. Cars stop to wait for people. People walk together towards vehicles and interact with one another as they get into the vehicles and drive. The structure of the parking lot significantly affects the patterns of motion of people and vehicles as they move to avoid hitting each other and other obstacles in the environment. Therefore, when people describe the scene, they use descriptions based on the relative position and motions of the multiple objects.

New 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 not only simple motion but actions involving the interaction between multiple agents [Nag88, Bob97]. This dissertation explores issues related to designing a representation of multi-agent action that can be used for the visual recognition of structured multi-agent activity in real domains with noisy perceptual data.

One goal of the work presented here is to motivate further study of multi-agent action recognition by developing, implementing, and testing a recognition algorithm for identifying a particular type of team-based, collaborative, multi-agent action. The approach that is developed applies only to a subset of multi-agent action recognition problems, but exploring this particular application domain has helped identify some general issues that any multi-agent recognition system must address. This document describes the performance, strengths, limitations, and scope of the proposed representation.

1.1 Motivation

This project resulted from a desire to design a real, multi-agent action recognition system for domains with noisy, perceptual sensors. Applications that might benefit from the development of a system that can robustly recognize multi-agent action are listed below.

Interactive environments. Sensing technology is currently being used to create interactive home and office environments that respond to the activity of occupants (see [BID+99] for a review). A house has been constructed, for instance, that uses motion detectors to reduce energy consumption by learning user activity patterns and then automatically controlling lights, temperature, and appliances [Moz98]. As people live and work in their environments, however, they are continually interacting with other objects and other people (e.g. a family cooking and eating dinner together). Versatile responsive environments need to be capable of recognizing multi-agent interactions between people. Existing interactive environments model only single agent activity (typical examples are [MPB+94, Moz98]; for an exception see [BID+96]).

Surveillance. The security surveillance industry employs thousands of workers in tedious jobs watching video that, for the most part, is uneventful. These error-prone security tasks might be improved using systems that can monitor video and automatically recognize typical and atypical action. For example, such a system might detect that a car is being vandalized in a store parking lot. A military surveillance system might automatically analyze satellite data and recognize complex actions like the interaction of ground military units [DR98] and the activity in commercial and military ports. Existing surveillance systems generally recognize single agent activity (e.g. [RE91]) or simple interactions between two agents using some scene context (e.g. recognizing actions like greeting, passing by, or picking up [FO96, RTB98, ORP99, ISBG99]). Some systems compute statistical correlations in data to detect unusual activity [MH98, GSRL98] and could potentially detect *unusual* activity involving more than two agents; however, more *explicit* models of the multi-agent action will be required to recognize *specific* multi-agent actions.

Traffic monitoring. Systems exist that can automatically track vehicle movements on short segments of road from video input (e.g. [KDTN92, TSB94]) and annotate action in traffic scenarios with English verbs [HSA+89, KHN91, KNO94, BG95, HTSN97, HN98] and text descriptions [GN96]. These systems, however, do not recognize actions performed collaboratively by more than one agent simultaneously. Recognition of multi-agent actions, such as vehicle caravanning or interactions of people and vehicles in parking lots, should be useful for future traffic management applications.

Robotic collaboration. Creating autonomous robots that can effectively navigate their environment and interact with other people and robots will require systems that observe the multiple agents in the immediate vicinity and recognize collaborative behavior [DM90, HD93].[1] Industrial robots, for example, may need to recognize group assembly tasks.

Sports entertainment and training. Broadcasters of sporting events already use computer vision video manipulation techniques to increase the informational or advertising content of their programming. Current applications include automatic insertion or removal of advertising in video from sporting arenas as well as semi-automatic tracking of players and manipulation of video for special video effects [Tra99, Ora99]. Systems that automatically analyze the activity in a game and extract action labels like "tackling," "touchdown," "interception," and "R34 run play" could be used to provide additional content to the viewer, make new interactive sporting applications possible, and provide coaches and fans with powerful tools for detailed post-game analysis of game action. Professional teams spend millions of dollars annually on video staff and equipment to manually annotate video of team practices and games. Automatic annotation requires that a computer system recognize actions of multiple people interacting in collaborative ways based on trajectory data of their movements over time. Nearly all agent movement in a sporting event is affected in some way by the movement of other agents.[2] Computer vision systems have been designed that recognize simple actions by analyzing video of sporting events [GSC+95, SLJ98], but these systems do not recognize team-based, multi-agent actions.

All of these domains have interactions involving groups of agents where the agents are responding to their environments based upon individual and *collaborative* goals; interaction with other agents is sometimes friendly and sometimes adversarial.

1.2 The task: recognizing team action

In this work, the task of recognizing American football plays has been selected in order to investigate the general problem of multi-agent action recognition. The input for the test system consists of trajectory data for each object (i.e. players and the ball) moving in the scene. Companies are actively developing and marketing systems that will acquire this data with high spatial and temporal accuracy using semi-automatic visual trackers and helmet tagging hardware [Tra99, Ora99]. The trajectory data can be partially recovered directly from video data using context-sensitive visual tracking [IB95].

[1] Robots that interact with other robots in environments without humans may not require systems that recognize collaborative activity of other robots if the robots can explicitly communicate their plans to one another. However, if the robots do not use the same communication protocols or are unable to communicate with one another, the devices will need the ability to recognize the collaborative multi-agent action of their robotic peers.

[2] Even agent movement that is predetermined before the play will be modified slightly to accommodate the action of players who are nearby in space. For example, an agent running "straight down the field" may swerve somewhat to avoid a defensive player.

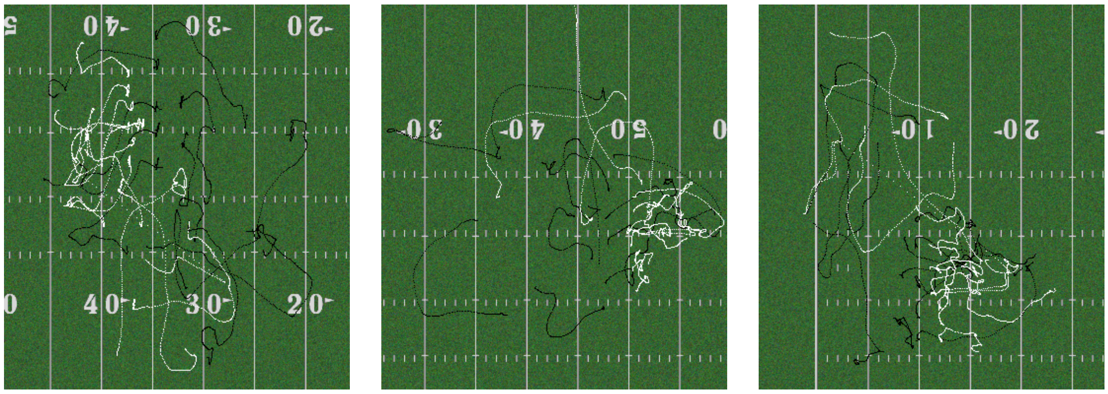


Figure 1-1: Three examples of a p51curl play. The lighter trajectories are the offensive players. The data provided to the system consists of trajectories for all the objects including the ball, the approximate orientation of each object at each point along its trajectory, and a position label for each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

The task in this work is to construct a system that can identify a given set of trajectories as a particular play from a pre-existing database of play models. A football coach provides the play models. Figure 1-1 shows 3 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 at each time. Using a database of plans, knowledge about football, computation of perceptual features from the trajectories, and propagation of uncertain reasoning, the recognition system developed here can correctly determine that each of the examples shown in Figure 1-1 is in fact an example of a p51curl play.

1.3 Contributions

The thesis 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." These issues are briefly described below.

- Although football play recognition is just one multi-agent recognition task, in this work, the football domain is used to uncover general multi-agent recognition issues. The proposed recognition technique is introduced within the context of existing object recognition and action recognition approaches, and this document explores when it might be adequate for recognition in other domains.

- One issue raised by this work is whether recognition of collaborative action requires representing and reasoning about the intentionality of the agents in the scene. Here the following idea is proposed. Although representing *collaborative* action may require explicit intentional reasoning, highly structured multi-agent collaborative action can be "compiled down" into soft (i.e. probabilistically-weighted) collections of visual features detecting spatially and temporally *coordinated* activity. The new representation is capable of recognizing typical collaborative activity for a real task using real data. The representation, however, does not support reasoning about atypical instances of action. Essentially, the representation proposed here can be used to recognize "what coordinated actions look like" not "what collaborative actions are." The strengths and weaknesses of the representation (and other possible representations for multi-agent action) are discussed by exploring this distinction.

- The method used to achieve a practical system that can recognize visual cues of agent coordination is motivated by prior work in static object recognition. The object recognition work suggests that, given a complex object, unary and binary matching constraints can be used to search for an object match in feature data and that higher-order feature comparisons generally escalate computational cost without resulting in significant performance gains [Gri90]. This work extends that observation to the temporal domain and demonstrates a recognition system based on the idea that low order temporal consistency typically implies correctness for a highly structured multi-agent action.

- The multi-agent actions investigated in this work can be described using temporally and logically linked "action components." These components are assumed to be agent-based goals. Each goal is detected by a small Bayesian network that softly weights uncertain perceptual evidence. These networks map from low-level, trajectory features to high-level, semantic concepts used by experts to describe action. They are the building blocks for group action recognition. The goal detectors integrate cues computed within a local space-time region centered on each player and keep the number of features computed manageable by using deictic feature references.

- Detecting multi-agent action in real domains requires representing three types of uncertainty: uncertainty in the data and sensing, uncertainty in the action component detectors and their semantics, and uncertainty in the multi-agent action descriptions themselves resulting from variability in action execution. The proposed system models all three types of uncertainty using multiple Bayesian belief networks with structures designed to maintain efficiency but also capture important temporal variations in multi-agent action.

- The representation for multi-agent action recognition, developed in Chapter 6, has four main components: a temporal structure description (an intuitive description of high-level, collaborative action), visual networks (Bayesian networks for single agent goal detection), temporal analysis functions (that temporally compare agent goals), and an automatically generated multi-agent belief network (probabilistically integrating evidence supporting the observation of particular multi-agent actions).

- Multi-agent actions are recognized using belief networks that are constructed automatically from the temporal structure descriptions. These networks have a particular structure that is efficient for exact evidence propagation and that represents the temporal dependencies of the multi-agent action. The networks are "compiled down" descriptions of visually observable coordinated activity.

The goal of this project is to evaluate the thesis via construction and evaluation of a perceptual, multi-agent recognition system. A computationally tractable system has been implemented that recognizes collaborative, multi-agent action. The system is tested on a real dataset of football play trajectories. A primary contribution of this work is a description of the issues and tradeoffs considered when selecting the presented representation for multi-agent collaborative action recognition.

1.4 Outline

The remaining chapters of this proposal are structured as follows. Chapter 2 discusses properties of multi-agent action that impact the development of a representation for recognition from perceptual data. The role of intentional reasoning is considered but explicitly excluded from the approach developed here. Chapter 3 describes the football play database used in this work and the data acquisition process used to obtain trajectory data from video of football plays. Chapter 4 describes a test system that was developed for labeling the position of each offensive player in static scenes of football formations. The labeling algorithm, which uses context-rule sets and a process that searches for a consistent description of a scene, proved difficult to extend to the temporal domain of play recognition; it did, however, influence the development of a new representation. Chapter 5 discusses previous work in object recognition, plan recognition, and the representation of uncertainty that motivates the principles used to develop the recognition algorithm presented in Chapter 6. Chapter 7 evaluates recognition results for the algorithm and critiques the proposed representation. Chapter 8 concludes with a summary of contributions and discussion of possible extensions.

Chapter 2

Multi-agent action

Multi-agent action is ubiquitous in the world. Computational systems that would benefit from the ability to recognize interactions involving more than one person and/or object include systems that perform analysis of sporting events, identification of military situations, recognition of traffic behaviors for traffic monitoring, surveillance of scenes, and collaborative robotic activity.

This chapter examines the general properties of *collaborative* multi-agent action that differ from single-agent action. Three specific example recognition domains are introduced that will be used for illustration in chapters throughout this document. These examples are analysis of traffic intersections, analysis of action in everyday interactive environments, and analysis of action in football games. The recognition system developed in this work provides insight into how to represent some, but not all, of the multi-agent actions that are typically present in these domains. All three domains, however, will be used to identify the strengths and weaknesses of the technique. The examples are followed by a discussion of general characteristics of multi-agent action and then issues specific to the football task. This chapter concludes with a discussion of the role of "intention" in multi-agent action and how intention impacts the development of a model for the recognition of collaborative, multi-agent action from noisy, visual input.

2.1 Examples

2.1.1 A traffic intersection

Computers that can recognize activity at a traffic intersection could be used for automatic traffic control and accident detection. Currently, traffic scene analysis and annotation is an active area of research [MNN81, MN90, TB92, HKM$^+$94, Nag94, PW95, HN98]. Typically, a single camera is placed overlooking a street or intersection. One goal is to develop computer systems that can accurately track as many objects as possible. Some systems recover not only a trajectory in the (static) image frame but also the trajectory on the ground plane, object pose, and object model parameter information for each object in view using contextual knowledge about the scene (e.g. [KDN93, TSB94]). Ultimately, the goal is to use the recovered trajectory and object identification data to annotate the video with useful descriptors of activity. Important actions include waiting (for some other vehicle), moving (with respect to the road or some other vehicle), turning (with respect to the road), passing by (another vehicle), giving way (to some other vehicle), stopping, and following (another vehicle). Chapter 1 contains a casual observer's description of a parking lot scene. That scene contains actions where cars "give way" to each other and to other people. Cars "wait" for people to cross the car's path, and people collaboratively enter parked cars.

2.1.2 An interactive environment

Computers that can observe everyday environments, detect what people are doing, and control devices in the environment could be used to simplify people's everyday lives by providing additional convenience, safety, and entertainment. So called interactive environments are people-oriented spaces instrumented with cameras and other sensors that are designed to unobtrusively sense and respond to human action (early examples are described in [Tor95, Pen96, Pro98]). The home kitchen is one common environment that may someday be augmented with video cameras and networked with sensor-enabled appliances. For example, assume a kitchen environment in a typical home is continuously monitored with visual sensing that detects the position of occupants and objects. Further, assume that many of the appliances in the kitchen have sensors that detect situations like "door open," "toaster heating element on," etc. and that all the sensor information is sent to a central processor. The system might observe the following typical action:

> A person (person-1) moves to the cupboard and removes a bowl. A second person (person-2) enters the kitchen. Person-1 moves to a shelf and picks up a box of oatmeal. Person-1 then pours some oatmeal in the bowl, takes the bowl to the sink, turns on the sink, and adds some water to the bowl. Meanwhile, person-2 has opened the refrigerator and pulled out some milk, placing the object on the counter. Person-1 opens the microwave and puts the bowl inside, setting the microwave to cook and closing the door. Person-2 finds another bowl. Person-2 then waits as person-1 removes the bowl from the microwave and pours oatmeal into the other bowl. Person-2 then adds milk to each bowl.

Making oatmeal is one example of an action the environment might attempt to recognize. A system that can recognize such events and that has output devices in the kitchen (e.g. controllable appliances, displays, and audio) could potentially create a safer and more interesting kitchen environment by providing cooking information and guidance.

2.1.3 The p51curl football play

Currently, all professional and many collegiate American football teams each spend hundreds of thousands of dollars annually to record and then manually annotate video of each play they run in every game and practice. The annotated video is used for analysis and for automatic video retrieval. These capabilities are unavailable to sports teams with more modest financial resources because manual annotation of video is a tedious, time-consuming, and therefore, expensive task.

A computer system could be used for automatic video analysis of the action in the game to provide information to a team for analysis, football viewers for review, or computer controllers for automatic camera shot selection. 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 2-1 shows four frames from video

Figure 2-1: Four image frames from a video sequence of a football play.

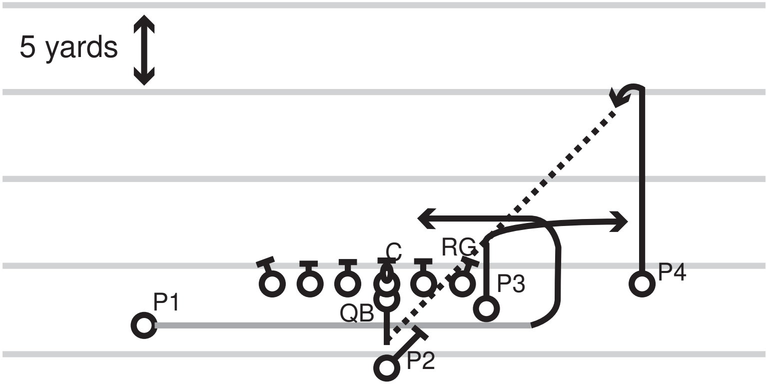


Figure 2-2: Coach's chalkboard diagram for a p51curl play.

of a particular play. Readers unfamiliar with American football and football terminology should refer to Appendix A before continuing.

In this video there are multiple people interacting with one another and with the ball. Action labels of interest in this play include blocking, cutting, running, and the offensive team's attempted play. In this work, recognition of collaborative team action in the football domain is used as the primary motivating example for the development of a representation for multi-agent action.

A football play is a plan that coordinates the movement of 11 offensive players. Typically, plays are represented using "chalkboard" diagrams, like the one shown in Figure 2-2 diagramming a play called a *p51curl*. Blocking motion is noted by a short perpendicular line. The man-in-motion (MIM) movement is indicated in gray. The primary pass option is noted by the dotted line.

The diagram indicates that most of the offensive players block for the QB player, as the QB receives the ball, drops back, and tries to throw the ball to one of the receivers. The receivers run pre-determined motion patterns. Although not explicitly shown in the diagram, the receiver patterns are temporally coordinated so that, for example, P1 and P3 turn at the same time.

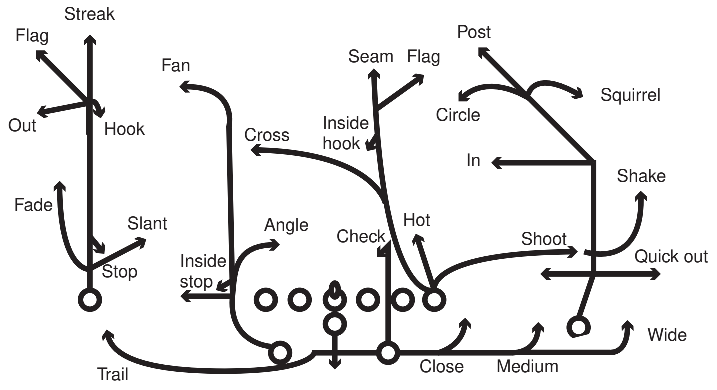


Figure 2-3: Football play pass patterns.

The diagram describes the most typical desired action *from a particular starting formation*. The offense, however, wants to prevent the defense from recognizing its plan based on starting positions. Therefore, most plays can be run from multiple starting formations.

Figure 2-3 shows the different types of pass patterns that receivers can run. The set is limited by constraints imposed by the game. Runners, for example, do not typically have time to make more than one cut during a play. Rules of the game also prohibit (or in practice make not useful) some patterns from certain starting positions. Pass patterns are usually coordinated by a coach in order to maximize the potential success of a play by confusing the defense and giving the offense several possible play options to pursue.

2.1.4 The simplified p51curl football play

Although offensive football plays consist of 11 offensive players, for clarity a simplified play example will be used in this document 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 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 2-4. 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

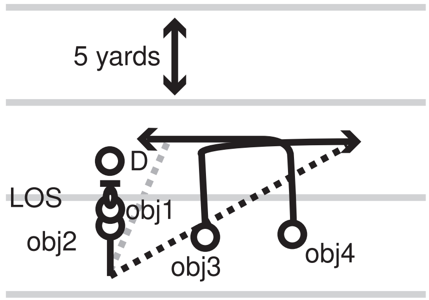


Figure 2-4: A football play diagramming the simple-p51curl example play used in this paper. The play consists of 4 offensive agents (obj1 through obj4), 1 defensive agent (D), and a ball. Also indicated is the line-of-scrimmage (LOS) and some 5-yard marker yardlines.

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.

2.2 Characteristics of multi-agent action

The three example domains all contain multiple people and objects. In this work, any object that moves or can be moved (sentient or not) is considered to be an agent. Fundamentally, multi-agent action recognition differs from single-agent action recognition because the interaction between the multiple agents requires modeling the *interaction* between the agents in the environment. Understanding the distinction between single-agent and multi-agent action is a theme of this work. Some general characteristics of all multi-agent action are described below.

State space size. When a single object is in a scene, an action representation that models the change in state of the object over time may be sufficient to recognize the action from noisy data. Probabilistic state transition diagrams have worked well in practice for modeling a single-agent undergoing changes in state [SHJ96]. In multi-agent action, however, the states of multiple agents must be represented. 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 time.

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. For example, hand gesture recognition algorithms typically use the shape or position of the hand or arm over time for developing motion models of hand gestures (e.g. see [WB95, RK94, SP95]). These features are usually computed without considering other objects in an environment (and usually not even considering self-occlusion). 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. For example, the *distance between* two agents over time is an important feature, often giving clues as to whether two agents are physically interacting with one another. The problem is that in an environment with N agents, there are $\binom{N}{2}$ binary distance features that can potentially be computed and used by the representation. It is also possible, however, to compute n-ary distance features, such as the distance between a single agent and *groups* of agents. Using this agent-to-group comparison substantially increases the number of distance features that can be computed. The number of features increases further when a distance feature is computed between *two* groups of agents. Distance is only one useful n-ary comparison feature. Some (of the many other) n-ary features of interest are relative orientation, whether some agent will intersect some particular subgroup of people, and whether someone is moving in some particular way with respect to some region. Recognition algorithms need an explicit or implicit focus of attention mechanism to avoid computing evidence that provides little discriminatory power. For instance, although it is possible to compute the distance between any player and any group of players in the football domain, only a small subset of all possible groups (e.g. the front line, the receivers) will provide information that may indicate or counter-indicate play actions.[1]

Feature complexity. Single-agent features like acceleration are straightforward to compute from trajectory data, primarily because only one free variable needs to be considered – the length of the temporal window – and this variable can be easily set by the system designer.[2] Many useful multi-agent comparison features of interest, however,

[1] This problem is compounded by the lack of large, labeled datasets in many domains. An algorithm that tries to learn which features are relevant is likely to make spurious data-to-label associations because the data will not contain examples that span the feature space.

[2] This window can also be made time and space invariant by using hierarchical processing.

have several free parameters because they combine input from more primitive feature detectors. For example, computing a relative feature like behind(A,B) that returns results consistent with the typical person's definition of "behind" can require that the function integrate information from other feature detectors such as size(A), size(B), xDistance(A,B), yDistance(A,B), orientation(A), and orientation(B).[3] Functions that compute features comparing objects to groups such as behind(A,(B,D,G)) (i.e. is A behind the group (B,D,G)) are even more complex.

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 [RJ93a]. 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 when one of the distinguishing characteristics between two multi-agent actions is the temporal relationships between each action's individual agent actions. Several issues arise: what type of temporal relationships can be detected, what are the primitives being temporally compared, how much propagation of temporal constraints is performed, and how (if at all) is temporal consistency between agent action used for constraining action recognition [Pin99]?

Explicit search and object assignment. In single object domains, there is never any doubt which object in an action model matches to which object in the scene – there is only one object. In multi-agent domains, however, 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 matching is required. Heuristic search and match evaluation criteria need to be specified.

Learning. The search space and the exponential feature space create at least two problems for most learning algorithms used for computer vision single-agent action representations, such as HMMs. First, most learning algorithms assume a complete[4] dataset of examples can be observed during training; therefore, models with even a small set of input features require hundreds of training examples [RJ93b, You93]. In contrast, for many real-world, multi-agent actions, it will be difficult, if not impossible, to obtain a database that contains a complete set of annotated examples. Examples include the

[3] Even typeOf(A) and typeOf(B) could affect a person's evaluation of behind(A,B).

[4] Roughly spanning the feature space.

football domain and the task of recognizing team plays. Learning algorithms that are not provided with a pre-existing structural model of the domain would require a database of plays where nearly every possible play outcome has been observed. Obtaining this database, however, is problematic because a play, which can start with players in a large number of field configurations, consists of 11 players reacting quickly and physically during an approximately nine second interval. When the feature set consists of the velocity and acceleration of a single agent (or two agents treated as one entity) with a few shape parameters at each time, learning is sometimes manageable (e.g. [WB95, SP95]). However, one of the few HMM recognizers that recognizes simple multi-agent actions, such as "two people greeting," required so much training data that a simulation program was designed to generate data used to train the HMM system [ORP99]. If simulation programs are required for training that require a user-specified model of action, the user-specified models themselves might provide sufficient constraint for a recognition system.

The complexity of agent interaction. As more agents are added to a scene, the complexity of multi-agent and single-agent action will change. In general, the more agents that are in the environment, the more constraint there may be on the way that a given multi-agent, collaborative action can be performed. For example, in football, the more receiver players that are on one side of the field, the more carefully a coach will coordinate their movements; the coach plans the movement so that players do not run too close together but so that their movements confuse the defensive players. When a receiver is alone on one side of the field, however, the coach can select from a much larger set of movements for the player because a direct conflict with another teammate is less likely to occur. Conversely, as more agents are added to the scene, the desired behaviors of any given agent will need to be adjusted locally in space and time more frequently to avoid the larger number of obstacles created by other agents. For example, if an 11-person offense runs a play against a 1-person defense, every player except one will probably execute the "ideal" motion scripted by the coach. Against an 11-person defense, however, most offensive players will need to adjust their motion as they interact with the defensive players. A "straight-line" motion will become significantly more variable as more agents are added to a scene.

Relative geometric spatial and temporal knowledge. Some single-agent gestures can be recognized using geometric matching techniques (e.g. see [DB97] for one example). Even recognizing an action as simple as "pointing," however, requires geometric reasoning about spatial relationships.[5] Actions in the football domain are defined by relative temporal relationships as well as spatial relationships. For example, one

[5]Pointing is defined as follows: "To indicate the position or direction of something" [Woo81]. Therefore, while it is possible to detect a person pointing by checking for a particular body and arm configuration [MPB+94, WB98], a more appropriate detector must detect that the person's arm is actually oriented *towards* some object in the scene. Even this detector is actually inadequate because it has not detected that the person is *indicating*, or intending, to point to the object.

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." Two issues are as follows: what are the primitives that the temporal detectors like "before" are comparing, and what are the semantics of these temporal detectors?

Scalability. In this document the actions of interest are assumed to involve more than one agent but less than 30. Some multi-agent domains such as troop movement analysis, however, can have hundreds or thousands of agents [DR97]. Action recognition systems for one agent do not necessarily scale to scenes containing 2 agents, and action recognition systems for two agents may not be capable of representing systems with 11 agents. Representations for 11 agents, however, may not scale well to thousands of agents.

Intentionality and the difficulty of physical modeling. In the computer vision community, single-agent or 2-3 agent actions are often modeled using systems that reason about dynamic and kinematic physical properties (e.g. [MJS96, RK94]). In domains with multiple agents, particularly when the agents are sentient and interacting in complex ways, modeling the action using *only* models of physical interaction of agents is computationally impossible and does not necessarily lead to improved recognition systems. For example, in the football domain, the action of each agent on the field *cannot* be explained using physical modeling alone. Explanation of the movement 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. In all three example domains – vehicle, kitchen, and football monitoring – the agent behavior cannot be fully described using only physical models of interaction because the agents are interacting in an *intentional* way. Intentionality will be more precisely defined in Section 2.4. The behavior of an agent depends upon its external environment, physical interaction with that environment, and pre-existing, history, and mental models of activity. Issues related to the recognition of such "intentional action" are discussed further in Section 2.4.

Availability of evidence. In addition to an exponential explosion in the number of features, some features may not be obtainable in certain contexts. Evidence is not always computable, and sometimes when it is it provides no discriminatory power. For instance, in some plays there will be no RFL; therefore, the distance between this player and the QB cannot be computed. Even when this information is available for a play, however, it may provide absolutely no useful evidence indicating or counter-indicating some play.

Contextual label interdependencies. In static object recognition, some objects in some imagery are difficult to recognize without considering the image of the object *and* surrounding objects. For example, many bridges viewed from an overhead camera will appear nearly identical to a segment of road, except that the bridge is flanked by water [SF91]. Analogously, in multi-agent domains, many actions or labels are impossible to determine without jointly considering multiple agents. For example, in football, the label of a player at the start of a play is defined by the person's position relative to other players and can depend upon how players far across the field are standing.

Representational uncertainty. Actions have three types of representational uncertainty. The first is that the temporal constraints of the action description itself are typically probabilistic (e.g. B follows A, but only 80% of the time). Actions cannot be fully described by necessary or sufficient conditions but rather by typical components. The second source of uncertainty is the fuzziness of attributes that must be assigned to individual elements (e.g. obj1 is *near* obj2). Finally, the third consideration is the probabilistic nature of perceptual sensing devices. Often, perceptual evidence, especially visual evidence, can be either missed or hallucinated.

Noisy trajectory data. In a multi-agent domain, the features with the most discriminatory power are relational features that compare the states of two or more objects; therefore, noise in one agent's trajectory can propagate to many features.

As discussed later in this thesis, these multi-agent criteria make it difficult to apply single-agent action recognition methods directly to recognize multi-agent collaborative activity. Those representations were not designed to model situations when *non-geometric spatial and temporal relationships* must be represented and recognized from trajectory data.

2.3 Characteristics of football team action

Using the football domain as an example domain influences the representation that is developed in this work. Therefore, it is worthwhile to describe some additional properties of multi-agent action recognition that are unique to this domain. These are listed below.

Infrequent collaborative re-planning. The actions of the offense in a football play are scripted by the coach prior to the play action. The agents do adjust their individual plans during the play in reaction to the actions of other agents, but major plan changes in order to accommodate some team goal are infrequent. The resulting behavior can be viewed as parallel event streams, one for each agent, that interact in pre-planned temporal and spatial ways. Locally, however, each player is continuously adjusting his motion to respond to the defensive agents. For some tasks in other domains (e.g. recognizing cooking oatmeal), each agent may make large adjustments to a long-term plan in response to the actions of other agents and with the explicit intent to facilitate the collaborative action.

Agent based goal descriptions. The actions to be recognized can be naturally described based on the *relationship between goals of individual agents*. Examples of agent goals in the football domain are catchPass, block(defensiveObject), and receiveHandoff. Each goal action for a particular agent can be recognized partly from direct visual evidence related to that agent (e.g. the agent just received the ball for a handoff) and partly from contextual evidence established by other agents (e.g. the agent could have just received a handoff because the quarterback has not thrown the ball and the play has started already). Collaborative goal actions can then be constructed by specifying temporal and logical relationships between agent goals. A team's football play, for example, consists of primitive goal components for individual agents like "the quarterback player will receive the ball via a hike action," then "the quarterback player will have the goal to move back to a particular position at the same time that the receiver players have the goal to run particular motion patterns downfield," then "the quarterback will have the goal to throw the ball to a receiver player," then "a particular receiver will have the goal to catch the ball." In short, a multi-agent action description can be described using many goal-like primitive components assigned to particular agents that are partially ordered in time and that obey certain logical constraints. This assumption may not be true for some multi-agent actions. The cooking oatmeal example, for instance, may require many more if-then contingencies (e.g. "if there is oatmeal, then ...", "if the other agent heats the water, then...") that not only temporally tie the actions of one agent to the other but that also result in the actions of one agent changing the actions that other agent will perform.

Existing taxonomies. Coaches and fans have developed semantically meaningful football annotation taxonomies that are used in this work. Consequently, the recognition approach presented can be evaluated with respect to the *entire* taxonomy, including those descriptions which are beyond the representational and/or recognition abilities of the proposed system. The labels describe both low-level, agent-based actions like "blocking," "intercepting," and "running through a hole" and high-level actions like "executing an R34 play." In short, in this domain at a given level of abstraction, most experts will agree on terminology.

No statistical databases. In the football domain, there are no existing large statistical databases with labeled actions that can be used to generate statistical models. Acquiring such databases is beyond the scope of this (and possibly any) project. Only a small set of examples is available for any given high-level description, such as a description of a play. Instead, a linguistic description provided by a domain expert, the coach, is available. The scarcity of a large, labeled dataset and the availability of a linguistic description of the action is a situation common to many real-world domains of interest.

Structured but uncertain. The group activity in a football play is both highly structured and highly variable. The offensive team always begins each play segment by trying to achieve a particular set of coordinated goal actions that can succinctly be described by a football coach. The system can attempt to recognize this activity. Typically, however, the defense prevents the team from achieving the plan exactly as described. This leads to significant variability in how individual players move, how individual goals are achieved, and the visual pattern of each particular play action. The multi-agent action taxonomy in some other interesting multi-agent domains is less well-defined. For example, there is probably greater variation in the order and the way that a single family could make oatmeal, which a representation for "cooking oatmeal" would need to encode [Pin99]. Although there is clearly structure to this type of activity, the structure permits more variation in the order of certain actions and the agents performing some actions than may be true for a coach's football play. A coach or fan can easily write down what is supposed to happen in a given play.

Reset point. In the football domain, the global and agent context can be initialized at the beginning of each play when the teams line up in their starting formations. This is a property that is not typical of other domains and simplifies the current task.

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. Although most behavior in everyday life has some reasonable explanation, the reasons underlying the behavior can often be more difficult to discern than they are in a football play (e.g. someone having a bad day cuts off a driver or slams the oatmeal box on the counter). This purposeful behavior results because a team has a limited set of resources that it is trying to fully maximize. Typically, a player is either explicitly creating a deception via his action or contributing some key component in the play. Occasionally, players are doing neither because they are confused, but the reason for the player's confusion is usually apparent to the observer.

Some of the mentioned football domain properties, such as adversarial action, make the domain a particularly challenging action recognition problem. Conversely, other football-specific properties, such as availability of a reset point, simplify the problem considerably.

2.4 Recognition of intentional action

The framework for action recognition proposed in this work will be discussed with respect to the multi-agent action characteristics and football domain characteristics in Chapter 7. However, one characteristic, the relationship between recognizing collaborative action and representing intentionality, is addressed in this section.

The previous sections stated that when multiple agents in a rich environment are sentient, their interaction cannot be fully modeled using physical rules of interaction because the agents will often be making "intentional" decisions. This raises the following two issues: what is intentionality and how does it affect a model for action recognition? In the multi-agent domain, when agents are collaborating, agents are responding to both individual and joint mental plans of activity.

2.4.1 Defining intentionality

"Intentionality" is a poorly defined word in both English and technical vocabularies. Philosophers and AI researchers do not agree on a single definition or theory of intentionality (e.g. see [Bra90, Hob90, Pol90a]). Most computational models of activity, however, define intentionality with respect to *agents*, *beliefs*, *plan recipes*, and *plans*. A belief is some stored knowledge about the world. A plan is a hierarchical construction of goals and sub-goals, when a goal is the desired achievement of some belief. As Pollack pointed out, "There are plans and there are plans" [Pol90a]. The first type of plan is one an agent knows – these are typically called "recipes." The second type of plan is the plan actually being executed. The distinction between these two types of plans is between *knowing* and *knowing and doing*. Routine behaviors are pre-compiled plans. An agent is continuously modifying its plan based upon its perception of the environment. Agents select and modify pre-compiled plan recipes so that executing the plan recipes will achieve the plan goals.

In multi-agent domains, each agent may be responding to its environment intentionally (i.e. using an internal plan to achieve certain beliefs). The behavior of agents, however, can also be affected by *joint intentions*. Joint intentionality can be defined as agents having both individual beliefs and *mutual beliefs*. Mutual beliefs are beliefs that are held with other agents [GK96]. Computational models of joint intentionality specify how agents maintain this mutual belief by *communication of intention* (or, in other words, the communication of partial plans) between themselves. Models of joint intentionality are used to model the interaction between agents engaged in collaborative (also known as shared cooperative) activities [Bra90].

A distinction can be made between *coordinated* and *collaborative* activity. For example, drivers can adeptly[6] maneuver around each other at high speeds, each guided by the same set of rules that determine how a vehicle should react to the environment. The rules have been explicitly designed to result in *coordinated* driving behavior. Drivers, however, also engage in *collaborative* behavior, when multiple drivers have a shared plan to jointly commit to the activity and communicate to achieve shared goals. One such collaborative activity is driving in convoys or caravans [CL91], when both the leader and follower will modify behavior to ensure that the other can achieve the goal of following. Agents are committed to modifying the plan in tandem if necessary to achieve some shared goal. Coordinating these changes means each agent is reasoning about the belief and mutual belief of the other agents.

A football play is a collaborative activity because players will adjust their individual goals in tandem in order to achieve a team goal. 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

[6] Some cities excluded.

way (e.g. the ball carrier pointing to the defensive player that he wants his teammate to block).

2.4.2 Using intentional models of activity for recognition

Given that football plays are collaborative actions with agents having joint intentions, it is reasonable to investigate whether theories of joint intentionality can be used to model the action for recognition.

One comprehensive theory of intentionality reasoning for domains with collaborating agents is the SharedPlans model [GK96]. This model is described in Appendix B and partially applied to an example from the football domain. The model reasons about activity using modal, agent-based belief and mutual-belief primitives that are derived by recursively expanding high-level properties of the collaborative activity.

Application of the SharedPlans formalism to the football example was hindered by several issues, as discussed in Appendix B. For example, the model, which was developed for analyzing discourse, implicitly assumes that agent communication is explicit and easily detected; however, in the football domain, some agent communication takes place through *agent perception of the shared environment* instead of by explicit (and detectable) cues like speech. One problem, therefore, is that reasoning about visual communication requires that a system make inferences about the visual perceptual abilities of individual agents. For example, in the football domain, the QB may have the goal to throw the ball to the LSE, who runs downfield ten yards. The QB, however, will adjust his plan if he observes that the LSE has slipped and fallen after running three yards. In this case, interpreting the QB's actions may require making assumptions that the QB observed that the LSE had fallen. Even from the original video signal, which contains more information than the data trajectories used in this work, these inferences can be difficult for a person, let alone a computer system.

Other problems of applying the SharedPlans formalism to the football example discussed in Appendix B are summarized here. First, in the football domain, all evidence for beliefs about the world are computed from noisy data and sensors, but the intentional model has no mechanism for propagating the resulting uncertainty. Second, the SharedPlans model does not provide insight into how the system can recognize visually primitive, non-intentional actions (or even what these primitive actions should be) and use those results to "bootstrap" the SharedPlans belief reasoning. Third, a mechanism is not provided for recovering from (or detecting) situations when agents have erroneously perceived some situation and therefore reacted in an anomalous but explainable way. Finally, the SharedPlans representation is computationally intractable for most real-world problems, especially given that it lacks any mechanism for representing and propagating certainty of belief and observations. The SharedPlans model, like other joint intentionality models, has been designed for application in the discourse domain, and therefore, the model makes some assumptions about the reliability and detectability of input data that are unrealistic for visual tasks like football play recognition.

2.4.3 "What things are" vs. "What things look like"

In summary, the SharedPlans model and related models for propagating joint intentionality do not provide tools for recognition of collaborative action. In fact, the methods presuppose the availability of detectors for cues of collaborative action that subsequently bias a logical inference process towards correct interpretations.

The focus of the work presented here is on developing a representation of collaboration that is sufficient to recognize some types of typical collaborative data from noisy data – without making any assumptions about the availability of particular "intentional" feature detectors. Therefore, it is useful to consider how a recognition system without a perfect model of intentional behavior might interpret the intentional behavior that it perceives.

Dennett [Den87] has proposed three different "stances" that an agent (or a recognition system) might use for interpretation of some observed activity. The first stance, the *physical stance*, is adopted when one agent can predict the operation of a second agent at every physical level. For example, a system predicting the behavior of an airborne football can use laws of physics to accurately predict the ball's behavior because the ball's reaction to its environment is well understood. Of course, the observer agent must have the ability to measure the relevant properties of the environment. To precisely determine the path of a football, a computer would need to measure the speed and rotation of the ball, wind velocity, and even factors like the field altitude and current temperature.

The second stance is the *design stance*. Here, "one ignores the actual (possibly messy) details of the physical constitution of an object, and, on the assumption that it has a certain design, predicts that it will behave *as it is designed to behave* under various circumstances" [Den87]. For example, using the design stance, a system might predict that a football will fly straight when it has been thrown by a competent quarterback, unless someone else touches the ball. "Only the designed behavior or a system is predictable from the design stance, of course. If you want to predict the behavior of [an object when] it is pumped full of liquid helium, revert to the physical stance" [Den87]. The design stance trades off the ability to explain any anomalous observed behavior for an interpretation of activity that may permit a simpler and more efficient computational model that can describe typical activity.

When the design and physical stances are not accessible, an agent resorts to an *intentional stance*. Dennett describes an agent's reasoning: "First you decide to treat the object whose behavior is to be predicted as a rational agent; then you figure out what beliefs that agent ought to have, given its place in the world and its purpose. Then you figure out what desires it ought to have, on the same considerations, and finally you predict that this rational agent will act to further its goals in the light of its beliefs. A little practical reasoning from the chosen set of beliefs and desires will in many – but not all – instances yield a decision about what the agent ought to do; that is what you predict the agent *will* do" [Den87]. In addition to reasoning about what an agent physically can do and what an agent is designed to do, here an observer reasons about the beliefs of the agents in the scene.

In short, any object can be described with an intentional stance. A light switch can be a device that "transmits current when it believes that we want it transmitted." This belief-based description, however, seems ridiculous because it does not "buy us anything" over a standard mechanistic description [Sho90]. The number of variables that affect the behavior of the light switch is relatively small; therefore, a first-order logic or probabilistic representation can straightforwardly model the switch's behavior accurately and completely using a design stance.

The intentional stance *is* useful in multi-agent domains like football play recognition because explaining some player behavior requires explicit reasoning about the belief and mutual belief of the players over time (see Appendix B). This work, however, 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. Instead, recognition of some multi-agent action can be accomplished using a probabilistic design stance. In some cases, the intractability of modal logic intentionality reasoning systems can be avoided by using a representation that describes complex activity using models of collaborative action that detect patterns of agent coordination using small number of temporal and logical connectors linking *probabilistic* visual components. The representation detects collaborative activity using a representation in which typical collaborative interaction has been "compiled down" into coordinated action. This representation is described in Chapter 6.

Some multi-agent recognition tasks will require explicit reasoning about intentionality, such as explaining how players react to another player who has fallen (see the example in Section 2.4.2). Most likely, however, a recognition system that does use an intentional model of activity will require preprocessing algorithms that can prune the space of possible actions being observed at any time by identifying typical examples of collaborative action without using intentional reasoning. Making intentional models like SharedPlans tractable for real problems with perceptual input will require some initialization. Here, initialization consists of "compiling down" collaborative activity to a set of relationships encoding typical agent coordination during a multi-agent collaborative activity. The approach trades off the ability to reason about complex intentional interaction for computational tractability in domains with noisy evidence detectors.

The model developed in this work is designed to use noisy perceptual data to recognize multi-agent coordinated action, but 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. For some interesting domains, however, visual detection of typical, multi-agent coordinated activity enables interesting, useful new applications. This dissertation develops one method for recognizing typical multi-agent action from noisy data.

Chapter 3

Data and data acquisition

This chapter describes the football play data used to test the recognition system, the method used to acquire the data, and some characteristics of the data.[1]

3.1 The play database

In order to study play recognition, a database of video data and the corresponding play labels was required. The New England Patriots football team provided an entire season of tapes of every offensive play they ran in an official game (1132 plays) and the name of the play they *intended* to run.[2] Of the full play database, 872 example plays (1.5 hours of video) has been transferred to an analog video laserdisk and has been sorted by play label. The laserdisk contains an average of six example plays for each of 133 types of Patriots' plays.

The coach of the 1993 Patriots withheld the 1993 playbook. Therefore, it has been necessary to "reverse engineer" the meaning of each play label by carefully viewing multiple examples and comparing sets of play examples. Models for plays with only a few examples can typically be reverse-engineered using other examples that are known variations on the particular play. For example, the p51curl and the p52curl plays are identical except for a vertical flip. Therefore, examples of both play types could be combined to determine the description for the ideal p51curl and the ideal p52curl.

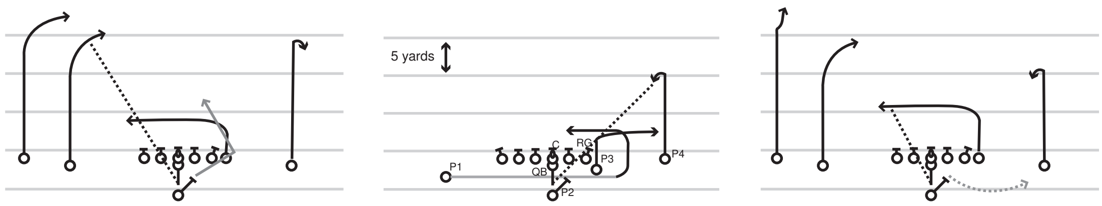
The description recovered from this comparison of play examples consists of the following:

- the ideal paths of individual players
- temporal relationships between the movements of some offensive players
- desired space-time relationships between some offensive players and nearby defensive players
- the primary and secondary path of the ball
- allowable variations in starting formation positions
- optional moves available to particular players as the play progresses.

This information can be encoded in diagrams like the p51curl diagram in Figure 2-2. Some information that is likely to be encoded in the original playbook has not been recovered from the examples. For example, most professional football plays consist of highly specialized blocking responsibilities for individual players that depend upon the exact configuration of

[1] The data used for this project, as well as some additional football-related image, video, and trajectory data, is freely available to researchers by sending a request via email to intille@media.mit.edu. Some of the data has been annotated.

[2] Unfortunately, the database is from the Patriots' 1993 season, when an unusually large percentage of the plays they intended to run were not completed successfully!



(a) (b) (c)

Figure 3-1: Three New England Patriots' play diagrams for examples used in this work: (a) p52maxpin, (b) p51curl, and (c) p56yunder.

the defense and sometimes on communication among the offensive blocking players prior to running the play. This type of information has not been extracted from the examples and is not used by the system described in this work. Three play diagrams for some of the example plays used in this work are shown in Figure 3-1. More diagrams appear in Appendix C.

3.2 Obtaining object trajectories

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.

There are three methods by which such object trajectories can be obtained: automatic visual tracking, tagged tracking, and manual tracking.

3.2.1 Automatic visual object tracking

A multi-object tracking system developed previously can track approximately half the players in a typical play from the football play database. The tracking method uses local "closed-worlds" of knowledge centered around each tracked object to adaptively select the most appropriate features and algorithms to use for tracking given the local space-time context [Int94, IB95]. The tracking algorithm handles non-rigid tracking of objects with erratic motion, even when the objects are frequently interacting and the imagery available is from a pan/zoom camera with relatively low resolution. The algorithm can track players well, except when large groups (i.e. >3 players) interact so that they are visually adjacent or occluding one another. Typically, about half the players on the field clump into such groups at the start of the play, prohibiting closed-world tracking. Figure 3-2 shows two automatically recovered trajectory images where approximately half of the players on the field have been tracked.

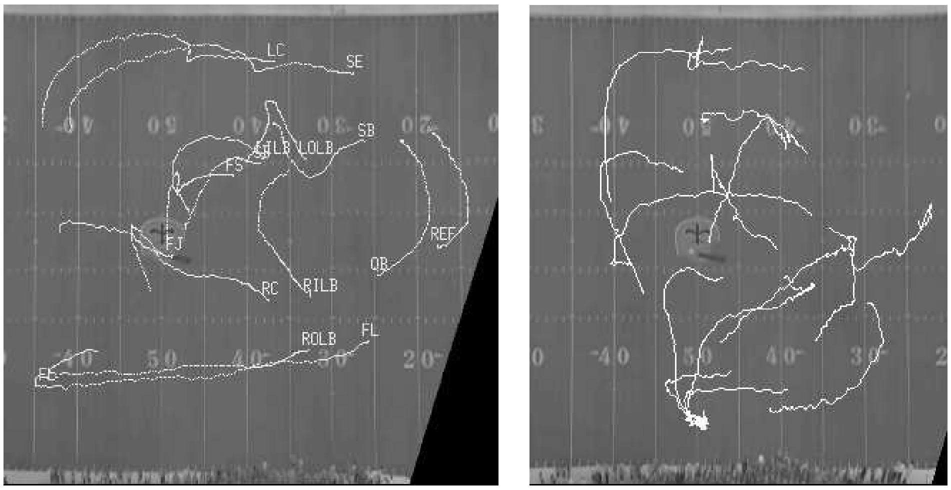


Figure 3-2: All recovered paths from a particular play using the "closed-world" tracking algorithm described in [IB95].

Figure 3-3: Partial frame of video containing players colliding with one another.

A few other tracking systems have been developed specifically focused on tracking video of players and the ball in sporting events with characteristics similar to those of the football video (i.e. rapid motion, non-rigidity, low-resolution imagery, collision and interaction). They use color or correlation templates [CSKH97, GSC$^+$95, YYYL95, KSH98] or color histograms of blobs [KYA94]. No fully automated techniques are known that are capable of robustly tracking all (or nearly all) the players on the field successfully given video of plays in the Patriot database. Figure 3-3 shows a portion of a single frame from a football game. The image contains colliding players that are visually difficult to differentiate and problematic for existing tracking systems.

3.2.2 Tagged tracking systems

To overcome the limitations of fully automated visual tracking, semi-automatic tracking systems are being developed for commercial sports broadcasting video highlighting (e.g. [Ora99]). Such systems are not easily adapted to provide trajectories for *all* the players in a scene.

The tracking problem is easily solved in the sports domain and in other useful domains (e.g. the home and office) by adding electronic tags to objects of interest. Tagged uniforms and sporting equipment will become commonplace in the next few years because the recovered trajectory data can be used to digitally enhance sports broadcasts (e.g. Fox's SuperPuck [Sup]). For instance, a tagged system using triangulated RF transmitters is being developed explicitly for professional football games that will provide tracking data for the players (and possibly ball) with an accuracy of 1 centimeter; it will be in use by the National Football League by 2001 [Tra99]. The system described in this document is capable of using such data. Further, inexpensive and wireless tagging systems for everyday household devices are currently being designed in order to enable new interactive applications in the home and office [Ger99]. These tagging systems are likely to make it easier to obtain the type of trajectory data used in this work in everyday domains for many objects of interest.

3.2.3 Manual data acquisition

In this work, manually acquired trajectory data for all players on the field is used. An interactive graphical interface was developed that allows a user to follow objects with the mouse pointer on the screen as the video plays slowly from the laserdisk. The position of the mouse is recorded in each frame. The user views the play once for each player, following the player with the mouse. Prior to tracking, the user marks the object being tracked with a football position label (e.g. QB, C, etc.) The user then tracks points on the field, ensuring that every frame has at least four tracked points. Finally, the user again plays the video once for each player and uses the mouse to indicate each player's upper-body orientation. Figure 3-4 shows a screenshot of the manual tracking software where some player objects have been tracked.

Once all the players, field points, and player orientations have been tracked with the mouse, a least-squares matching process is used to convert the tracked player points into a common field coordinate system using the tracked field points and a model of a football field.[3] This rectification process assumes that all the player points are in the plane defined by the field points.[4]

[3] Although it is possible to use the grid structure of a football field to automatically stabilize the imagery [Int94], these techniques are not robust to muddy fields, rainy days, and fans blocking video cameras, and therefore, they have not been used in this work.

[4] The person tracking is therefore told to track the *feet* of the player, which are assumed to be touching the field plane.

Figure 3-4: A screenshot of the GUI interface used to manually track players and field objects by following objects with the mouse pointer as the video plays.

Acquiring the trajectory data is a time-consuming and tedious process. Tracking a single 9 second play carefully requires upwards of 2.7 hours of mouse work and much longer when (necessary) rest breaks are included. Currently 27 plays have been fully tracked.

Only plays with more than four examples were selected for tracking off the laserdisk. Otherwise, the class of plays currently used were considered randomly, with a slight emphasis towards running plays. Within each class, some effort was made to track multiple examples.[5] In this work, "special teams" plays like punts, kickoffs, and extra points will not be considered in order to maintain a manageable level of football knowledge engineering.

The three chalkboard images of a p51curl play in Figure 1-1 were obtained using this manual tracking system. Figure 3-5 shows two single frames of video with the player labels marking each players position and a line indicating approximate upper-body orientation. The field points used for transforming image points to the field coordinate system are also displayed.

[5] Plays in the current database were selected somewhat haphazardly because initially a set of at least 150 plays were slated to be tracked and the first plays tracked were simply the first on the list. However, tracking turned out to be more time-consuming than expected, which prevented the full set from being completed.

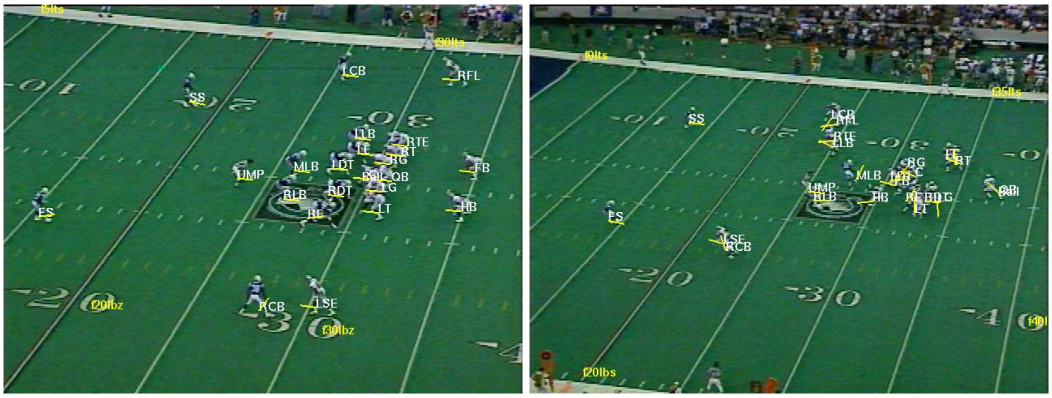


Figure 3-5: Player position and upper-body orientation data overlaid on two frames of video from a play example. At least four field points must also be tracked for each frame.

3.2.4 Characterizing trajectory noise

The manually acquired trajectories are noisy due to four types of systematic errors: placement errors, center-of-mass errors, occlusion errors, and orientation imprecision.

Placement errors occur when human trackers fail to accurately mark the position of points due to fatigue. The position entered via the mouse pointer can often be displaced by 3 or more pixels in the x and y direction from the ideal image position. Placement errors for tracked field points are especially problematic because those points are used to transform the player trajectories to the field coordinate system. Therefore, any error in field point tracking propagates to all tracked objects. Consequently, field points are typically tracked more slowly and carefully (e.g. about 1 frame per second) than player points (e.g. 2-3 frames per second). Figure 3-6a shows an approximately 9 pixel error when tracking the ROLB player.

Center of mass errors result from the difficulty of estimating an object's center of mass as it projects onto the field. Only points that are in the plane of the field will be properly transformed to the field coordinate system. Therefore, the person tracking attempts to mark each player at the center of mass between the feet. The rapid motion of the players and their animated body movement sometimes makes estimation of this point difficult. For example, players often jump out of the ground plane, making it difficult to estimate where the center of mass projects on the field. Figure 3-6b shows a situation where a player has fallen and where the best point to mark is ambiguous. Typically, the feet are marked in this situation.

Ball tracking is handled by a separate interface component. When a player holds the ball, the person tracking marks the feet of the player holding the ball as the ball position. This position is marked independently of the player position, so the ball position rarely perfectly matches the position of the player carrying the ball. When the ball is thrown and is far from the ground plane, the ball's correct projection onto the field is difficult to estimate. Therefore, the person tracking specially marks the frame where the ball is thrown and then specially marks either the frame when the ball is caught or the frame when the ball hits a player or the ground. The position of the ball is then linearly interpolated for frames when the ball is in flight. Similarly, if a ball is bouncing on the ground, the ball position is marked each time the ball hits the field and interpolated when it is in the air.

Occlusion errors occur when a player is temporally occluded by another player or by external factors such as an arm thrust before the video camera, as shown in Figure 3-6c. In these situations, the person tracking has been instructed to smoothly guess the position of the object until it is visible again.

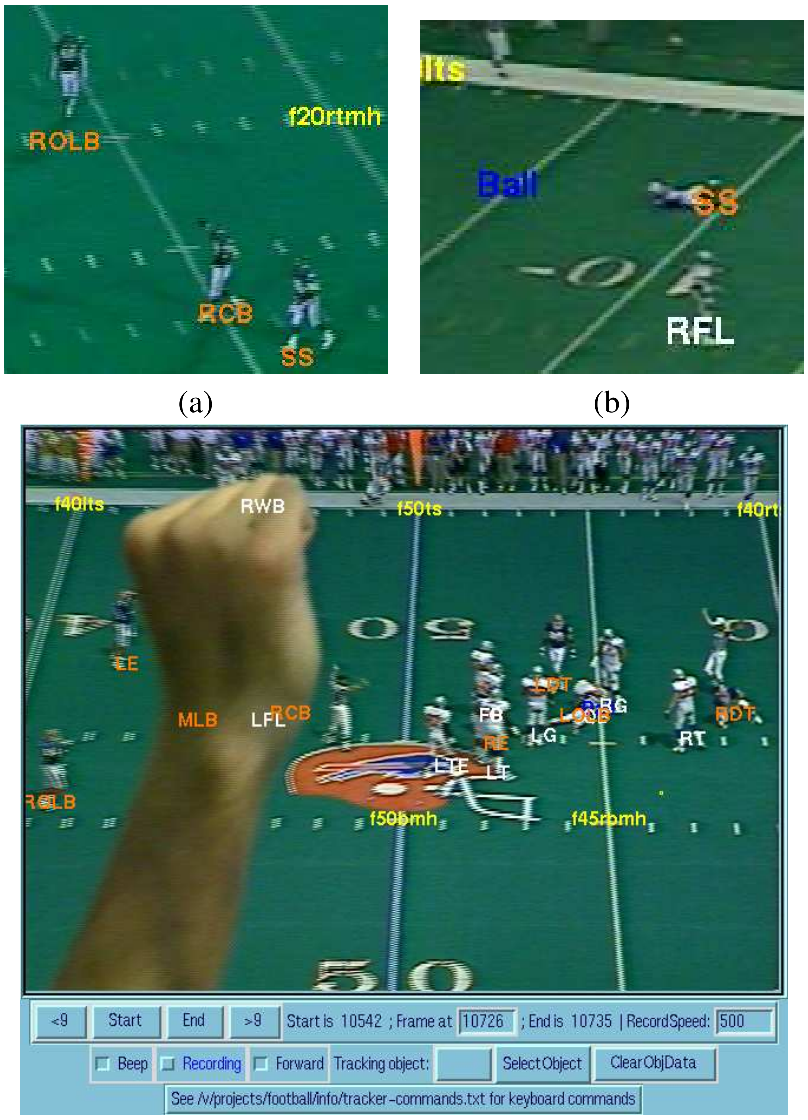
Estimating upper body orientation is inherently imprecise. Often the torso faces one direction and the head faces another. In these cases, the person tracking will indicate some compromise orientation. The orientation itself is indicated simply by placing the mouse on the video on the field in the direction of orientation.

Systematic errors present in the data are easily seen when the data is graphically displayed in the field coordinate system and played at full frame rate. One such frame is shown in Figure 3-7.[6] The errors are difficult to characterize once the play has started. Analysis of several plays containing segments where players are standing still in the original video, however, has demonstrated that the standard deviation of a tracked object's position from its actual position is, at best, .35 yards (.32 meters) in any frame. The positional accuracy is certain to be worse as the play begins and the camera view undergoes rapid panning and zooming, resulting in more human tracking error.

Finally, although the cameraperson tries to keep all players in view for the majority of the play action, near the end of a play the camera is zoomed in on the area of the field where the ball has been carried or thrown. Some players, therefore, drop out of the field of view. In these cases, player trajectory data simply ends when a player is no longer visible.

In summary, the data used in this work are the trajectories of all moving objects given by (x,y,orientation,label) tuples as a function of the time. Orientation denotes the approximate upper-body orientation of a player, and the label is the name of the player's starting position. The player and the ball trajectories are provided. This data was obtained by manual tracking of objects in video, but the data is noisy due to human error and imprecision.

[6] A few QuickTime videos of player trajectories using this graphical representation are available at http://vismod.www.media.mit.edu/vismod/demos/football/.



(a) (b)

(c)

Figure 3-6: (a) Placement errors result from the difficulty of obtaining perfect mouse tracking when following players with a mouse in video. Here, the ROLB label is about 9 pixels below the ideal position. (b) Center of mass errors result because it is sometimes difficult to estimate where the center of mass of a player is in the plane of the field as players run, jump, and fall. Here, a player has fallen and the individual tracking has been instructed to track the player's feet. (c) Occlusion errors result when views of some players are blocked by other players or by factors external to the play itself.

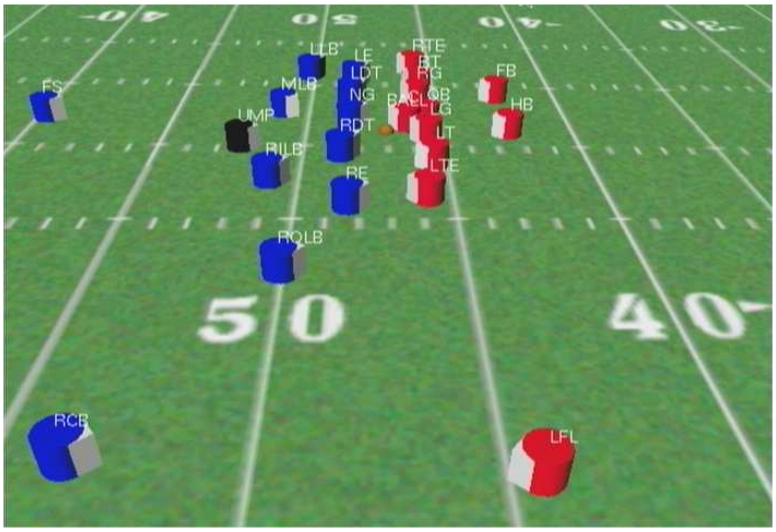


Figure 3-7: The trajectories of the players, ball, and markers on the field are used to rectify the trajectories to a common field coordinate system. This image shows one frame of a graphical representation of the play as trajectories. Each player's position is marked by a cylinder that has a small arrow indicating that player's approximate upper-body orientation.

Chapter 4

A test system: labeling a multi-agent scene

As discussed in Chapter 2, the complexity of the multi-agent recognition task results from, in large part, the *interaction* between agents as they move. The more agents there are in a scene, the more ways there are to measure different types of interaction between agent actions.

This chapter examines an analogous non-temporal problem – labeling of static, multi-agent scenes. The complexity of this task results from the large number of spatial relationships between agents. The more agents there are in the scene, the more ways there are to measure the spatial configurations between particular agents. Given a large possible feature set of spatial relationships, the task is to assign labels to objects even though any given label depends upon other labels that may or may not be assigned.

In particular, the task is to develop an algorithm that, given the (x, y) location of objects on the football field at the start of a play, can determine the "player label" of each object. For example, the algorithm must determine which object should be labeled "QB" (quarterback), which should be labeled "C" (center), etc. The algorithm described in this chapter uses a rule set, specifying allowable relationships between objects. Using these rules, the algorithm searches for a labeling of objects in the scene such that each label is *consistent* with all the other labels in the scene given the rule base.

The algorithm is based upon an approach developed for the recognition of natural objects in static scenes. Natural objects like trees, shrubs, grass, and clouds are difficult to define using the precise, geometric models often used to model man-made objects such as vehicles and machined parts. Therefore, one strategy for recognizing a natural object is to find an interpretation of a scene – a labeling of objects – such that each object's label is consistent with the labels of all adjacent objects [SF91]. Consistency is defined by rules encoding domain knowledge. This approach appears to be applicable to multi-agent scenes when the labels of individual agents depend upon the labels of other nearby agents. For example, the QB player label cannot be assigned without assigning the label for the C, but the player label for the C cannot be assigned without considering the label for the QB.

The formation-labeling algorithm described in this chapter overcomes the multi-agent labeling interdependency. Problems, however, were encountered when trying to extend the approach to the dynamic problem of recognizing football plays. These representational inadequacies motivated the development of a fundamentally different algorithm for play recognition, which is discussed in Chapter 6. That system exploits one observation made in this chapter that a large number of simple consistency checks can be used to generate a good, multi-agent labeling hypothesis.

4.1 Multi-agent scenes and context

The problem with the multi-agent formation labeling task is that objects are defined less by what they look like than by what other objects are around them and how those other objects are configured. Recognizing an object's label requires finding objects with particular attributes set in a *specific context* established by surrounding objects. This context-labeling problem is similar to the problem of recognizing natural objects.

Strat and Fischler recognized that objects in outdoor scenes are often difficult to model using geometric models [SF91]. As a result, a recognition system trying to identify some object is forced to use more information about how that object *relates to other objects* in the scene (e.g. a shrub will almost always touch the ground, the QB always stands somewhere behind the C). Analogously, a recognition system for multi-agent action exploits how one agent's actions *relate to the actions of other agents* because these relationships may be the most discriminating features for determining the action's label. Some objects and actions are difficult to define independently of the other objects or agents in the scene.

Strat and Fischler, therefore, developed an object recognition algorithm with the goal of avoiding the two main assumptions made by object recognition systems for geometric objects: (1) that all objects can be defined by a small number of explicit shape models, and (2) all objects have characteristic features that are reliable identifiers [SF91]. The second assumption is made by some object recognition work when highly-characteristic "focus features," which are assumed to always be present on an object, are used to rapidly prune object search spaces [BC82]. Powerful focus features are more difficult to identify for natural objects because most focus features are defined based upon view-invariant, geometric edge intersections that are not found on many natural objects. Natural objects like shrubs can vary substantially in shape, color, and texture from image to image, but their relationships to other objects may be more invariant. For example, a tree has a region of sky near the top. Unfortunately, the way that the sky appears can depend upon the weather and upon the viewing angle. Neither the tree or sky has visual properties that are invariant over time and imaging situations. The *relationship*, however, between the tree and sky will nearly always hold true.

Strat and Fischler use explicit reasoning about multi-object context to minimize their recognition algorithm's dependence on stored geometric models; instead, the algorithm is dependent primarily on the detection of important relationships between objects. They use contextual rules to simultaneously assign labels to natural objects in a scene and to partition an image into semantically meaningful regions that are self-consistent given the rule set. The strategy is to use conservative feature detectors, significant redundancy in the representation of objects, and (primarily pairwise) contextual consistency rules to achieve robust recognition of natural objects; this is done by enforcing that relationships between hypothesized labeled objects be valid given a set of rules. The problem is that the mutual dependency of object labels requires a (typically iterative) relaxation search process.

4.2 A formation labeling test system

Strat and Fischler developed a hypothesize and test algorithm for region partitioning and object labeling of natural outdoor scenes of ground, sky, trees, and shrubs. The system, CONDOR, searches for a consistent interpretation of a scene using three sets of context rules [SF91]. The system presented below is based directly on this work.

Given the *prima facie* similarity between natural object recognition and action recognition, a context rule labeling system for football formations was developed. The design intentions were to test the Strat and Fischler system on a static problem in the football domain and then extend the system to consider temporal features and recognize multi-agent actions.

To develop the formation-labeling system, each player's (x, y) position (but not orientation) in the first frame was used from the trajectory play data described in Chapter 3. The data is noisy due to the issues discussed in Chapter 3. Although the LG will always be slightly behind and to the left of the C (with respect to the LOS) in the actual scene, in the formation data the LG may actually be in front of the C. Similarly, some formation labels depend upon knowing whether a given player is next to the LOS (i.e. < 1 yard) or just off the LOS (i.e. > 1 yards). Discriminating these two cases, however, can be difficult due to rectification errors and the fact that the LOS position must be estimated from the player position data.[1] The data does not contain noise such as the addition of spurious player blobs or missing player blobs that might result from a system that visually detects player positions.

Figure 4-1 illustrates the formation labeling task. Assume that the (x, y) field positions of the 11 offensive players are given, as shown in Figure 4-1a. The goal is to determine a correct player label for each of the 11 offensive objects; one such interpretation is shown in Figure 4-1b. This is a context-sensitive problem because the player label of any given player can (and often does) depend upon the labels of several other players. For example, the object marked P11 in Figure 4-1a can only correctly be labeled as LSE (left split end) by knowing the player labels for P10 and P9. However, knowing P9 requires knowing the player label for P6, and knowing the player label for P6 requires knowing the label of P5, and so on. Noise complicates the problem. Figure 4-1c shows the original object positions with positional error bars drawn around each player's x and y axis typical of the data. The player's body (indicated by the circle) must be within the error bars, not just the player's center point. These small positional errors increase the number of valid interpretations and require a more extensive search during the labeling process. One of the incorrect possible interpretations is shown in Figure 4-1c; C has been incorrectly assigned to P4 instead of P6. Technically, the formation labeling shown in Figure 4-1c is correct because it is self-consistent given the rule set. When noise is considered, however, it is not the *best* interpretation of the data. This point will be discussed later in this chapter.

4.2.1 Searching for consistent interpretations

The goal for the labeling system is to compute a *consistent* position labeling for all of the objects in the data given a set of football position labeling rules. The algorithm uses conjunctive rules with the following form: $L : C_1 \wedge C_2 \wedge ... \wedge C_n \Rightarrow A$, where L is a class name, C_i is a boolean context condition, and A is some action to be taken. A class in

[1] The LOS is defined in Appendix A. It is an imaginary line splitting the offensive and defensive players. Multiple hypotheses for the LOS position may therefore be warranted if the positions of the players are noisy.

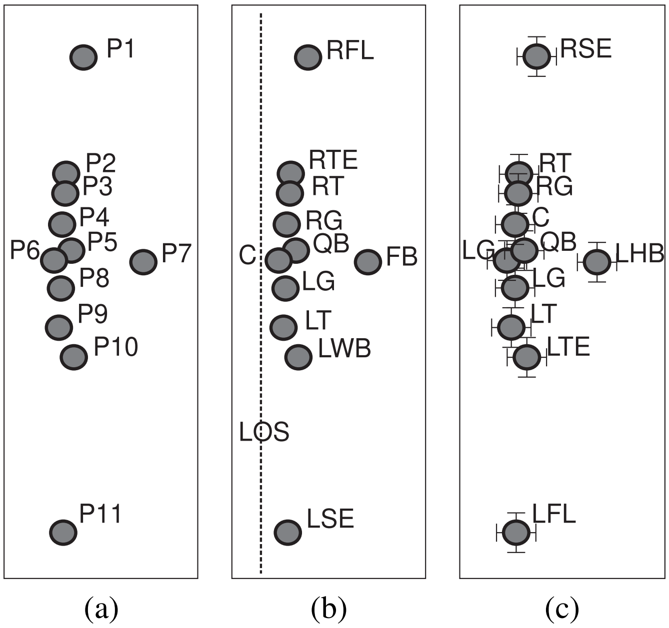


Figure 4-1: (a) Objects in a formation pattern to label with position labels. (b) One self-consistent player label assignment for the formation given the formation rules knowledge base. (c) One self-consistent player label assignment given noisy object positions, indicated by the error bars. The label names are defined in Appendix A.

this domain is a player label (e.g. QB, LT, C, etc.) or an intermediate system label (e.g. offensive-player-blob). An example rule is below:

$$\text{QB: exists(OBJ)}\wedge\text{leftOf(OBJ,RG)}\wedge\text{rightOf(OBJ,LG)}\Rightarrow\text{hyp(OBJ,QB)}$$

This rule encodes that some data object (OBJ) can be given a hypothesis candidate label of QB when there is an unlabeled data object that has an object to the left that has been labeled a possible LG and an object to the right that has been labeled a possible RG.

The algorithm uses a production, or blackboard, architecture. The results of applying rules are deposited in a common blackboard memory space, $\mathcal{B}$. Adding new information to $\mathcal{B}$ may satisfy the context condition $C_1 \wedge \ldots \wedge C_n$ of other rules and those rules will be activated. Some rules have context conditions that are satisfied merely by placing the input data on the blackboard; these rules initiate the reasoning process, adding information to $\mathcal{B}$ that triggers new rules, and so forth. Also stored on the blackboard is a set of mutually consistent labels of the scene data, called a clique, $\mathcal{C}$. The clique is constructed iteratively by checking that each new label added to the clique is consistent with all of the existing labels in the clique. The clique stores one partially labeled interpretation of the input data. Some context rules have context conditions that activate based upon labels in $\mathcal{C}$, and others deposit new information in $\mathcal{C}$.

```
(ISA (ob-candidate db-candidate los-candidate person-candidate)
     label-candidate)

(ISA (offense-candidate defense-candidate)
     person-candidate)

(ISA clineman-candidate
     lineman-candidate)

(ISA (c-candidate guard-candidate tackle-candidate)
     clineman-candidate)
```

Figure 4-2: A sample of object classification hierarchy rules used by the formation labeling algorithm. More rules can be found in Figure D-1.

The recognition algorithm uses a database of context rules for the domain of interest. Individually, the rules are relatively simple and activate only when specific contextual conditions are met. To assign a label to some object in a scene, the label must satisfy a large set of these relatively simple rules. The rules are used to iteratively label each object in a scene by constructing a labeling of the *entire* scene. Hundreds of rules check that each object label is consistent with all other object labels the system has proposed to describe the scene data.

Objects in the data can be labeled with multiple classes. The classes are hierarchically organized. Some examples are shown in Figure 4-2 (and more in Figure D-1). An object can be assigned a football label such as qb-candidate or c-candidate, but it can simultaneously receive the labels offense-candidate, person-candidate, and ob-candidate.[2] The context rule sets use the class hierarchies to simplify the encoding of rules that apply to multiple types of objects or labels.

The formation labeling algorithm consists of four stages of processing: hypothesis generation, hypothesis evaluation, grouping mutually consistent hypotheses into cliques, and selecting the best description from sets of mutually consistent hypothesis cliques. Each stage uses a particular set of context rules and is described below.

Hypothesis generation

The hypothesis generation stage assumes that it is easy to write hypothesis generation rules that work in specific contexts. These rules are designed to aggressively generate all remotely feasible label hypotheses given the observed data and other label hypotheses that have been made. The proposed label candidates are added to $\mathcal{B}$. Each generation rule has the format shown below.

[2] Ob-candidate is short for "offensive-blob-candidate", which represents any input data that could be an offensive player. Defensive player positions are used only to compute candidates for the line of scrimmage (LOS).

```
Candidate: los-candidate
Constraints: Clique: nil
             Object: db-candidate ob-candidate
             Funcs:  vertical-midline-players

Candidate: c-candidate
Constraints: Clique: los-candidate lg-candidate
             Object: ob-candidate
             Funcs:  (and (on-los OBJ los-candidate)
                          (same-side-los-p OBJ los-candidate lg-candidate)
                          (right-of-p OBJ lg-candidate los-candidate))

Candidate: c-candidate
Constraints: Clique: los-candidate rg-candidate
             Object: ob-candidate
             Funcs:  (and (on-los OBJ los-candidate)
                          (same-side-los-p OBJ los-candidate rg-candidate)
                          (left-of-p OBJ lg-candidate los-candidate))

Candidate: lg-candidate
Constraints: Clique: c-candidate los-candidate
             Object: ob-candidate
             Funcs: (and (on-los OBJ los-candidate)
                         (same-side-los-p OBJ los-candidate ob-candidate)
                         (left-of-p OBJ c-candidate los-candidate))
```

Figure 4-3: Example hypothesis generation rules. More rules can be found in Figure D-2.

```
Candidate: c-candidate
Constraints: Clique: los-candidate
             Object: ob-candidate
             Funcs:  (and (on-los-p OBJ los-candidate)
                          (flanked-both-para-los-p OBJ los-candidate
                                                   ob-candidate))
```

This rule indicates that if the current clique, $\mathcal{C}$ contains a los-candidate and the blackboard, $\mathcal{B}$, contains an ob-candidate (offensive blob candidate), then the test in "Funcs" should be activated. The Funcs test is applied to each object, OBJ, labeled as an ob-candidate on $\mathcal{B}$. "(on-los-p OBJ los-candidate)" tests that the OBJ is next to the los-candidate, using the los-candidate from $\mathcal{C}$. "(flanked-both-para-los-p OBJ los-candidate ob-candidate)" tests that the OBJ is parallel to the los-candidate and flanked on both sides by other obj-candidate objects. If these two tests hold true, then the rule indicates that it is reasonable to hypothesize that the OBJ could be labeled as a C, and this hypothesis is added to $\mathcal{B}$. A sample of other hypothesis generation rules is shown in Figure 4-3 and Figure D-2.

The algorithm is initialized when the object (x, y) position data is added to the blackboard. Each object generates an initial blob-candidate (db-candidate or ob-candidate, for defensive and offensive blob respectively). A demon process, which is continually observing the blackboard and activating applicable context rules, will activate hypothesis generation rules with satisfied contextual preconditions. For example, most rules are not immediately activated because they require entries in $\mathcal{C}$ and $\mathcal{C} = \{\phi\}$. The first rule in Figure 4-3, however, is activated because a db-candidate and ob-candidate are found in $\mathcal{B}$. The function "vertical-midline-players" is run and generates some los-candidate objects that are added to $\mathcal{B}$. Note that the function can (and usually does) return more than one hypothesis location for the line-of-scrimmage, which is based upon the location of the input objects. Later in the processing, one of those los-candidate hypotheses will be added to $\mathcal{C}$ and trigger the activation of other hypothesis generation rules.

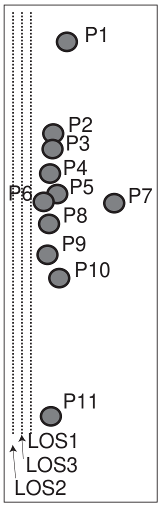
Hypothesis evaluation

The hypothesis generation rules are liberal. They propose too many hypothesized labels for any given object and assume that later processing will sort out the inconsistent information and select the best labels for the given input scene. The hypothesis evaluation stage uses a second set of context rules to rank hypotheses *within the same class*. For example, if the hypothesis generation rules have added several los-candidates (LOS1,LOS2,LOS3) to $\mathcal{B}$, where each candidate has a different position on the field, a hypothesis evaluation rule can compare and partially order the los-candidates. For instance, suppose a function, *rankLOS*, exists to compute how well each LOS candidate partitions ob-candidates from db-candidates and how well the partition is centered between the offense and defense players. *rankLOS* can rank order two LOS candidates. By *pairwise comparison* of each label candidate on $\mathcal{B}$ within the LOS class, a partial order within the class can be established. For example, the partial order could be written as $\{LOS2 \succ LOS1 \succ LOS3\}$, indicating LOS2 is clearly the best hypothesis label. Often a class will have multiple hypothesis ranking rules that apply in the given context. In this case, a given label must rank above some other label *for all applicable hypothesis evaluation rules* in order to top the partial order. For example, a partial order written as $\{LOS2 \succ LOS3, LOS1 \succ LOS3\}$ indicates that LOS2 and LOS1 rank higher than LOS3 for all applicable ranking tests, but that neither LOS2 or LOS1 consistently ranks above the other.

Figure 4-4 shows some of the class partial orders at a particular time during the search process. For instance, data object LOS1 has been ranked as a better choice for the LOS class using the ranking rules than data object LOS2. LOS3 and LOS1, however, are ranked as equally good choices. P5 is clearly the best choice for the QB class at this time.

Examples of hypothesis evaluation context rules are shown in Figure 4-5 and Figure D-3. Whenever the hypothesis generation rules add a new hypothesis to $\mathcal{B}$, the hypothesis evaluation demon will activate any applicable evaluation rules and recompute the relevant preference orders, which are stored on $\mathcal{B}$.

Intuitively, if a label-candidate outranks all other hypothesized labels using a comprehensive set of applicable ranking tests that take into account all known contextual information (and therefore dependencies between label-candidates), the label-candidate is likely to be



LOS:	LOS3
	LOS1 $\succ$ LOS2
QB:	P5 $\succ$ P4 $\succ$ P7
C:	P6 $\succ$ P5
	P6 $\succ$ P4 $\succ$ P8
LG:	P8 $\succ$ P6
LT:	ϕ (no hypotheses on $\mathcal{B}$ yet to rank)
FB:	P7
LHB:	P7
LFL:	P11 $\succ$ P10
	P11 $\succ$ P9
LSE:	P11 $\succ$ P10

Figure 4-4: Example showing some of the label partial orders at a particular time during the search process. For instance, data object LOS1 has been ranked as a better choice for the LOS class than data object LOS2 using ranking rules. LOS3 and LOS1, however, are ranked as equally good choices. P5 is clearly the best choice for the QB class at this time, given the object labels hypothesized on the blackboard, $\mathcal{B}$, and the ranking rules.

the correct label. When evaluators do not consistently rank one label candidate above another, no ordering is imposed. Therefore, preference ordering is based upon consensus, given all applicable hypothesis generation rules. This ranking mechanism is a variation on the theory of endorsements [CG83], when all endorsements are of equal weight. To avoid using a weighted sum of components, the algorithm instead relies on redundancy. The implications of this decision are discussed shortly.

Grouping mutually-consistent hypotheses

A third set of context rules is used to create the clique, $\mathcal{C}$, which is an internally consistent, partial description of a scene given the rule set. $\mathcal{C}$ is constructed by incrementally adding label candidates from $\mathcal{B}$ so that each new member of $\mathcal{C}$ is consistent with all previous members. A search process selects a label candidate from one of the class partial orders. A demon process then applies all applicable consistency rules, based on the label candidate, $\mathcal{B}$, and $\mathcal{C}$. Each rule runs a consistency check. When all applicable rules evaluate to true, indicating the selected candidate label does not conflict with some label in $\mathcal{C}$, the label is added to $\mathcal{C}$. Examples of consistency rules are shown in Figure 4-6 and Figure D-4.

When no more labels can be added to $\mathcal{B}$ and $\mathcal{C}$, the clique, which is one labeled interpretation of the scene, is saved. The order in which label candidates are selected for addition to $\mathcal{C}$ can significantly impact the labels that are selected. For instance, a poor choice for the LOS class may invalidate nearly all other label candidates for entry into the

```
Candidate: los-candidate  ; Prefer LOS that split the off/def blobs well
Constraints: Clique: nil
             Object: db-candidate ob-candidate
             Funcs: (los-off-def-split-cs)

Candidate: c-candidate    ; C: Prefer shorter distance to LOS
Constraints: Clique: los-candidate
             Object: nil
             Funcs: (eval (- (dist-los los-candidate cand)))

Candidate: c-candidate    ; C: Prefer closest to lg
Constraints: Clique: lg-candidate
             Object: nil
             Funcs: (next-to-cs OBJ lg-candidate)

Candidate: c-candidate    ; C: Prefer closest to rg
Constraints: Clique: rg-candidate
             Object: nil
             Funcs: (next-to-cs OBJ rg-candidate)

Candidate: c-candidate    ; C: Prefer C near to QB
Constraints: Clique: qb-candidate
             Object: nil
             Funcs: (quantify (near-p OBJ qb-candidate))
```

Figure 4-5: Example ranking rules. For more rules see Figure D-3.

clique due to checks that enforce reasonable distance-from-LOS constraints.

Constructing cliques

The clique construction is heavily dependent upon the order in which label candidates are selected for addition to $\mathcal{C}$. Given N label candidates, there are $N!$ possible label-candidate orderings, which would need to be tested to ensure that the best achievable clique is constructed. Search order heuristics are therefore required for (1) selecting a class and (2) selecting a candidate from that class (from their partial order classes) to attempt to add to the current clique. If a poor candidate is added to a clique early in the clique formation process, the erroneous candidate will most likely prevent the addition of valid candidates due to consistency rule conflicts. Therefore, the algorithm should add the most promising label candidates to the clique first. Strat and Fischler randomly select a class. Within the class, they try to add the label candidate at the top of the partial order for the class. If no candidate is at the top of the partial order (as in the LOS class in Figure 4-4), the algorithm selects a top candidate randomly [SF91].

In the football domain, if each class has a label candidate that ranks at the top of the class, the search complexity is roughly $M!$, where M is the number of possible classes,

```
Candidate: wb-candidate ; Backs must be off LOS
Constraints: Clique: los-candidate
             Object: nil
             Funcs: (just-off-los-p OBJ los-candidate)

Candidate: guard-candidate ; C and Guard need to be a bit apart in ydir
Constraints: Clique: c-candidate
             Object: nil
             Funcs: (bit-of-space-y-p c-candidate OBJ)

Candidate: c-candidate ; C needs to be between the QB and LOS
Constraints: Clique: qb-candidate los-candidate
             Object: nil
             Funcs: (between-p OBJ qb-candidate los-candidate)

Candidate: c-candidate ;C needs to be flanked on left (wrt LOS) by LG
Constraints: Clique: lg-candidate los-candidate
             Object: nil
             Funcs: (flanked-left-p OBJ los-candidate lg-candidate)
             Funcs: (flanked-right-p OBJ los-candidate rg-candidate)
```

Figure 4-6: Example consistency rules. More rules are shown in Figure D-4.

or approximately 20. Therefore, an additional heuristic is used to bias the random class selection towards a breadth-like search that is significantly different each time a new clique is constructed. Each time the algorithm finishes constructing a clique, the algorithm saves the clique and resets. On the following clique construction iterations, classes are randomly selected, except that selection is biased to select classes not selected early in previous clique constructions. Similarly, within a class that has multiple candidate labels at the top of the partial order (e.g. LOS in Figure 4-4), the random label selection is biased towards labels not tested early in previous clique constructions.

Strat and Fischler assume that the partial-order ranking system never mis-ranks labels within a category. In the formation labeling system, a final heuristic is used so that when a candidate label is tested unsuccessfully for addition to a clique, it is removed from the partial order for the remainder of that clique construction. A new class is then selected. This prevents a single, partial-order ranking error from blocking a desirable candidate label from being added to a clique. Therefore, if in Figure 4-4 the algorithm selects the QB class and attempts to unsuccessfully add P5 to the current clique, then P5 is removed from the clique and in some future construction P4 will be considered.

Selecting the best consistent hypothesis

Label candidates are added to the current clique until either (1) no more label candidates are being generated and all generated label candidates are inconsistent with the clique or

(2) the clique is judged to be "good enough" based on a scoring function.

The evaluation metric for a clique selected by Strat and Fischler and used here is the percentage of data that has been labeled [SF91], which is a commonly used heuristic (e.g. [JJM85]). For the formation-labeling task, a heuristic scoring function evaluates cliques based upon the number of the 11 offensive players that have been labeled and an additional requirement that certain player positions such as the C and QB have been assigned. At any time, the clique that has the highest clique score is considered the best match. Several cliques can have the same clique score, in which case they are equivalently good interpretations of the scene.

Once each clique is constructed, the clique and its score are saved and a new clique is constructed. Intermediate feature computation results are cached to improve performance. After each clique is completed, the cliques are ranked by their clique score. The stochastic search process can be halted when a maximal score indicating a good description of all the provided data has been achieved. Alternatively, the algorithm will provide the best answer after some specified number of clique constructions.

Knowledge engineering

Encoding the three rule sets is a laborious process. Strat and Fischler argue that three properties of this system mitigate the difficulty of rule construction [SF91]. The first is that rules are broken down by label class, limiting the amount of knowledge that needs to be considered when designing rules within a class. For example, when designing ranking rules, the rules only need to compare between similar player labels (i.e. is P4 a better C than P5?). Second, the rules reflect typical, sufficient conditions making the rule sets relatively succinct. Finally, rules need not be perfect because the rule set is highly redundant.[3] In the formation labeling system, new hypothesis generation rules *are* relatively easy to add, and simple functions can typically be used to check for consistency and for hypothesis generation. Ranking rules, however, have been more difficult to construct for reasons discussed shortly.

Formation labeling algorithm overview

To summarize the formation labeling system, the algorithm consists of the following steps once the rule sets from the formation-labeling problem have been specified:

1. 11 object-candidate labels, one for each player, are added to the blackboard, $\mathcal{B}$.
2. Activate applicable generation rules based on preconditions examining $\mathcal{B}$. Rules add label candidates to $\mathcal{B}$ for objects.
3. Activate applicable ranking rules based on preconditions examining $\mathcal{B}$ and produce partial orders within each label class. Partial-order ranking information added to $\mathcal{B}$.

[3] As discussed shortly, rule accuracy is critical for good system performance when unanimous endorsement by ranking rules is required.

4. Loop until maximum clique score achieved.
   (a) Create a new empty clique, C_i.
   (b) Select a label class randomly among classes, biasing selection towards classes that have not been previously marked on $\mathcal{B}$. Mark chosen class as selected on $\mathcal{B}$.
      i. If label class contains no candidate labels on partial orders, goto step 4.b.
      ii. From label class, select top-ranking candidate label (or randomly select among multiple equally ranked options, biasing the selection towards labels that have not been selected early in previous iterations). Pop candidate label from partial order.
      iii. If candidate label is accepted by consistency rules, given the state of $\mathcal{B}$ and C_i, then add candidate label to C_i. Otherwise, goto step 4.b.ii.
      iv. Activate applicable generation rules based on preconditions examining $\mathcal{B}$ and C_i. Rules add label candidates to $\mathcal{B}$ for objects.
      v. Activate applicable ranking rules based on preconditions examining $\mathcal{B}$ and C_i and update partial orders within each label class. Partial-order ranking information added to $\mathcal{B}$.
      vi. Goto step 4.b.
   (c) Compute the clique score for C_i and add C_i and the score to $\mathcal{B}$.
   (d) Output the clique on $\mathcal{B}$ with the maximum score as best current result.

4.2.2 System performance

The construction of the system helped identify some problems with the approach described here that led to the development of an entirely new algorithm described in Chapter 6 for the dynamic-scene labeling problem of football play recognition. The system, therefore, which suffers from problems to be discussed in Section 4.3, was only tested on a set of ten formation examples – primarily to evaluate architectural problems.

In practice, on these ten examples, the stochastic search process is unlikely to find the best clique without at least 1000 clique constructions, often 10,000 constructions.[4] During development of the ten examples, few situations were encountered when system rules could not be added or adjusted to accommodate a new type of formation.

One specific troublesome example, shown in Figure 4-7a, was presented to the system. The desired labeling is shown in Figure 4-7b. Figure 4-7c shows the best result produced by the system. The first difference to note is that the system does not make distinctions between the LSE (left split end) and SE (split end). The system was designed to minimize the number of classes because every added label class significantly increases the complexity

[4]Each clique construction takes approximately 1 second on an AlphaStation 500MHz using (rather inefficient) LISP code.

of the search space. Therefore, an assumption was made that a post-process could add directional descriptors to labels like the FL and SE. The only other difference between the two scene interpretations is the labeling of the RTE in Figure 4-7b as the WB in Figure 4-7c. This is an error due to the label of the FL and RT; the object labeled as a WB must be the TE, which is a player who is next to the LOS. The WB is a player who should be 1 yard off the LOS. This error results from several missing rules in the rule set. One missing rule is the preference to label an object that is close to the LOS a TE versus a WB, which should be 1 yard off the LOS. Due to noisy data, the rule sets permit the object mislabeled as WB to be detected as either on or 1 yard off the LOS. The rule sets, however, should encode the information that an object in that position is *more likely* to be the TE than the WB.

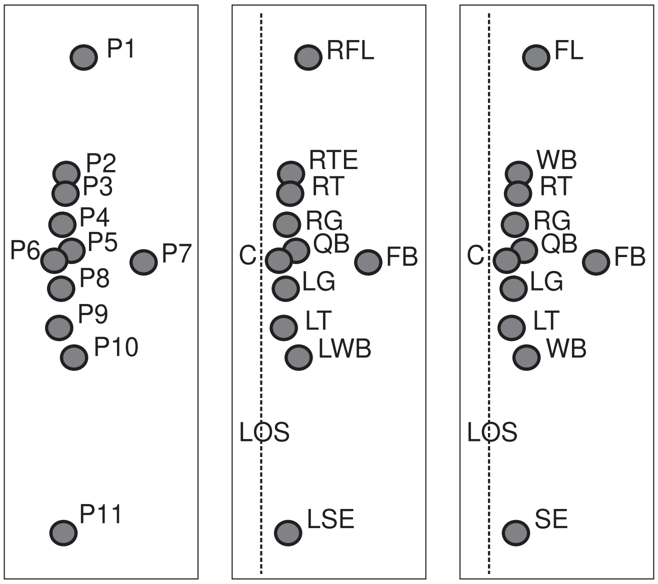
The second missing rule pertains to the rule in football limiting the number of players that stand in formation directly next to the LOS to exactly 7. All the other players must stand at least 1 yard back. A rule is in the database that specifies that *only* 7 players can be next to the LOS. Therefore, since the SE, LT, LG, C, RG, RT, and TE positions are all objects that start on the LOS, a scene labeling where both WBs in Figure 4-7c are instead labeled as TEs will never be proposed. No rule is present, however, that enforces the inverse: *no less* than 7 players can be on the LOS. This rule would be added to the database indirectly by using a constraint rule that indicates no more than 4 players can be more than 1 yard off the LOS. Unfortunately, to improve search performance, this constraint on the number of objects on the LOS needs to be added to the ranking rules, but the ranking rules cannot rank a TE above a WB because labels are only ranked *within* their own categories.

4.3 Evaluating the representation

The formation-labeling system was developed to explore one method for recognizing scenes in a contextually rich, multi-agent problem domain. The goal was originally to expand the method to the temporal domain and apply context-based generation, ranking, and consistency rules to the multi-agent domain of play recognition.

The system did successfully label static scenes. However, experience developing the system and evaluation of the system's representational limitations prompted a complete switch from a rule-based search to a probabilistic, graphical model classification approach described in later chapters. The rule-based system was conceptually difficult to extend to a temporal recognition system. The representation inadequacies are discussed below. Later in this document, these issues will be discussed again with respect to the play recognition system developed in Chapter 6.

Partial order sensitivity. The effectiveness of the search process is critically dependent upon the partial orders generated by the ranking rules. Inaccurate partial orders will often result in incorrect labels being added to cliques early in the search process. These erroneous labels prevent correct labels from later being added to cliques. The partial orders not only need to be correct, however, they need to be maximally ordered. Partial orders are most informative when a single label candidate is ranked above other label



(a) (b) (c)

Figure 4-7: (a) objects to annotate with position labels. The (b) ideal and (c) computed labeling for the formation in (a).

candidates. Partial orders are least informative when no label candidates are ranked above the others. Overall, three cases are possible: (1) informative partial orders that are correct, resulting in a single or small number of label candidates at the top of a class, (2) non-informative partial orders that are correct, providing no useful searching guidance within a label class, and (3) incorrect partial orders, which will seriously impede the searching process and can sometimes make recovery of a correct solution impossible. In the formation labeling system (and in Strat and Fischler's CONDOR system[SF91]), a candidate label, L_i is ranked above other candidate labels $L_1 ... L_n$ only if *all* the ranking rules prefer L_i over each of $L_1 ... L_n$. Otherwise, no ranking information is used, even if only one ranking rule is in disagreement. This strategy was motivated by the observation that any candidate label that tops all the ranking tests is likely to be a good label to try and add to the clique. This strategy, however, results in the following counterintuitive behavior: the more ranking rules that are provided to the system, the more difficult it is for the system to compute an informative partial order. More information results in a less-directed, not a more focussed search. If p is the probability that a ranking rule will produce an erroneous result, the probability that n ranking rules will produce no error, $(1-p)^n$, is continually decreasing as more ranking rules are available. As a result, the knowledge engineering process is made more complicated because the system lacks a clear separation between the search control and the rule set. Adding ranking rules can adversely affect search efficiency and, in some cases, prevent the system from ever recovering a correct solution.

Typical solutions can be hard to detect. As a consequence of partial orders losing their discriminatory ability, possible but unusual solutions can require an extensive search. A more problematic issue, however, is that highly typical solutions can also require extensive search because the ranking rules are only ranking within label class. A ranking mechanism that provides a heuristic to help select which class to consider next, in addition to which label within the class, is needed to ensure that highly typical solutions are found quickly.

Lack of mechanism to "connive." The formation-labeling algorithm reuses little information between each clique-generation phase. With the exception of some information used to bias the search away from quickly selecting label candidates chosen early for previous cliques, each clique generation process proceeds independently.[5] The system has no mechanism for "conniving" [SM72] by checking for problematic rules and candidate labels. A conniving system might statistically monitor each rule and label candidate throughout the recognition process. Rules and labels that are causing an unusually high consistency rejection rate might subsequently be tagged so that they are given low selection priority during the intra-class and inter-class search selection. During the generation of a single clique, if one label in the clique is causing a large percentage of consistency rules to reject other labels, then the troublesome label might be removed. The algorithm would then backtrack to the time when the troublesome label was added to the clique and resume searching.

Lack of mechanism to "tweak." The formation labeling system suffers from a lack of ability to "tweak" clique solutions. Cliques are commonly formed that are close to the ideal formation-labeling but have the labels for two adjacent players swapped (e.g. a RSE is labeled the RFL and visa versa). Some clear patterns are apparent that could be addressed using an additional set of context rules that take a finished clique and perform a set of common tweaks on the set, such as swapping two frequently confused player labels (a form of case-based search, see [Ham89]). The system would then recheck the consistency rules and continue processing either the original or the newly tweaked clique, depending upon which has the higher clique score. The tweaking rules could make use of the probabilities of observing particular formation configurations.

No probabilistic ranking and evaluation. From the outset, the system described in this chapter was developed with the assumption that one major flaw of the framework must be overcome in the subsequent system: the lack of a mechanism for reasoning about uncertainty. The resulting brittleness of the system is apparent in both the partial-order ranking process and the evaluation process. The ranking process suffers

[5] Cached computation from feature detectors is also saved between clique searches for computational efficiency.

substantially because a candidate label L_i is only ranked above L_j if all applicable ranking rules suggest that $L_i \succ L_j$. Uncertainty is replaced by percentage of consistency. However, the ranking rules themselves depend upon continuous and uncertain measurements of spatial concepts like proximity and relative orientation, and sometimes when combining two spatial measurements into one detector, information is lost as each spatial measurement is conservatively quantized. A better mechanism would propagate uncertain information throughout the reasoning process. Finally, this algorithm can only detect a consistent scene labeling. It is often possible to have multiple consistent formation labeling, but it should be the case that one of the labelings is preferred over the others (i.e. has a higher probability).

No mechanism to evaluate partial interpretations. The algorithm proposed here does not provide a mechanism for evaluating partial interpretations of the scene. The first problem is that the entire scene must be observed. In the case of play recognition, detectors are desired that can partially (i.e. causally) observe the play in time and make a hypothesis early in the play action. This is analogous to the labeling system having player data revealed incrementally and the system continuously computing a best guess formation. Second, as cliques are constructed, it would be desirable to have a mechanism by which the reliability of operators used to generate the interpretation combined with the percentage of the data explained could be used to compare the clique being constructed with cliques already proposed. For example, a probability or goodness function for the clique might be used to terminate the construction of a poor clique early if other constructed cliques are clearly superior. Alternatively, the algorithm might postpone the construction of the poorly scoring clique in favor of completing construction of more promising cliques first.

Trading off space for time. The formation-labeling algorithm's performance could be improved by trading off memory for computation time using the rule-monitoring RETE algorithm [For81, For82].

Tuning the search complexity vs. brittleness tradeoff. The formation-labeling algorithm essentially trades off the size of the searched space for confidence in the evidence and ranking rules. For example, in the worst case assume that the evidence is so noisy that no ranking rules obtain a consensus within some label class, $L : L_1, L_2, ..., L_n$. As a result, instead of a partial order that rapidly prunes the search space (e.g. $\{L_2 \succ L_1 \succ ... \succ L_n\}$ designates a clear search order), every entry in L is equally likely to be a clique member. A large search space results. As more ranking rules are used, however, the performance of the algorithm will become more brittle. A single faulty ranking rule can dramatically increase required search time and, sometimes, prevent the desired solution from ever being recovered in a reasonable time. The current algorithm scales poorly.

Using independence. Independence can be used to reduce the maximum size of the search space, and recognition problems in real domains will require that the problem be grouped into mostly independent sub-problems of manageable chunks. For example, in the football domain, the formation-labeling task described in this chapter can be accomplished independently of the play-labeling task described in later chapters. The system presented in this chapter does not make independencies explicit to the designer, and it is not clear how they would be exploited.

Multiple rule sets. Having the three distinct sets of rules for three tasks: generation, ranking, and consistency checking has proven problematic. Rules in each set are similar but not exactly the same. For example, a generation rule specifies that an object to the right of the LG could possibly be the C, and therefore, a c-candidate label is assigned to the object. A ranking rule specifies that a c-candidate label for the object that is to the right of the LG in just the right way should rank higher than a c-candidate label assigned to some other object that is to the right of the LG but in some problematic way (e.g. it's too far to the right). Finally, a consistency rule checks that the c-candidate label for the object is to the right of the LG. Encoding three rules for each concept in the domain is a tedious process that, in the case of the formation system, proved prone to error. If the rule sets were extended to include a representation of uncertain information, maintaining probabilistic consistency between the three rule sets would be problematic for the rule designer. The rule base is more verbose, more difficult to construct, and more difficult to debug than it probably needs to be.

Comparing across label categories. As described in this chapter, the framework has no mechanism for comparing label candidates across classes using ranking rules (e.g. ranking the TE and WB might fix the problem described in the example in Section 4.2.2).

Incorporating *a priori* information. The algorithm presented here has no mechanism for using uncertain *a priori* information to bias the search towards a particular set of solutions. If a particular label is known, it can simply be added to the clique at the start of each search process. If some label is known only to be *more likely*, however, no mechanism exists for entering suggestive evidence so that it is used by the ranking rules.

Handling fuzzy evidence. The algorithm described in this chapter does not have a mechanism for dealing with uncertainty in data. Further, the algorithm fails to provide a robust mechanism for handling the fuzzy nature of spatial definitions like "behind" and "near." Feature detectors are designed with worst-case thresholds and consequently often ignore useful but subtle measurement differences. For instance, as in the example in Section 4.2.2, the system often has trouble selecting between a label for a player who is supposed to be next to the LOS and a label for a player who is supposed to be 1 yard off. When the object is somewhere between 0-1 yards, the distance detectors typically are not using the distance information that is available to discriminate between the cases. Further, even when ranking rules do use this information, they are doing so without taking into account the noise in the data. A better alternative would be to explicitly model the uncertainty of the detectors and data.

Multiple labels per data item. The algorithm presented here assumes that the best labeling of a scene is one where each data item in the scene has one label assigned to it and where all labels are mutually consistent. In many domains, however, a given label may span more than one data item. Further, the "best" labeling may consist of a scene where some data items have multiple labels and others only have labels shared between data items. In particular, when actions with temporal extent are considered, multiple actions can involve the same data items but overlap only partially in time. The method does not prevent multiple label assignment to single data items. Doing so will increase the complexity of the rule sets and require a completely new mechanism for computing a clique score. The clique score most likely would need to be based on the strength of the evidence for a solution in addition to the data coverage of the solution. This suggests a probabilistic approach that integrates uncertain information may be preferable to the current winner-take-all ranking approach.

Knowledge construction difficult. Constructing the knowledge base for the formation-labeling system proved difficult and time-consuming. Rules, despite only being active in limited contexts, are frequently dependent upon other rules. For example, an entire set of rules for detecting the LG may depend upon the c-candidate label. However, if one c-candidate rule eliminates the correct C hypothesis due to spurious noise, then detection of the LG is adversely affected. These *non-explicit* rule dependencies are time-consuming to "debug," requiring hours of run-tweak-analyze cycles.

Learning from examples. One way to make the knowledge base easier to construct is to learn the context rules from examples. The difficulty of learning using several popular learning frameworks in multi-agent domains will be discussed later in this document.

In summary, the representation used for formation labeling suffers from a lack of a probabilistic framework to guide the search for a solution without relying upon hard, brittle thresholds. Some of the assumptions made by Strat and Fischler [SF91] that led to the use of a large set of simple rules were reasonable in practice *except* with respect to the critical hypothesis ranking step in the algorithm; here the algorithm proved too brittle. Rules that softly weight relative evidence – modeling some important dependencies between related rules and evidence – seem necessary to make the algorithm practical given noisy trajectory data. Additionally, a mechanism for representing temporal relationships is required.

4.4 Lessons learned: desirable recognition properties

Overall, perhaps the most important lesson learned from the exercise of implementing the context-rule, formation-labeling system is that a large number of relatively simple, context-based comparisons between properties of multiple objects can be used to evaluate an enormously large search space. The idea that a large set of simple, binary comparisons can be used to recognize structured objects is a core principle of the multi-agent action recognition algorithm described in Chapter 6.

The problems discussed in the previous section can be used to develop a checklist of desirable qualities for a representation for recognition in a multi-object domain. These criteria are listed below and used later in this document to evaluate the framework described in Chapter 6.

- Exploit the power of many low-order comparisons between objects.
- Represent and reason about uncertainty in models and evidence detectors.
- Explicitly encode the "fuzziness" of spatial and temporal descriptions.
- Permit *a priori* information to be used to bias the recognition process.
- More information should improve, not degrade, system performance.
- As more information is added, less effort should be required to do so.
- Typical situations should be easy to detect (i.e. require the shortest computation time to label). Unusual situations should be possible to detect given additional computational time.
- A clean separation between the knowledge representation and the control mechanisms should exist.
- Multiple, independent rule sets encoding similar information should be avoided if possible. Unify the rule base.
- Allow early pruning of unlikely options.
- Encode domain rules in an intuitive format that a non-expert can use.
- Exploit independence in the problem domain to reduce combinatorics.
- Automatically generate some or all rules from examples.
- Feature evaluation should be cached for performance improvement.

Some intellectual effort was invested in trying to improve the recognition method presented in this chapter so that it would be (1) more robust to uncertainty in the data and descriptions of actions, and (2) extendable to the temporal domain of play action recognition. On both counts, extending the representation directly proved problematic. However, the play recognition algorithm described in Chapter 6 does maintain the idea of using a large number of simple consistency checks between objects in the scene – only it does so without ignoring the problem of propagating uncertain information from sensors and *a priori* models and does so for the task of labeling a dynamic scene.

Chapter 5

Approaches for multi-agent action recognition

The analysis in Chapter 2 suggests that intentionality modeling may be necessary to explain *why* some action happened but that a computational system may be able to identify *what* an action is by "compiling-down" collaborative intentional behavior into patterns of typical coordinated activity. The labeling test system described in Chapter 4 further suggests that one way to encode the compilation of multi-agent activity is to develop action models that use low-order relationships between the multiple objects in the scene. These models can then be matched with perceptual data.

This chapter examines some prior approaches to object recognition, plan recognition, and representing uncertainty. The work and representational issues discussed here and in the previous chapter motivate the development of the representation proposed in Chapter 6 and will be used for comparison in that discussion.

5.1 Object recognition

In certain respects, the problem of object recognition, which has been well-studied (see [Gri90, Pop94] for surveys), is similar to the problem of action recognition. The goal of this section is to identify commonalities between the well-studied, static object recognition task and the relatively new temporal and multi-agent action recognition task using an object recognition framework developed by Grimson and Lozano-Pérez [Gri90, GLP87].

Although some work has explored the recognition of objects by reasoning about object *function* [SB94] and some work has investigated the use of color histogram features for object recognition (e.g. [SC96, EM93]), the majority of object recognition research has focused on shape-based, geometric representations for matching man-made geometric objects to images. Systems that recognize natural objects in natural settings are less common (e.g. [SF91, HS93]).

Most generically, the task in geometric object recognition is as follows. Given some model, M, compute a set of attributes, $f(M)$. Given an image, I containing some objects, o, compute some set of features, $g(I)$. Find a good match between $f(M)$ and $g(I)$ if such a match exists using a set of allowable transformations. When designing a recognition algorithm, critical questions are (1) what are the models and model attributes, (2) what are the image features, and (3) how is the correspondence between features and some model using model attributes found?

5.1.1 Geometric correspondence and search trees

Grimson has developed an instructive, general framework for analyzing the object recognition problem [Gri90]. The goal is to estimate a "set of feasible interpretations" of some data, which is most typically a two-dimensional image. The interpretations consist of sets of known models situated in a coordinate system, where the models have been transformed in some way to match with the given data. Given an image and a set of models, object recognition consists of three stages: (1) determining which object or objects are present in the image, (2) determining which subset of the image data should be matched with the

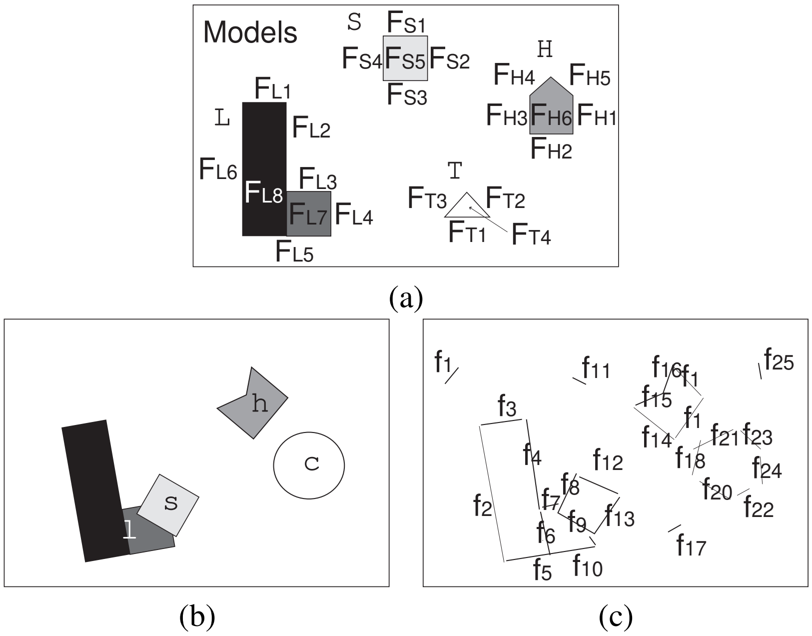


Figure 5-1: (a) Object models, (b) an image of a scene containing some objects, and (c) a feature image.

object model(s), and (3) determining the transformations on the models that map the models to the data or subsets of data. These tasks are commonly referred to as *indexing*, *selection*, and *correspondence*, respectively.

A simple example, an extension from [Gri90], is shown in Figure 5-1. Figure 5-1a shows simple, two-dimensional models of four shapes: an L-shaped model ($\mathcal{L}$), a house-shaped model ($\mathcal{H}$), a triangle-shaped model ($\mathcal{T}$), and a square-shaped model ($\mathcal{S}$). $\mathcal{L}$ consists of two components, one of which is $\mathcal{S}$. Here it is assumed that these models are sufficiently well-defined in a geometric and color space so that a user can easily provide model specifications.[1] Edge attributes (e.g. F_{L1} and F_{H5}) and intensity region attributes (e.g. F_{L8} and F_{T4}) are indicated. Attributes can be computed on these models using functions that detect edge lengths, relative orientation between edges, edgel intensity, parallel edges, etc.

Figure 5-1b shows a particular scene of an image containing objects. Object l is occluded by object s. Object h is distorted, and the circle object c does not appear in the database. All the objects are rotated relative to the database models. Figure 5-1c shows a possible edge image obtained by applying an edge-detection algorithm to the image in Figure 5-1b, with each edge labeled, f_i. The feature image contains spurious features that correspond to none of the known models – some due to noise and some due to an unknown object.

[1]Not all models of interest can be as precisely conveyed. For example, non-rigid objects like trees and clouds are difficult to define in a geometrically precise manner.

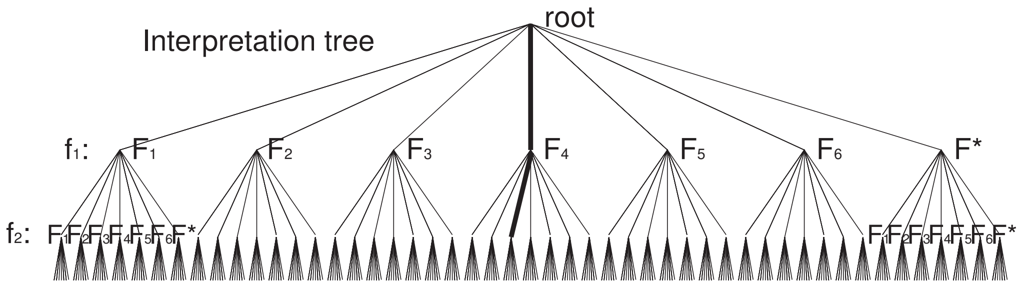


Figure 5-2: An illustration (adapted from Grimson[Gri90]), showing the image interpretation tree used to search matches from model image segments to image feature segments.

The task is to search for correspondence between the known model attributes, F_i (edgels and region intensities), and the features extracted from the scene image, f_i, using some set of allowable transformations. Consider just a single model, $\mathcal{H}$, with attributes, $F_{Hj}(1 <= j <= 6)$ and a set of features, $f_i(1 <= i <= 25)$. The search space can then be described by an *interpretation tree* [Gri90], illustrated in Figure 5-2.

The correspondence search begins at the root of the interpretation tree. Each layer of the tree, i represents all possible matches (given matches higher up in the tree) between one feature, f_i and all possible model attributes, F_{Hi}. A *hypothesis* is a path from the root to a leaf that specifies the match of each model attribute to specific image features. For example, the highlighted path in Figure 5-2 denotes a hypothesis where image feature f_1 has been matched to model attribute F_{H4} and image feature f_2 has been matched to F_{H3}. Following Grimson's notation, F_* denotes no match, or a "wild card" match. One good match for model $\mathcal{L}$ to object l might be f_1:F_*,f_2:F_{L6},f_3:F_{L1},f_4:F_{L2},f_5:F_{L5},f_6:F_*,f_7:F_{L5},f_8:F_*,f_9:F_*,f_{10}:F_{L5},f_{11}:F_*, and then for $(12 <= i <= 25) f_i$:F_*. Note that F_{L5} matches to two image features, f_5 and f_{10}, since the edge was "broken" by the detector. Also note that some edges are missing due to occlusion caused by s.

In this example, there are two attributes that could be matched – edgels or region intensities (region intensity features are not indicated in Figure 5-1c). A scoring function is required to test if $r(f_i) \approx F_{Lj}$, where r is some allowable matching transformation (e.g. translation, scaling, or rotation). One unary scoring function, considering only one attribute and feature at a time, might be (length$(f_i)-$length(F_{Lj})) where some threshold is defined specifying the maximum difference in distance required to declare a good match. Alternatively, no threshold could be used, and the function could return a "goodness value" related to the distance. The unary length scoring function is not useful if one allowable transformation is scaling. Binary detectors that simultaneously compare two features can also be defined. For example, one scale-invariant binary detector checks if the angle between two model edgel attributes is the same as the angle formed by two feature edgels, once again using some threshold to indicate a good match (or a continuous-valued goodness instead). A robust goodness value might incorporate the reliability of the feature detectors as well as the distance between the match. In addition to unary and binary comparisons, trinary and n-ary $(n > 3)$ relations can be computed and used to compute a match value at a particular node in the interpretation tree. Examples of trinary features are the relative angles of three model attribute edgels or the maximal distance between three model attribute edgels.

Each hypothesis consists of a set of attribute-feature comparisons. Therefore, a method for integrating the uncertainty associated with each comparison is required so that hypotheses in the interpretation tree can be explored using an informed search (e.g. heuristic best-first). A string of bad, but passable matches should eventually lower the goodness value of a particular hypothesis; a hypothesis with many extremely good matches and a few poor matches might be a better hypothesis overall.

One problem is that even though only one model is being matched to the image, the search tree has $(i+1)^j$ states, where i is the number of model attributes and j is the number of detected features.

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 $F_1 ... F_*$. If these tests meet the matching criteria, then binary constraints in the current hypothesis path are checked. For example, if the current hypothesis is $\{f_1{:}F_*, f_2{:}F_{L6}\}$ at level 2, then each child would be checked for binary consistency with $\{f_2{:}F_{L6}\}$ and also with $\{f_1{:}F_*, f_2{:}F_{L6}\}$. 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 [GLP87], 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 5-3, taken from [Gri90], 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 5-3. All binary pairs being consistent does not imply that the match is correct; the model and features must be transformed to the same coordinate system and all attributes must be checked for matches. This appears problematic, because it is computationally intensive to check all n-ary relationships.

Grimson and Lozano-Pérez made a useful observation, however [GLP87]. 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 conjecture allows them to construct heuristic pruning methods that search for the correct interpretation by only maintaining unary and binary consistency.

It is this idea, *that massive low order consistency typically implies correctness,* that

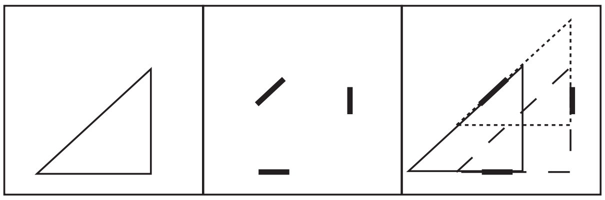


Figure 5-3: Low-order consistency does not imply global correctness.

drives the approach to recognizing complex actions presented in this work. 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, as described in Chapter 6.2. The issue is how to apply this idea to a noisy, multi-agent action recognition task.

5.1.2 Insights from object recognition

The goal of section is to compare the object recognition task to the action recognition task. So far, the analysis in the previous section raises three issues: (1) what are the fundamental feature model attributes for multi-agent recognition, (2) what are allowable transformations on those attributes, and (3) what are the match scoring functions and how do they integrate uncertainty over a hypothesis for a team activity? These questions are answered for the football play recognition system in the next chapter.

Another question is how the issues that make object recognition difficult map into the multi-agent action recognition domain. Those issues (as outlined by Grimson [Gri90] and others) are described below. How the system proposed in this thesis handles these issues will be discussed in Chapter 7.

Occlusion. Objects frequently occlude one another in image scenes, requiring that some models match only partially to image data. $\mathcal{S}$ occludes $\mathcal{L}$ in Figure 5-1b. Consequently, the recognition algorithm must either allow objects to be matched incompletely or use information about other objects in the scene to explicitly reason about the occluded model components. Both methods significantly increase the complexity of the search space. In the football action domain, the defense creates situations analogous to static occlusion; sometimes part of an action may be missing or performed in an atypical manner because of interference from some defensive player. Occlusion results from unmodeled interactions between objects or actions of objects.

Noise and spurious data. Noise in the data and imprecision in feature detectors results in noisy individual features (e.g. misaligned edges, shortened edgels, etc.). Further, as shown in Figure 5-1c, edge attributes in the model be split in two or missing altogether in the imagery. Additional data not associated with any known model is also common. Noise in the football action domain is caused by inaccuracies in object tracking but also from bad judgment on the part of people in the scene (i.e. a player makes a mistake) and from the unpredictable behavior of the defense.

Imprecise models. Model $\mathcal{H}$ poorly models the "house" object that appears in Figure 5-1b. Although small, un-modeled distortions can be treated as noise and handled by the matching metrics, larger, un-modeled distortions are problematic. In the football action domain, it is less clear how to measure the precision of a particular multi-agent model.

Hierarchical labels. There is an ambiguous situation in Figure 5-1b because the scene can be interpreted as containing $\{\mathcal{L}, \mathcal{S}\}$ or some other rectangular object (assume a model for that is also available) and two $\mathcal{S}$ objects since $\mathcal{S}$ is a subpart of $\mathcal{L}$. Therefore, there are multiple "correct" object assignments for the given scene. The question arises, which description is the "best" description. Further, should object l be modeled as a single entity or as two attached components? The same problems arise in the temporal domain, where basic units for recognition must be combined into temporal patterns of action.

Hypothesis vs. testing differences. Object recognition systems that search the correspondence space like the interpretation tree algorithm distinguish between making a match hypothesis and testing that match for correctness. In Grimson's algorithm, unary and binary detectors are used to generate hypothesis, and a small number of good hypothesis are tested for full correspondence with the given model using a different pose-transformation and matching metric. This situation is directly analogous to the multi-agent action recognition problem of recognizing collaborative activity by recognizing low-order consistency between features that indicate coordination. A good match generates a hypothesis for some collaborative activity, such as a team play. However, definitively confirming that the hypothesis is "globally consistent" may require intentional reasoning about agent interaction; intentional inference may be required to confirm that the observed coordination definitely resulted from team collaboration and not random agent interactions.

Indexing. In the example, only one model, $\mathcal{T}$ does not appear in the image. However, for many tasks there may be hundreds of objects in the model database; a football team can have a repertoire of several hundred play variations. Which plays are matched to the scene? Grimson notes that the expected amount of search to determine that an object is not present is exponential in the number of features – even with early termination based on match scores [Gri90]. Clearly, good hypotheses should be identified as early as possible via a best-first search algorithm.

Feature clustering. In real domains, pre-processing of features to find clusters of highly-characteristic features for an object is typically required to make the interpretation tree search practical [Gri90]. One such method uses focus features [BC82] and can be used to determine a small subset of data to match. Such focusing methods are potentially exponential in the number of features, however. An alternate to feature clustering and then correspondence matching is global feature matching (e.g. Hough clustering, see [Gri90]), but global matching algorithms do not cope with occlusion. In the multi-agent domain, a representation for each agent is desired because a single agent can create a single visual outlier but not invalidate the entire play. Additionally, representing agents individually allows for the recovery of individual agent actions (e.g. blocking, catching) that are useful for certain tasks.

Match score. In practice, real recognition problems require wild card matching to allow for model attributes being occluded in the feature image, but wild card matching increases the search space substantially. In these large search spaces, to limit complexity, search cutoff based on matching scores is required for reasonable performance [GLP87]. The search can be terminated when a "good enough" solution is found with either a high match score or a high percentage of data features accounted for.

Invariants. In object recognition, features are most valuable when they are invariant to changes in the environment such as viewpoint and scale. A weak feature can sometimes be made more powerful by checking for higher-order relationships between features. Alternatively, features can be used to robustly recognize higher-level components of an object and then recognize the relationships between the object components. In the temporal domain, the important features are the spatial and temporal relationships between objects. In fact, in the football domain, the temporal relationships between agent goals are critical, as discussed in the next chapter.

Context-based models. In consistency generation using interpretation trees, the assumption is made that object models are sufficiently strong to make similarity matching possible. However, Strat and Fischler, studying the recognition of natural objects, noted that "the verification problem changes from identifying objects based on sufficient conditions (e.g. of similarity) to that of eliminating alternatives based on a failure to satisfy necessary conditions" [SF91].

A primary difference between the object recognition task described in this section and the multi-agent recognition task is that, visually, multi-agent action will tend to be less constrained due to the interaction between individual agents. However, that does not mean that the multi-agent action to be recognized does not have structure. In this work, the structure will be temporal relationships between agent goals. Agent goals can be detected visually with some likelihood value using a probabilistic framework, and then temporal comparisons between goals can be used to check for consistency. The constraint results from the agent *coordination*.

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. In effect, the conjecture is that if many low-order temporal comparisons between goals are observed as expected given some play model, it is unlikely the given data is not that particular play. Absolute confirmation of a match (i.e. checking for full correspondence) would require a system that reasons about intentionality and propagates all temporal information to check for global temporal consistency. Such a check will not be attempted here.

The ability to detect agent goals and integrate uncertain information from detectors will require a framework for representing and integrating uncertain information.

5.2 Representing uncertainty

Any model used to recognize an object or action from noisy, perceptual data will benefit from (if not require) the ability to represent uncertain information. For example, empirical analysis suggests that when weak or conflicting evidence is observed, which is typically the case for perceptual recognition tasks, probabilistic expert systems exhibit better diagnostic capability than non-probabilistic rule-based systems [PHP+96]. As discussed in Chapter 2, multi-agent action has three primary sources of uncertainty: (1) the temporal constraints of the action description itself are typically probabilistic, (2) the fuzziness of attributes that must be assigned to individual elements, and (3) noisy data and sensors. A representation is called for that can robustly process uncertain input.

The search for geometric correspondence using interpretation trees discussed in Section 5.1.1 suggests that it may be possible to recognize some structured objects using unary and binary feature comparison detectors. However, not discussed as of yet is how these detectors encode uncertainty. Grimson notes that in their simplest form, these detectors return a binary decision indicating whether the features match within some specified tolerance [Gri90]. However, a preferable approach for applying the interpretation tree matching algorithm in domains with noisy data and sensors is to use probabilistic detectors; in this case a method must be specified for combining the uncertainty from all the unary and binary decisions associated with any given interpretation into a single "goodness" score.[2] The next few sections explore some options for representing uncertainty.

5.2.1 Options for representing uncertainty

Pearl noticed that expert systems that reason with uncertainty break roughly into two classes; he called these extensional and intensional [Pea88]. Extensional systems are procedure-based and have truth values attached to formulas. Examples of extensional systems are n-state logics, certainty factors, and fuzzy logic. Intensional systems model states of the world and attach uncertainty to these states. An example of an intensional system might be a representation that computes via Bayesian analysis. In general, extensional systems tend to be computationally convenient but with semantically sloppy rules. Intensional systems, conversely, offer more semantic clarity, using elastic constraints about the state of the world, but at the cost of greater computational complexity [Pea88].

[2] For example, assume a search has progressed down three levels in a tree, but each unary and binary detector has returned "consistent but with low probability." Eventually, the series of low probability detections should reduce confidence in the score for that particular interpretation.

Extensional probabilistic representation

Extensional systems perform inference incrementally by applying individual rules guided by search heuristics. Each time a rule is applied, it can change the certainty associated with other represented knowledge. N-state logics, which in practice are most often binary-state and sometimes tri-state logics, are widely used and well-understood [RN95]. Non-deterministic search algorithms can be used to infer "explanations" for any particular decision and to provide a sequence of logical inferences to the user if required. First-order n-state logics, however, suffer from several problems when common extensions such as the situation calculus [MH69] are used for temporal recognition tasks: (1) they cannot represent duration, (2) they cannot represent delays between cause and effect, (3) they have no explicit representation of time, (4) they cannot predict natural changes not triggered by actions or events, (5) they cannot represent continuous changes, and (6) they cannot predict the effect of concurrent actions [Taw97]. Although the situation calculus has been extended to deal with some of these problems, even the most efficient representations typically require a computationally-expensive non-deterministic search; unfortunately, practical search heuristics are usually not defined. Probabilistic logics extend the logical rules to include Bayesian probability combination rules [Had91, DHW94]. The rules can be applied incrementally, but this requires an assumption of independence between logical rules that places a design burden on the knowledge engineer.

Another once popular extension system for representing uncertainty is the certainty factor (CF) model [Sho76]. The AI community has largely abandoned CFs, which can be interpreted as measures of change in belief within the theory of probability [HS92]. The CF model implicitly imposes assumptions stronger than simple Bayes. Heckerman has suggested that the problem with the CF model is that it attempts to reason about uncertainty in a modular way, much like a logical system, but that uncertain reasoning must account for conditional probabilities in a non-modular way [HS92]. The CF model does have the following advantages. First, the certainty combination heuristics are computationally simple, leading to efficient propagation of uncertainty. Second, the CF model can be embedded in a logical search framework, making it easy to apply to problems where logical formulations have been proven in practice. Third, the assignment of certainty in the model generally conforms to an expert's intuition. Fourth, the CF framework models the narrowing of a hypothesis set as more evidence is accrued. Finally, the representation can represent hierarchical relationships.

The CF model has several other serious disadvantages, however, as a representation for uncertainty. First, it is deceptively intuitive but actually *ad hoc*. The designer is therefore lured into thinking the representation is correctly representing uncertainty in a normative way. However, results can be obtained using the CF representation that are contradictory and that violate Bayesian reasoning [HS92]. The second disadvantage is that the transitivity of rules are unjustified and lead to the model allowing (and sometimes encouraging) the representation of inconsistent knowledge. The third disadvantage is that assumptions of conditional independence are hidden from the user. Rule interdependencies are not explicitly stated but do exist because rules interact with global certainty variables as they are applied. The result is that the rules encode "invisible" facts and rule exceptions not covered in formulas [Pea88]. A fourth problem is that the CF model, while seemingly making uncertainty assessments easier for the knowledge engineer, actually adds additional modeling complexity. For example, some have argued CF assignment is more difficult than assignment of conditional probabilities [HS92] because the CF method assumes that evidence supporting the same explanation is independent in order to avoid an explosion in the number of rules required for evidence combination. As a consequence of this assumption, however, the knowledge engineer must carefully structure sets of rules to maintain this independence as rules are applied. Finally, encoding knowledge in the CF framework can be difficult because the model is unable to accommodate the tendency of experts to reason about abstract entities before reasoning about a single hypothesis [GS93].

A third class of extensional probabilistic representation for uncertainty consists of fuzzy logics, which model degree of belief and degree of truth. Fuzzy logic models suffer from many of the same disadvantages as the CF models; they can produce result that are inconsistent with first-order logic [RN95]. Fuzzy logic systems have generally been applied to problems that require only a few inference steps, thereby reducing the likelihood a logical inconstancy will be encountered; hence, the scalability of fuzzy logic reasoning is in question (e.g. see [Elk93]).

Intensional probabilistic representation

Bayesian probabilistic inference systems avoid some of the problems with CF and fuzzy logic propagation methods. The primary reason to adopt a Bayesian framework is that the resulting decisions will be normative given appropriate estimates of prior and conditional probabilities. The propagation of probabilistic information uses a widely-accepted, well-understood mathematical formalism. For problems with large datasets, some priors (i.e. $P(explanation)$ and conditional probabilities (i.e. $P(evidence|explanation)$) can be statistically estimated from large example databases, reducing the number of quantitative estimates required of the knowledge engineer.

In practice, several assumptions must be made to implement Bayesian inference in real domains. First, most evidence is assumed to be independent to avoid an explosion in the number of evidence combinations which is exponential in the number of variables. Second, to accurately estimate relative probabilities of hypotheses, all possible world states are assumed to be modeled. Third, the models typically assume mutual exclusion of world states. Fourth, the models assume that the model has been provided correct and complete statistics, where probabilities are internally consistent. Fifth, any piece of evidence information used by the system is generally assumed to be counted only once.

The problems with applying the Bayesian model are as follows. First, there is no notion of preciseness of a system's output probabilities. For instance, when no evidence has been observed, a Bayesian system might compute that the probability of some hypothesis being true is .5 based on priors. However, in actuality, the system has no evidence upon which to base this decision and the user has no basis upon which to evaluate whether .5 was obtained after evaluation of a massive amount of evidence or if .5 was obtained after evaluation of little or no evidence. The second problem is the burden placed on the knowledge engineer of assessing large numbers of conditional probabilities. Although these assessments may not be any more difficult than the assessment of CFs or fuzzy values, the task is still tedious, difficult, and prone to error. A third problem is that the numerical precision an expert must specify to use a Bayesian reasoning system can be unrealistic to obtain in most real-world applications; the mathematical precision of the propagation of uncertainty is typically overwhelmed by the imprecise estimation of probabilistic information when modeling real problems. Fourth, some knowledge engineers believe that experts tend to think in terms of degrees of certainty rather than actual probabilities, making elicitation of probabilities challenging. The difficulty of probability estimation is compounded because conditional probabilities are not modular (e.g. $(P(H|E)) \rightarrow P(H)$ if E *and* E *is all that is known.*). Experts will introduce inaccuracies into the probability assignment as they attempt to assume conditional independence. Finally, the Bayesian framework is not straightforwardly embedded into a logical framework, and so the method is not conducive to incremental searching.

A second intensional probabilistic representation is Dempster-Shafer evidential reasoning [Sha76]. Bayesian inference computes the probability of truth given evidence. Dempster-Shafer theory computes the probability of a probability given evidence. Intuitively, the method provides a measure of confidence in some statement based upon the amount of observed supporting evidence for the statement (or conversely, the amount of ignorance).

The Dempster-Shafer method makes the following assumptions. First, it computes $P(explanation|evidence)$ instead of $P(evidence|explanation)$. $P(explanation) = X$ indicates the probability that the explanation lies between $X\%$ and 100%.
$P(evidenceAgainstExplanation)$ is represented by the probability for the complement of the explanation, which is the set of all other explanations. Second, independence of evidence is assumed except where pieces of evidence support the same explanation. Finally, mutual exclusion of explanation states is assumed, as are correct and complete statistics.

The Dempster-Shafer model has the following advantages. First, it generates uncertainty intervals around each hypothesis probability. Second, the model naturally handles the narrowing of a hypothesis set as more evidence is accrued. Third, it is conducive to the representation of explanation hierarchies and hierarchical relationships.

The primary disadvantage of using Dempster-Shafer theory is its computational complexity. Computation where there are N possible results requires the consideration of 2^N sets. Exponential calculations are required for evidence source combination [Orp90]. The method can be computationally simplified if the data is restricted to a tree dependency hierarchy, but at the cost of lost generality [SL87]. Like Bayesian inference, the basic model does not account for the possibility of multiple, simultaneous explanations. No formal theory is used in practice for decision making because the notion of utility is not well understood [RN95]. Finally, estimating $P(explanation|evidence)$ is typically more difficult than estimating $P(evidence|explanation)$ (for the knowledge engineer or automatically) because databases are normally sorted by the major class (e.g. diagnosis) [GS93].

Pearl has argued that evidential reasoning such as Dempster-Shafer reasoning can be accomplished within the Bayesian framework [Pea86]. Probability intervals can be generated in a Bayesian framework by evaluating how a belief would change as the result of future observations [RN95].

5.2.2 Bayesian networks: a compromise

Bayesian networks, also known as belief networks, are graphical representations of probabilistic distributions (see [Jen96] for an introduction). Links in the graph represent dependencies between the nodes, which represent random variables in the problem domain. Algorithms that exploit the independence relationships encoded by the graphical structure can (often efficiently) compute the impact of observed evidence on the given distribution. The mathematics of Bayesian network uncertainty reasoning are based upon restricting the structure of graphs to enforce a Markov assumption between variables – variables are only conditionally dependent upon variables from which they are explicitly linked. The Markov assumption and the corresponding graph structure make Bayesian inference practical for real problems; just as importantly, the graphical representation aids the knowledge engineer in the specification of domain knowledge and leads to knowledge bases that are easier to analyze than those encoded using logic-like rule sets.

Bayesian network reasoning has several desirable properties:

Normative. A system using appropriately constructed Bayesian networks will generate Bayesian results that are consistent with how people wish they could behave when evaluating decisions [Hec91].

Dependencies explicit. One of the problems frequently encountered with rule-based systems and encountered in the formation labeling system in Chapter 4 is that dependencies between rules are not explicit. Bayesian networks *force* the knowledge engineer to make all variable dependencies explicit in the network's structure. The complexity of a particular domain can be assessed by analysis of the graphical structure itself. Further, the graphical design of the knowledge base can expose expert inconsistencies during knowledge encoding. The graphical representation is a cognitive simplifier for the knowledge engineer.

Computationally efficient. Propagation of uncertainty in Bayesian networks is efficient because algorithms use the dependency information (or more specifically, the independence assumptions) specified in the graph to minimize extraneous computation. For many tasks, networks can be constructed that can be solved quickly and by direct methods [RN95].

Off-line initialization. Once a graph structure is determined, the NP-hard computation that manipulates the graph structure for efficient uncertainty propagation (i.e. triangulation) can be performed off-line. Propagation of probabilities upon receiving new evidence is then $O(NM^C)$, where N is the number of nodes, M is the maximum number of states in any node, and C is the size of the largest clique. Appendix E describes how to compute the size of the largest clique, which is dependent upon the particular structure in the graph.

Mathematically grounded. Propagation of uncertainty in Bayesian networks operates using principles from Bayesian theory. Networks can be designed to take input from noisy sensors and propagate the sensor uncertainty to other random variables. Bayesian mathematics is well-accepted and well-understood.

Continuous or discrete input. Bayesian networks can be designed to use input from discrete sensors (e.g. inView(object)) or continuous sensors (e.g. distance(object1,object2)).

Evidence computation. Bayesian networks implicitly encode default evidence information and can operate when only partial evidence is available. Sometimes evidence is not available, conclusive, computable, or worth computing.

Utility theory. Utility theory is a well-understood mathematical formalism for using Bayesian networks for optimal decision-making given evidence and probabilistic prior cost functions [Jen96].

The useful computational and representation properties of Bayesian networks were popularized in the AI community by Pearl [Pea88], and the one of the first uses of the networks in a large, real application was the Pathfinder diagnosis system that modeled 100 features and 60 diseases [Hec91]. Prior work that uses Bayesian networks for object recognition and action recognition is discussed in upcoming sections. Appendix E includes a brief discussion on the complexity of exact propagation of uncertainty with Bayesian networks using the Jensen algorithm [Jen89, Jen96], which is based upon the graph theory work of Lauritzen [Lau96, LS88] (see [Cha91, Jen96] for other introductory material).

In practice, belief networks are typically limited in size to several hundred nodes for the following reasons: (1) exact propagation of probabilities is NP-hard for networks with arbitrary structure [Coo90], (2) approximate propagation of probabilities is NP-hard for networks with arbitrary structure [DL93], (3) the knowledge engineering task of specifying prior and conditional probabilities is manageable, although often difficult, for networks with only several hundred nodes (e.g. similarity networks can be used [Hec91]), and (4) existing learning algorithms require complete and error-free datasets for learning conditional probabilities automatically [Hec95]. Even with only several hundred nodes, a sparse and tree-like linking structure is generally needed to make the networks practical for real domains, and the nodes in the networks are usually limited to a small number of states to minimize the number of probabilities that must be specified (e.g. one of the larger networks used in practice has 448 nodes and only 908 links [PPMH94]).

Sometimes algorithms are applied to networks to reduce the network structure by creating "approximate" networks that, for example, set rare states to zero probability to compress the corresponding conditional probability tables [JA90]. Similarly, "sensitivity analysis," can be used to analyze networks and detect node and link combinations that result in uncertainty propagation that is overly sensitive to minor variations in input data or nodes that have little affect [JCNJ91].

Once a Bayesian network is specified, a solution method can be selected that will compute the impact of any new evidence on the probability distribution for any random variable. However, the design of networks dramatically affects computational performance and usefulness of the representation. The network knowledge engineer, therefore, faces several challenges:

Selecting variables. A set of random variables to represent states of the world in the problem domain must be selected. Some of these variables are sensor outputs (e.g. *InContact(object1,object2)*), and some are internal "beliefs" of the agent being modeled (e.g. *ReadyToBlock(object1)*).

Selecting state spaces. Each random variable has a discrete or continuous state space. Most complex networks use discrete states to simplify the knowledge engineering tasks; otherwise, a large number of multi-variable distributions might need to be specified. For example, in the football domain the variable "speed" is broken into these states: {stopped,walking,jogging,running,airborne}.

Assigning probabilities. The designer must typically estimate hundreds or thousands of prior and conditional probabilities that are often not possible to obtain from datasets. To reduce this onerous burden, the use of qualitative Bayesian networks has been proposed; however, the discriminatory power of qualitative uncertainty propagation diminishes as the complexity of the problem domain increases (i.e. the outputs tend towards the mean) [HD91, Dru96].

Managing large networks. Even small belief networks can be difficult to construct so that they exhibit the desired, robust performance. The networks must be checked for errors such as inconsistent conditional probability tables that result in impossible conflicts given some particular set of observations. In practice, networks of just 30 nodes can require hours to construct and test; networks over about 100 nodes are a significant knowledge engineering challenge. In practice, large networks require design approximations such as using similarity networks [Hec91], generalizations of "noisy-or" constructs [Pea88, PPMH94, Sri92][3], and modular network partitions.

Avoiding cycles. Efficient propagation in Bayesian networks to compute an exact solution is achieved by restricting the graphs to be directed, acyclic graphs (DAGs).

[3]Some networks can be substantially simplified using generalizations of the "noisy-or" approximation that reduces the number of probabilities required for an n-ary conditional or from $O(2^n)$ to $O(n)$ by making an independence assumption [Pea88].

This requirement imposes modeling restrictions that must be overcome using design heuristics. The knowledge engineer can carefully chose the direction of variable dependencies to maintain a non-cyclic network and to control which conditional independencies and priors must be provided. Links are typically interpreted as indicating causality. A directed edge from A to B is typically interpreted as "A causes B."

Using learning. Until recently, learning algorithms for Bayesian networks assumed an existing network state and linking structure and used a large, complete, and error-free dataset to learn priors and conditional probabilities for the network. Recent work has focussed on modifying an existing structure. Developing algorithms for learning network linking structure given the relevant random variables has proven difficult – even for complete, error-free datasets [Hec95].

Continuous variables. In practice, moderately large Bayesian networks are designed with random variables that have discrete state spaces. Continuous concepts (e.g. distance or velocity) are typically quantized into a small set of discrete states. The Jensen exact propagation algorithm can be extended to handle continuous variables; an exact solution can be computed when a network has continuous nodes with discrete parents but not when the network has discrete nodes with continuous parents [Lau92, Ole93]. The continuous variables can reduce the complexity of some inference but at the cost of increasing the complexity of the triangulation step; therefore, the computational complexity of the discrete and continuous solutions are, in practice, almost equivalent.

Leak probabilities. In practice, it is rarely possible to model all variables that can possibly impact a given node's state. Consequently, practical systems typically compress possible but low likelihood states into a single state called a "leak" state, which can improve network performance substantially [HPF+96, PPMH94]. The knowledge engineer must assess if some variable must be explicitly represented or incorporated into a leak state.

Unfortunately, Bayesian network construction is as much a "black art" as it is a science. Empirical studies have shown that it is more critical to get the qualitative linking structure correct than the quantitative probability assignments [HPF+96]. Assignment of dependencies between variables is probably the most critical step in network construction. For example, if a problem can be modeled using singly-connected networks, it can be solved in time linear to the number of nodes [Cha91]. A fully-linked network modeling the same problem may require NP-hard computation and an exponential increase in the number of conditional probabilities to assign.

Studies have consistently shown that networks with a reasonable structure encoding some domain problem are relatively insensitive to most of the assigned conditional probabilities [NA91, HPF+96, PHP+96]. In fact, in practice, most conditional probabilities can be off by as much as +/- .1 without degrading the performance of the exact solutions for a network [RN95].[4] The exception are probability values close to 0 or 1 which represent "absolute certainty"; networks have been shown in practice to be sensitive to these values [NA91]. To simplify probability assignment. some work has been done on correlating English quantifiers (e.g. "very" "some") with changes in probabilistic values [EH90]. In addition, research and practice suggest that binary nodes are as effective as nodes with more states for many problems [PHP$^+$96], and binary nodes can offer significant computational savings. Representing action using Bayesian networks requires representing time either in the links (transitions between random variables) or the random variables themselves. Some temporal representations that have been proposed will be discussed in an upcoming section.

Overall, Bayesian networks can be a computationally-efficient representation for propagating uncertainty, but the weaknesses above must be overcome. The structure of the networks is the most significant factor determining how effectively a network can represent some problem and how computationally practical the network can be for a real domain. For a given type of action recognition problem, therefore, a structure must be established that is powerful enough to represent the problem at hand but has a structure that results in practical computation time.

5.3 Bayesian networks for object recognition

Bayesian networks have been used in computer vision systems for integrating uncertain information into object recognition approaches. This section describes some of the network structures that have been proposed.

5.3.1 Part-of models

Belief networks can be used to represent decomposable part object models for visual recognition of objects in static scenes [Lev86, BLM89, Che90, MB92, SG92]. The networks can naturally integrate high-level knowledge about expected states of the world with low-level detected evidence, permitting top-down and bottom-up evidence to be used simultaneously for inference. Belief network models can then be used to rank order possible hypotheses.

The simplest part-of model consists of a tree structure, shown in Figure 5-4a. S^i represent state random variables for the problem domain and E^i represent evidence random variables for the corresponding state variables. The root node contains states representing all possible hypotheses and sometimes an "other," or leak, hypothesis. The root points to nodes that represent the existence or non-existence of particular parts of the root node's object. Borrowing an example task from [RB94], assume the example shows a network used to recognize place settings (discriminating a continental setting from a British setting), nodes might have the following states: S^1 = "Setting" with states $\{$continental, British, none$\}$, S^2

[4] For exact algorithms, performance is determined by the network topology but for approximation algorithms the performance is determined by the network's assigned probabilities [Cha91].

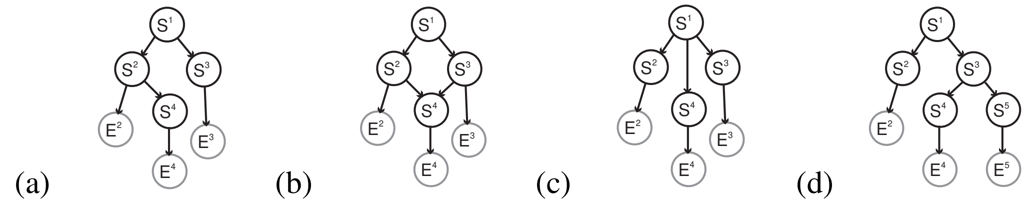


Figure 5-4: (a) A singly-connected tree part-of network, (b) a part-of network where one part, represented by S^4 is dependent upon two other parts, (c) a naive Bayesian part-of network, (d) a hierarchical part-of network.

= "Place setting" with states $\{$continental, British$\}$, S^3 = "Pot" with states $\{$coffee, tea$\}$, and S^4 = "Cup" with states $\{$coffee, tea, none$\}$.

The object-to-part tree structure can be interpreted either as "the whole causes the part," or "the object's interaction with the sensor "causes" the percept" [BLM89]. All sub-components are conditionally independent given their parent component. This independence assumption generates a singly-linked tree structure that is computationally efficient (linear in the number of nodes in the tree) [Pea88].

The same graph structure with nodes based on aspect graphs can represent a 3d object and its part components, where nodes represent generalized cylinders, faces of 3d components, and contours [BLM89, LJC96, MF92]. For complex objects, different components of the object may "cause" the same subpart, which breaks the singly-linked tree structure, as shown in Figure 5-4b. Even for the place setting example, this is probably a better linking structure since the type of cup is probably related to the type of pot on the table *and* the type of place setting. In a third alternative structure, a naive Bayesian network, each observed variable (or class attribute), $\{S^2, S^3, S^4\}$, is assumed to be conditionally independent given the class, S^1, as shown in Figure 5-4c. Even for this simple example with only four state random variables, it is not clear how to best model the problem, which presents a design challenge for the knowledge engineer.

These top-down network structures makes it difficult to encode dependency on global recognition variables, such as spatial location or object orientation. Further, note that the relative importance of object components is hidden in the numeric differences in the conditional probability tables; these numbers have typically been manually assessed in vision applications. Finally, the representation assumes hypotheses are mutually exclusive.

The number of nodes required to represent a real domain can grow quickly. One domain, ship identification from side and top silhouette views, required using 598 rules (i.e. nodes) and 1000 links for a 10 class classification task [BH88]. A full 640 class system would probably require significantly more links and nodes.

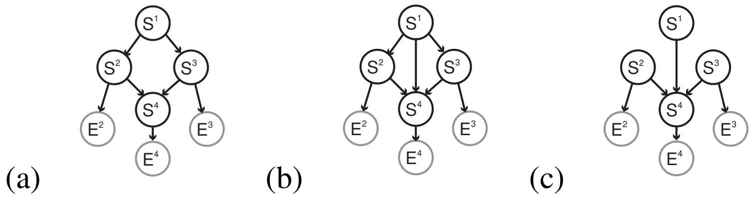


Figure 5-5: Constraint links.

5.3.2 Part relationships

Modeling object descriptions that include the relationships between object parts (e.g. spatial, size, exclusivity) requires special node and link structures in the part-of networks. One method adds subpart nodes that detect relationships between other subparts [BLM89]. For example, in Figure 5-4d, if S^4 represents "PartA" and S^5 represents "PartB," then S^3 might represent "PartA nextTo partB" and S^2 might represent "Size PartA." This model assumes that the un-modeled dependence between S^4 and S^2 will not affect the value returned by S^1.

Sometimes two parts are mutually-exclusive and an *xor* logical relationship needs to be encoded in a graph using a constraint node. This constraint node can be linked into the graph in several ways. Figure 5-5 shows three possible linking models. In all three representations, node S^4 must have a conditional probability table designed to enforce mutual exclusivity between S^2 and S^3 when S^4 is observed. The model in Figure 5-5a can only enforce a constraint between two parts, but it cannot explicitly use the certainty of the *xor* constraint to support S^1 [JCN92].

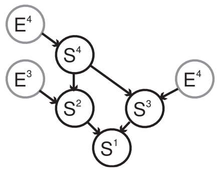
Alternatively, a link can be added from S^1 to S^4, as shown in Figure 5-5b, linking the constraint directly back to the parent subpart (e.g. [SG92]). This extensions permits uncertainty about the observed relationship to influence degree of belief in the parent subpart. The added link, however, increases the complexity of the network and forces the knowledge engineer to specify incorrect information when assigning some probability values (e.g. $P(S^2|S^1)$).[5]

The graph in Figure 5-5c can also model the mutually exclusivity constraint but requires the knowledge engineer to specify priors on S^2 and S^3. For more on modeling mutual exclusivity, see [Pea88].

5.3.3 Constraints in nodes vs. constraints on links

Relationship information can be entered into a network via detector nodes or they can be modeled in the conditional probabilities between node states. The part-of models generally have states that store $\{$observed, NotObserved$\}$ information. Consequently, the

[5]Further, many of the probabilities that must be specified are non-intuitive.

(a) 

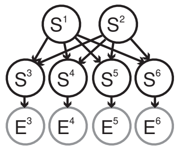
(b) 

Figure 5-6: (a) Inverted object recognition network, (b) example structure for a multiple hypothesis network.

only constraint that can be modeled using links is mutual exclusivity or mutual observation. However, if the observation nodes store object characteristic information (e.g $\{$bigCup, smallCup, noCup$\}$) then relationships can be encoded in conditional probability tables. For instance, given cup detectors that store cup size, a detector constraining one cup part to be "bigger than" another can be encoded within a link. However, adding additional nodes to states increases the complexity of inference [Jen96]. Although encoding relationships on links is common for temporal relationships (and will be discussed later), static object comparison information has typically been encoded within nodes.

One problem is that representing global variables (e.g. position) and relational data can explode the complexity of a network because nearly every measured feature may depend directly on the global variables. Agosta has proposed that "local relationships – exclusion, co-incidence, proximity, and orientation – can be exploited that do not require "completely global" influences" [Ago90] and therefore avoid this additional link complexity. The approach of separating inference about global variables like position from other part knowledge and only representing *relational* information in a network, primarily within nodes, has made it feasible to use standard, exact propagation algorithms for some recognition tasks where they otherwise would have been impractical [DKB98].

5.3.4 Inverted networks

Networks with a top-down, object-to-subpart structure make the assumption that the sub-parts are conditionally independent given the part. For instance, in Figure 5-4a, where S^1 is the setting, S^2 is the plate, and S^3 is the cup, given the observation of a British table setting, observations of the plate and cup are statistically independent. To more accurately model the dependence, the structure of the part-of graph can be reversed, as shown in Figure 5-6a [SB93], which makes the sub-parts conditionally dependent given the encompassing part; low-level part concepts are dependent if the high-level concept is known [SB93]. Tractability of this inverted structure is maintained by exploiting independencies among variables, many of which arise because "features that are dependent tend to be close together spatially" [SB93]. To minimize network complexity, *composite nodes* can be used to reduce the number of links in a graph at the expense of increasing the number of states in some nodes. For example, composite nodes can be used to represent a feature that can occur at

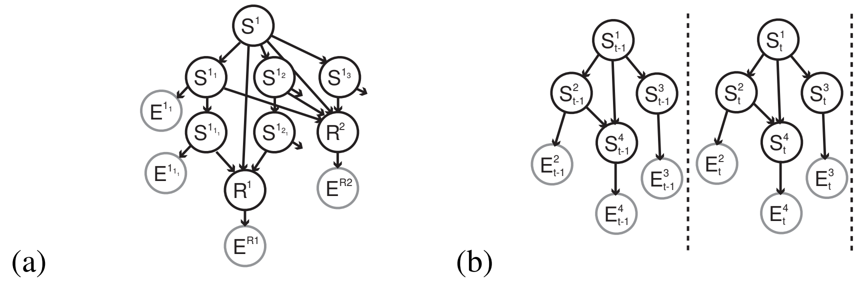


Figure 5-7: (a) Network with constraint nodes, (b) copying a static BN from time t-1 to time t.

several different locations.

In this evidence-to-part framework, priors are specified for the occurrence of each type of evidence. Equal priors are assumed when no information is available. Usually, however, it is easier to estimate $P(evidence|S)$ than $P(S|evidence)$. This framework makes some other assumptions that are unrealistic for visual processing. First, the priors are $p(evidence)$ instead of $p(object)$. Estimation of $p(evidence)$ requires computing the evidence over all possible inputs (or a large representative sample) and for many features will result in numbers approaching 0. Although networks are typically insensitive to small changes in the probability assessments, the networks are highly sensitive to numbers close to 0 or 1 [NA91]. Further, a large, labeled database of objects and subparts can be used to accurately estimate $p(subpart|part)$ whereas $p(part|subpart)$ and $p(subpart|evidence)$ cannot be accurately estimated from a modest database that labels parts and subparts.

5.3.5 Multiple networks

Networks with a single hypothesis node enforce independence between the top-level hypotheses [BLM89]. Part-of networks with multiple hypothesis nodes, however, like the network shown in Figure 5-6b, can model exclusion and co-incidence of hypotheses [Ago90]. Another way to account for multiple hypothesis is to use multiple detection networks. While sometimes it is possible to encode all relevant information in one network, independent multiple networks have been used to integrate different types of knowledge during a recognition task [RB92, RB94]. When evidence is obtained, it is propagated in each network independently and then the resulting probability is used as evidence in a composite network.

Another network composition approach for object recognition has separate networks for recognizing 3d objects, CAD primitives, aspects, faces, and contours [LJC96]. Binary spatial relationships (in particular, adjacency) are modeled using constraint nodes, as shown in Figure 5-7a. For example, R^2 is a constraint limiting S^{1_1} and S^{1_2}. Small component networks are run and results are input into the next network in the hierarchy using likelihood evidence. For a 3D recognition problem of complex CAD parts, one large belief network has been found to be too difficult to manually construct. Even the construction of small networks was found to be tedious and error-prone [LJC96], but the problem is alleviated somewhat by using CAD models to automatically generate some conditional probabilities ($p(aspectview|model)$). It is worth noting that while the authors point to the relatively small network size as being critical for fast evaluation, the clique size, which bounds the computationally complexity, is highly dependent upon the order of the feature constraints selected. The higher the order of R^2, the less "tree-like" the linking structure of the network and the more likely it is that large cliques will be generated.

Use of multiple networks at different hierarchies in the recognition process has also been proposed [MC93]. Using about 50 features, at each level BNs try to differentiate between possible models. If they can, the networks for the next level are loaded. If not, the most specific result obtained is returned. The same network structure with different conditional probability tables is used to compare across model types, and each modular network returns a binary decision of "isCategory" or "otherCategory".

The input from one network can be incorporated into another network using "virtual evidence," or relative likelihood evidence [Pea88]; however, doing so usually results in an approximate model that does not account for dependencies between variables in the two networks.

Multiple agents can also be modeled using independent networks [JCN92]. In one system, two agents in a multi-agent system communicate information and each try to improve recognizing by maximizing models of internal "benefit." The two agents are an interpreter, which uses a probabilistic network and a findings catcher, which performs image processing.

Overall, even for the task of object recognition, a variety of network structures have been proposed. Each structure makes different assumptions about the problem domain, and each structure places different burdens on the knowledge engineer. The system developed in this work presented in Chapter 6 uses an augmented naive Bayesian structure, where many constraints are embedded in nodes instead of links. Reasons for selecting this structure are discussed in Chapter 7 and 8.

5.4 Bayesian networks for action recognition

Action requires change and change requires time [Taw97]. In this work the goal is to develop a system that can recognize real-world, multi-agent actions from perceptual input. This system must, either explicitly or implicitly, make representational commitments about time and monitor degree of belief in noisy observations and uncertain temporal models.

The previous section discussed network representations used for object recognition. This section examines some Bayesian graphical model structures that encode temporal representations for action recognition. Graphs have proven popular for representing uncertainty over time for several reasons. Most importantly, the world is fundamentally *causal*. Events cause other events. Graphical links can naturally represent the temporal causality of the world.

Non-probabilistic temporal logics are used extensively within the AI community for reasoning about time (e.g. [All83]). These systems, however, are brittle when applied to the noisy data and uncertain models encountered in real-world recognition tasks. Non-deterministic search, logic-based models such as the situation calculus [MH69] or default logic [Rei80] are difficult to extend to noisy, probabilistic, temporal domains. An alternative, extensional approach is to use temporal representations that probabilistically integrate temporal evidence [Web89]. If the world is stochastic in nature, representations of the world will probably need to be stochastic as well [Had91].

Plan recognition systems based on extending Bayesian belief network graphical representations to recognize dynamic activity have been investigated (e.g. [GC90, CG93, PW95, Hub96]). In the computer vision community, work to extend finite state machines to handle noisy input data has led to the use of Bayesian belief networks and hidden Markov models for static object recognition and temporal recognition tasks. All these graphical frameworks, however, share a similar, computational framework.

One way to extend static networks to a temporal problem is by applying static networks independently at each time point in order to acquire temporal information, as shown in Figure 5-7b. Output of networks can be used to set global variables that are used to propagate information though time by setting evidence nodes. Alternatively, time-based functions that change the probability distributions stored in each node can be employed [TN94]. In these approaches, change is modeled externally to the network. Other approaches model the change in state between observation times *within* the network itself. Some of these representations are discussed in the following sections.

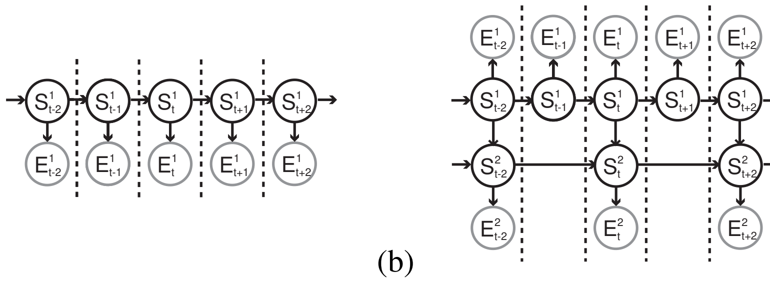
5.4.1 Hidden Markov model networks

A graph is kth order Markov if any given state only links to states at the kth previous time periods. In graphs where time is encoded as the transition between two states, the Markov assumption has proven to lead to network representations that are adequately constrained for efficient propagation of uncertainty but yet still rich enough to represent some temporal activity well.

The justification for making a Markov assumption is that the world is *causal*: states of the world cause other states to occur. States of the world cause changes and changes will (eventually) have observable consequences in the world [Taw97]. If most causal events depend on a relatively small percentage of the states in a particular graph, causal effects can be modeled by sparsely-connected graphs.

A strict Markov model assumes that states are directly observable. For recognition problems, a more useful model incorporates the uncertainty of visual observation by probabilistically modeling hidden states (the states of the world) and observable states (the observations of the states of the world). Hidden Markov models (HMMs) can represent both uncertainty in the world and in observations of the world [RJ93b]. Several comprehensive tutorials are available describing HMM use in practice [RJ93b, You93].

HMMs are simply BNs with a particular linear structure [SHJ96]. Figure 5-8a shows a basic HMM encoded as a BN, modeling a single state changing over time given a sequence



(a) (b)

Figure 5-8: (a) A HMM(1,1) Bayesian network, (b) a HMM(1,2) Bayesian network.

of observable evidence. The states are $S_t^i = \{s^1, s^2, ..., s^n\}$, where s^i is a possible hidden state of the system.

This basic HMM structure has been used extensively for auditory and visual recognition; start and end states are specified and a sequence of observations is entered into the network. Because the structure of the network is a singly-connected graph, the transition probabilities can be estimated from a large data set of example actions.[6] The variability in the training set is probabilistically encoded in the network and, for some problems, can lead to promising recognition results, even in real-time domains (e.g. [You93, SP95]).

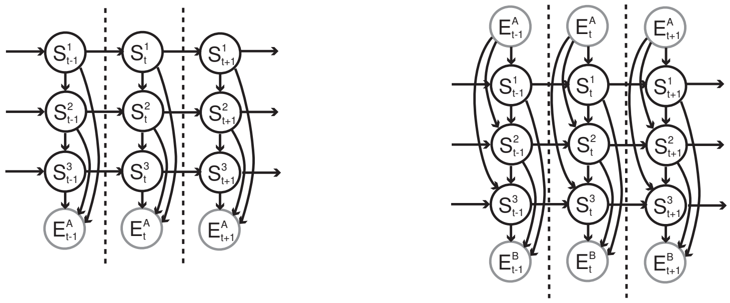
Designers of HMM structures select the features to be used by the model. Then, they typically manually estimate the number of states to model, after which a large dataset can sometimes be used to automatically learn the states and the conditional probabilities in the network using an algorithm that exploits the singly-connected tree structure of the networks [RJ93b, You93]. Essentially, HMMs perform statistical, dynamic time warping, making them effective for recognition of sequential patterns, such as a single-object action (e.g. a hand waving [WB95]).

5.4.2 HMM(j,k)

The HMM model's representational power can be improved, potentially making it more useful for modeling the interaction of multiple agents. An HMM(k, j) is defined in [SHJ96] as a HMM with state memory k and j underlying state processes. The network in Figure 5-8a is the belief network representation for a HMM$(1, 1)$.

Some actions to be modeled have multiple underlying state processes. Figure 5-8b shows an example of a HMM(1,2) network. The network consists of two semi-independent HMM(1,1) models that have been linked periodically. Such networks have been used to model tasks where there are two independent but occasionally linked processes such as lip/speech reading [JJ94] or the interaction between two objects [BOP97, ORP99].

[6] The more hidden states the problem must represent the less likely it is that a complete set of examples will be obtainable.



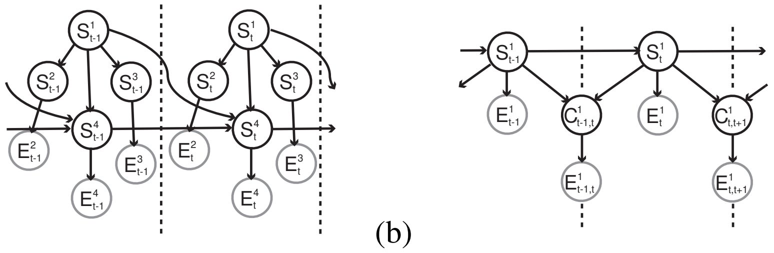
(a) (b)

Figure 5-9: (a) A factorial HMM. (b) Dynamically linked probabilistic decision trees.

HMM(1,2) models have met with some success for visual processing because they can represent two independent but interacting world states that may make them better approximators for certain motions [BOP97]. However, the coupled models do not scale well. Analysis of the clique structure demonstrates that HMM(1,k) for large k is intractable [JJ94].

An HMM encodes information about the history of a time series in the value of a single multinomial variable (the hidden state). To represent 30 bits of information about the history of a time sequence, an HMM needs 2^{30} distinct states, but a HMM with a distributed state representation could do the same with 30 binary state variables [GJ97]. This observation led to the factorial HMM representation shown in Figure 5-9a [Gha97]. Each state variable evolves independently of the remaining state variables according to some dynamic model; the state variables are then "dynamically mixed." Like the HMM(1,k) representation that also tries to model semi-independent processes, this network structure is intractable. The network in Figure 5-9b removes the factorial HMM assumptions that each variable is *a priori* independent of the other variables by coupling the variables within the same time frame. This coupling results in a set of probabilistic decision trees that are dynamically linked [Gha97]. This structure will model multiple resolutions of both temporal and spatial structure but is also intractable [JGS97]. For HMM$(1, k)$ models and relatives, even $k = 3$ is problematic and generally not used in practical systems.

HMM models are often used in practice as dynamic time warping devices. The models have no explicit model of how long a particular state can remain active; the state duration model is exponential, which is not appropriate for many problems [RJ93a]. DHMMs, called duration HMMs or semi-Markov HMMs, are modifications of the standard HMM framework to handle states with expected durations. For some problems where the relative time in particular states is of importance, DHMMs can significantly improve system performance [RJ93a]. However, modeling duration requires D times the storage of a HMM(1,1) and $D^2/2$ times the computation, where D is the maximum time length of stay in a particular state. The estimation problem is also more difficult and requires much larger training sets,



(a) (b)

Figure 5-10: (a) A dynamic belief network, (b) representing evidence of change in nodes.

so DHMM systems are not widely used [RJ93a] (for one example where they are, see [WBC97]).

5.4.3 Dynamic belief networks

HMMs are subclasses of a more general network model commonly called dynamic belief networks (DBNs), which are independent belief networks that are causally linked in time [RN95]. DBNs integrate classical time series analysis with BNs. The result is networks that can represent rich static and temporal relationships and perform some temporal forecasting [DGH92]. DBNs were motivated by the observation that it is sometimes difficult to evaluate time-series processes for models with large numbers of variables; therefore, "due to certain conditional dependencies among variables, it may make more sense to model the temporal evolution of only the subset of variables which are in fact evolving, and use these processes to drive the changes in the dependent variables" [Pro93]. As with HMMs, typically the networks are sparsely linked. They are most often manually coded. DBNs have been used for action recognition where time dependencies can be adequately modeled using only the change in adjacent input samples.

Figure 5-10a shows a simple dynamic belief network that causally propagates some temporal information between temporally adjacent belief networks. The networks model the conditions at each time using a standard belief network. The network is "copied" at each new time, and some state variables are linked from $t-1$ to t. The conditional probabilities model first-order temporal changes. The networks can be solved using an extension to the Jensen and Lauritzen junction-tree algorithm [Kja92, Kja95] that "folds in" past time slices of the network and allows the size of the active window to be adjusted dynamically. These networks have been termed *semi-static* recognizers because recognition occurs in static frames using clues from observations from previous time frames [SG92].

Temporal prediction can also be performed by scrolling a state-based BN model one time step (adding a new network for the new frame) and using parameterized functional forms for conditional probability assessments [DGH92]. The DBN structure is fixed over time and, depending upon the exact structure of the network, some networks may require approximation solution algorithms [DG93].

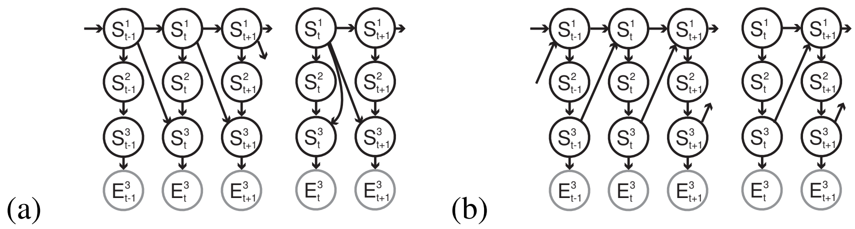


Figure 5-11: (a) Temporal "roll-up" example, (b) a "temporally-invariant" network for roll-up.

An alternative DBN model is shown in Figure 5-10b. Here the network uses evidence that indicates a particular state as well as evidence that explicitly indicates a *change* in state. This has been termed *dynamic recognition* because explicit information about change in state is required for recognition [SG92]. All change is no longer represented by the link-based conditional probabilities.

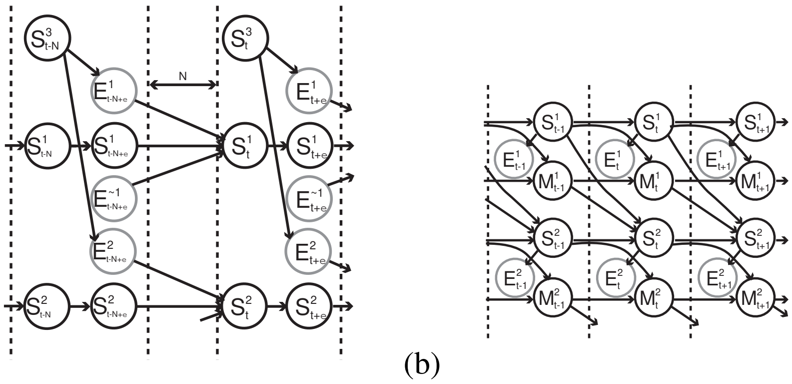
5.4.4 Rolling-up time

When implemented for recognition, HMMs are typically trained for a fixed length of observations with an explicit start and end node. The same technique can be used with a DBN. However, for some networks, the structure of the networks (given a long T, $0 < t < T$) leads to large networks that are computationally demanding. Some work, therefore, has examined networks that "roll-up" over time. As time progresses, copies of the belief network are added to the front of the DBN and copies are removed from the end of the DBN. The result is a network that can be run continuously and that operates over a window of time.

Figure 5-11a illustrates the rolling-up process and how the structure of the network affects the structure of the rolled-up network. In the example, a new link from S_t^1 to S_t^3 must be added when the $t-1$ network is rolled into the network because S_t^1 and S_t^3 are both dependent on rolled state S_{t-1}^1. Figure 5-11b is a "temporally-invariant" network [FHKR95] which is designed so that a roll-up can occur without changing the link structure of the resulting network, thereby enabling fast re-computation.

Most network models copy identical node structure at every time point and model the change between every measured time point (i.e. typically 30 Hz, the frame rate). An alternative model only instantiates evidence into the network at "important" time points [HM94]. Figure 5-12a shows one such network, where the time between copied networks, N is variable. The major events and the (nearly instantaneous) consequences of those events are modeled each time slice. An independent process is required to detect the occurrence of the important events and trigger network expansion and roll-up.

In Section 5.4.2, a DHMM model was discussed. The implementation of this model requires *memory* of some global or semi-global variable. In a dynamic belief network, memory can be implemented by adding a memory state, or "history nodes," for each variable



(a) (b)

Figure 5-12: (a) A network for a DBN that rolls-up only at "important" time points. Evidence is observed immediately after the important time point. Time interval N changes based upon when important events are detected. (b) Incorporating memory nodes into a DBN.

that is copied from state to state, remembering previous values and possibly influencing some network nodes [Kan92, NB94]. Such nodes can model information like validity in certain sensors or "how long an expert has been doing the same thing" [NB94]. Figure 5-12b shows one such network, but incorporating memory nodes within networks is inefficient because a single variable can require the addition of a large number of links.

Overall, a variety of structures have been proposed – each with representational and computational tradeoffs. In this work a new type of structure is developed in Chapter 6 combining attributes from some of the networks discussed in this section.

5.5 Action recognition

This section reviews some prior work that explicitly addresses multi-agent action recognition.

5.5.1 Multi-agent plan generation

Some of the earliest work in multi-agent systems investigated simulation, not recognition, of collaborative multi-agent activity. These systems model communicative acts between planning agents that result in coordinated action. For example, one system uses STRIPS-like plan conditions to generate synchronization communication actions in collaborative plan generation [Geo83]. Another system accomplishes team modeling for military air mission modeling using a distributed multi-agent reasoning system, dMARS, that employs first-order logic with a team hierarchy [TSH95]. Yet another system for modeling agents with team behavior for a synthetic battlefield uses the SOAR system [JCG$^+$97] with a hierarchical task-decomposition and partial-order planning. Developing dMARS required ten work years of effort to construct, demonstrating the complexity of the rules required to model agent interaction. Simplifying the manual construction of the knowledge base is mentioned as an important focus of future work.

An alternative approach is to have the simulated agents broadcast broad behavior details instead of specific details about their planned actions [DM90]. When one agent receives behavior information from other agents, it anticipates the behavior of other agents, using a behavior hierarchy, in order to determine which agents to interact with. In a similar approach, agents predict other agents actions given knowledge of their sensory input and make decisions based on that using a recursive modeling method (RMM) [VD95]. One system, based upon an intentional model of teamwork [CL90], models individual and team plans by having agents reason about joint goals and intentions [Tam97b]. Agents assume other agents have identical sensing capability [Tam97a]. Since communication is sometimes not possible, the author states that each agent needs the capability to visually recognize the plans of other agents, but this is left for future work.

Overall, this and other work on generating team behavior [JCG+97, TSH95, STW+92] highlight two key features that must be modeled to simulate team activity. These features are as follows: (1) the ability for each agent to observe other agents and use predetermined plans for coordination and (2) the ability to use agent-to-agent explicit communication [TSH95]. Since these multi-agent simulators have access to the complete state for each agent being modeled, they do not need to implement routines for each agent that recognize the activity of other agents from sensor data. Although often complex reasoning and communication is required to generate *coordination*, recognizing the coordination may not require intentional reasoning.

5.5.2 Multi-agent action and plan recognition

There has been little prior work describing frameworks for real systems that recognize multi-object action with approximate models and noisy input signals. Most multi-agent plan recognition work uses modal logics that can represent joint intentionality but that have not been implemented in realistic domains with noisy evidence detectors [GK96, CL91, RM94].

A few systems have been developed that recognize actions from static scenes or the change between just two time frames. For example, multi-agent soccer actions have been modeled using blob transitions of histogram back-projected blobs [KYA94]. Soccer actions like "shot at left goal" have been identified using color, ball position, and a field model [GSC+95]. These systems, however, only recognize simple actions in specific contexts and probably will not scale well. In general, a stronger visual model or plan model is required that models change over longer time periods.

Plan recognition tasks can be classified as *cooperative* or *keyhole* [Loc94]. A cooperative plan recognition task is one in which the agents in a scene are explicitly generating perceptual cues to help observers infer their plans. For example, during a conversation between two people, the conversant will use intentional cue phrases to help the listener identify shifts in intentional focus [GS86]. Similarly, listeners will use cues (e.g. "uh huh" and head nods) to indicate that they are grasping the speaker's intent. A keyhole recognition task is one in which the agents being observed are making no effort to convey information about their goals. Therefore, the tasks of identifying football plays or identifying action in a kitchen from visual input is a keyhole plan recognition problem.

Early keyhole plan recognition systems used sets of rules and hierarchical temporal decompositions to infer an agent's plan from a set of observations (e.g. see [KA86, WL92, Hon94]). These techniques infer any agent goal consistent with the rule set and the observations. The knowledge engineer must indicate which plan goals are "top-level goals." Search-based systems designed specifically to recognize multi-agent goals and actions outside of probabilistic frameworks [RS88, AFH89, Tam96] are sensitive to noisy data and detectors.

One multi-agent keyhole domain of particular interest is tactical plan recognition for military aviation. Tactical plan recognition differs from typical plan recognition tasks (e.g. recognizing cooking plans) because the scenes of interest contain multiple, adversarial agents. Adding additional complexity is the uncertainty and size of the input set and the time-critical nature of the recognition task [AFFH86]. One early plan recognition system used the Plan Recognition Model (PRM) [AFFH86] that consists of plans expressed as hierarchies and deterministic finite automata. Heuristic evaluation rules and a best-first matching algorithm are used to identify and rank plans. The PRM model has been extended to handle multi-agent goals [AFH89]. In the extension, each agent is assumed to be pursuing action in support of the higher level goals of a shared "master" agent. The plan descriptions encode goals (e.g. destroy-target), multi-agent missions (e.g. attack-mission), and single-agent plans (e.g. strike-target). Missions are scripts with required and optional components. For example, the destroy-target goal has a multi-agent attack-mission with required plans such as strike-target and optional plans like tank[7] and feint[8]. The STRIPS-like plan scripts are used with a heuristic measure of belief and disbelief, but all incoming data is assumed correct, all agents are assumed to be operating under the same goal, and the system's performance is not characterized on a real example.

The REPLAI system was developed to recognize intentional action in multi-agent domains [RS88]. The input to the system is events such as "run," "pass," and "have ball" that are assumed to be provided by some pre-processing system. A plan hierarchy, where each node in the hierarchy is a goal with precondition attributes and a deterministic finite automata, is used to represent action. Computational complexity for the domain with 11 agents is minimized using a "focus handler" that identifies "interesting' players using a set of heuristics (e.g. have-ball, near the goal) and then adds players who are likely to interact with agents already in the focus. The system performs top-down recognition on the hierarchical goal structure using the preconditions of each node. A database of prior probabilities of a particular agent performing a particular action is used to select when more than one plan is valid for an agent. The system does not explicitly encode information

[7] Here "tank" indicates the goal is to fill up the plane's fuel tank remotely.

[8] Here "feint" indicates the goal of launching a fake attack to distract the opponent.

about the interaction between agents. An extension to this work, REPLAI-II, uses Kautz-like plan graphs with Allen's "meets" arcs and disjunctions and conjunctions. Each node has preconditions, and the graph is automatically converted to a temporally-ordered automaton for processing. The preconditions are used to limit top-down search complexity. The method is critically-dependent upon the (non-probabilistic) preconditions. The system, which forces unrealistic, hard transition boundaries between actions, is most likely quite brittle and does not recognize team activity.

One system for recognition of aircraft maneuvers [RM94] uses "plans as recipes to guide the recognition process and ... plans as mental attitudes to constrain the recognition process." Inference is performed using means-end analysis. The assumption is made that in many domains, given a particular situation, the observing agent will have have access to the set of desires that the executing agent is likely to pursue and may decide to be attentive to these desires. Computation is minimized by considering only a small number of plans, and the system does not use real sensor data.

Tambe and Rosenbloom proposed a method for "agent tracking," defined as plan recognition where the observed agents are engaged in a mix of goal-driven and reactive behavior (since they are in dynamic environments like air-combat simulations) [TR95b]. The model commits at each time step to a particular plan for the observed behavior and uses single-step backtracking, where backtracking is determined based upon the current context. Unlike some other domains, in this domain ambiguities cannot be resolved using communication due to the adversarial task. Heuristics are used to resolve ambiguities whenever possible (e.g. by assuming hostility is more likely than friendly behavior). Overall, the method attempts to "keep tracking firmly tied to the now" in order to recognize action efficiently. The method is extended so that it can handle *two* adversarial agents [TR95a].

Non-probabilistic finite state machine representations have been used in early action recognition systems. Events are described using labeled, directed, hierarchical graphs where links have conditions. One such system output natural language descriptions of activity using spatial referents (e.g. describing an object as next to the circle). The input to the system was not noisy, perceptual data, however [AGR88]. The NAOS system also generated natural language descriptions of traffic scenes from manually-obtained geometric primitives [Neu89]. Another recognition system using FSMs for single-agent recognition in multi-agent simulated domain [Thi86] detected actions like chasing and opening a door.

Charniak and Goldman [CG93] observed the following problems with some of the hierarchical decomposition models. First, the models do not commit to a particular plan explanation as long as there is another plan that could also explain the actions. Second, there is an artificial distinction between top-level goals, which are minimized during plan recognition, and the remainder of the goal actions in the hierarchy. Finally, the strategy of using inference that minimizes the top-level goals recognized is flawed because it is often possible to have multiple causes of the observed actions.

Carberry introduced the idea that a plan recognition system should incorporate a mechanism for reasoning about uncertainty [Car90] to deal with this first problem. Carberry was interested in developing an algorithm that could analyze a discourse and return the conversant's most probable plan, even it is not the only possible plan.[9] A system using the Dempster-Shafer theory of evidence [SL87] was developed so that unwarranted decisions could be deferred until more evidence is available. Dynamically computed preference rules are used to order evidence according to plausibility, and sanctioning a default inference depends on its plausibility with respect to plausibility of alternatives. The use of probabilistic networks is dismissed because the networks provide poor explanation capabilities.

Charniak and Goldman, however, adopt Bayesian networks for probabilistic inference because Dempster-Shafer theory is computationally more complex than Bayesian inference using graphical networks, and making decisions using the Dempster-Shafer formalism usually requires relating the DS intervals to probabilities [CG93]. Both formalisms require the non-intuitive specification of conditional and prior probabilities. Charniak and Goldman's system constructs BNs automatically from English narratives using forward chaining rules relating objects, parts of speech, and prepositions [GC90]. No distinction is made between actions and graphical, probabilistic plans; actions are simply plans with sub-actions.

Huber and Durfee [HD93] observed that in some robotic domains where agents can not explicitly communicate, plan recognition is required for effective collaboration and interaction. They advocate the importance of plan recognition techniques that can handle observational uncertainty and design a robot interception task that uses simple Bayesian networks with noisy input data. This work led to a system for recognizing multi-agent action using belief networks [HDW94, Hub96]. Huber has shown that simple goal recognition belief networks can be constructed automatically from representations of action used by a plan generation system and then used by a planning agent in a multi-object scene [Hub96]. This thesis builds on Huber's work of automatic construction of networks, but none of the existing systems for probabilistic plan recognition have been tested on real data and recognized actions requiring the representation of fuzzy temporal relationships among multiple goals of multiple agents.

Pynadath and Wellman use Bayesian networks to recognize agent plans in the driving domain [PW95]. The recognition networks are designed to "[account] for the context in which the plan was generated, the mental state and planning process of the agent, and consequences of the agent's actions in the world." Plans are thought of as "descriptions of action patterns" and are probabilistic to account for noisy sensing and so that probabilities can be used to distinguish between equally possible but not equally likely plan options. The model uses BNs with about 13 nodes that model low-level driving activity like lane changing. Modeling interaction between agents is not addressed [PW95].

Simulated data of vehicles is used in another probabilistic plan recognition system that uses temporally-invariant DPNs [FHKR95] (see Section 5.4.4). Blob data is extracted automatically from video. Each vehicle is modeled using a separate DPN and the networks each use deictic features computed from the blobs in the scene. Decision making using decision trees leads to behaviors being generated such as vehicles passing slow cars, reacting

[9]For example, assume a person asks how late a supermarket is open. The system should infer that the person intends to purchase groceries even though there are other high-level goals that could be valid given the query

to unsafe drivers, avoiding a stalled car, aborting a lane change, and merging into traffic. One DBN is active per vehicle with fast roll-up [HKM+94]. A similar system, but one that infers longer-term goals, has been developed for keyhole plan recognition in a multi-user dungeon (MUD) online system [AZNB97] (following [LE96, FHKR95]). A DBN recognizes one of 20 goals where each goal can consist of series of actions from a set of 7200 and 47000 locations. Extraneous actions are allowed but goals must be well-specified. The probabilistic structure models the ways that users typically perform to achieve a goal, not actions that necessarily advance the user toward a goal (as would be represented in a more traditional STRIPS-like plan recognition system).

Overtaking, following, and queueing actions from the vehicle surveillance domain are recognized in one system using DBNs and deictic relationships such as "behind," "beside," and "now" [BG95]. The use of an active focus of attention for gathering evidence under some set of expectations is proposed.

Finally, Bayesian networks have been used in pedestrian and car parking lot surveillance system for modeling the interaction between pairs of agents [RTB98]. Each agent has a BN and each two-object interaction (defined by close spatial proximity) has a BN. The networks are less than 9 nodes each. Group states such as "standing next to" as well as single agent states such as "parked" and "walking slowly" are detected.

Devaney and Ram investigate the problem of recognizing group action in real and noisy military training data with trajectories for hundreds of agents. They have had some success identifying simple group "gathering" actions in real time using binary comparisons between all pairs of agents [DR98].

One of the most comprehensive recognition systems has been developed by Nagel *et al* in a series of papers [Nag88, KHN91, KNO94, Nag94, NKHD95]. In this system, three-dimensional geometric models of vehicles are automatically extracted from video of road scenes. The trajectories and orientations of each vehicle are used as input to a recognition system that uses a fuzzy metric temporal logic (see [SN] to recognize motion verbs associated with driving activity such as "follow" and "drive in front of" [Nag94]. Motion verb actions are identified as follows: (1) features are fuzzily quantized using piece-wise linear membership curves (e.g. speed is quantized into null, very small, small, normal, fast, and very fast), (2) features are combined using fuzzy set rules into precondition, monotonicity conditions, and postconditions, and (3) a finite state automaton (FSA) matches observed data to the verb model. Each verb model has its own FSA. Hierarchical "situation trees" can then be defined that represent "possibly occurring situations" in scenes such as intersections and parking lots. Situation analysis (i.e. recognition of action) consists of finding a path through the situation tree [HN98]. The situation graphs represent the typical activity of single agents. This same framework has been used to generate linguistic descriptions of activity [GN98].

5.6 Summary: previous work

The goal of this chapter was to highlight some previous work on object recognition, the representation of uncertainty, and multi-agent plan recognition.

First, analogies between the geometric object recognition problem and the multi-agent action recognition problem were discussed. In the context of the discussion from Chapter 4, the following insight was drawn from work in object recognition: low order (e.g. unary and binary) features used for recognition of static, structured objects might also be useful for the recognition of structured, multi-agent action. The representation described in the next chapter is designed to explore this idea.

Second, representations for reasoning about uncertainty were briefly introduced. Overall, Bayesian networks are found to be a principled, robust, and sometimes computationally practical representation. In fact, the networks have been used extensively for plan and action recognition, but generally not in multi-agent domains. The usefulness of a network as a representation, however, is dictated by the way the problem domain is encoded in the network. The representation described in the next chapter uses Bayesian networks to represent uncertainty. The question is then how to model multi-agent action in the networks so that a structured, multi-agent action can be recognized from perceptual data for a real-world task. One structure is proposed. Practical tradeoffs between using Bayesian networks and other frameworks for representing uncertainty are discussed in Chapter 8.

Chapter 6

The representation

In this chapter a representation for recognition of multi-agent action is developed. The motivation for this representation, which is based on insights from the object recognition and plan recognition literature, was discussed in Chapter 5. Results of using this representation on a database of football play trajectory data of the type introduced in Chapter 3 are discussed in Chapter 7.

6.1 Representational principles

The representation presented in this chapter was developed to overcome some of the problems encountered with the formation labeling system described in Chapter 4. Specifically, the representation is designed to accommodate a noisy temporal input signal. Further, the representation proposed here permits multi-agent coordinated activity to be easily specified by a domain expert and then converted into a Bayesian graphical network formalism that can be used to evaluate the likelihood that the observed data was caused by a particular play.

The representation is based upon six primary representational principles:

1. **Reducing intentional action to visual components** The difficulty of applying existing intentionality reasoning systems to some multi-agent action recognition problems was briefly discussed in Chapter 2. Instead, this work is based on the assumption that for some multi-agent recognition tasks, a system can recognize "what things look like" instead of an intentionally-based description of "what things are." Recognizing team activity does not *always* require that a system represent and reason about the intentional communication and interaction of the agents being observed. Instead, the intractability of modal logic intentionality reasoning systems can be avoided by using a representation that describes structured activity using a small number of temporal and logical connectors linking *probabilistic* visual components. Some multi-agent recognition tasks clearly do require explicit reasoning about intentionality. One example is understanding a multi-agent discourse in which coherent dialogue between speakers depends upon intentional "speech acts" [GS86]. Even understanding some variation in football plays appears to require explicit intentional reasoning, as discussed in Appendix B. However, the representation presented in this chapter may provide sufficient power to recognize some "intentional" action without explicit intentional reasoning. In cases where the representation is inadequate, it may still provide useful pre-processing information for intentional reasoning systems.

2. **Using low-order temporal consistency** The second principle is that low-order unary and binary temporal relationships are sufficient for recognition of some structured action. This observation was drawn from analogy to work done in computer vision object recognition discussed in Chapter 5. 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 [MH69, PR95], temporal modal logics [Sho88], or temporal interval propagation [All84, All91] to enforce complete temporal consistency. For instance, in the framework proposed here, if A is around B and B is before C, the temporal closure that A is before C is not enforced.

3. Using probabilistically-detected "goals" as action primitives If A is *before* B, just what are A and B? The third principle advocated by this work is that 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.

4. Representing uncertainty using graphical models A method is required to integrate uncertainty resulting from noisy data and faulty detectors. The representation proposed here uses Bayesian networks, for reasons discussed in Section 5.2.

5. Using local space-time modularity Any complex problem that requires specification of domain-specific rules by a person will need to allow the knowledge engineer to modularize concepts. This framework uses local space-time modularity to reduce complexity and simplify knowledge engineering. Goal detection networks are designed to primarily consider evidence local in space and time.[1]

6. Deictic referencing The last principle advocated in this work, which has been used successfully in prior work [AC87, FHKR95, BG95] is that deictic, or agent-centered, goal detectors can manage the complexity of multi-agent feature selection. By using deictic features such as "the closest agent" instead of "agent-5," some detectors can be built without an explicit search that matches data to objects.

The remainder of this section describes the details of how these representational criteria are achieved in a framework used to recognize multi-agent action.

6.2 Representational framework overview

The approach consists of the following representational elements:

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

[1] Occasionally the networks do use state information which has been set by the evaluation of other networks that persists through time, such as whether the ball has been thrown. These exceptions are discussed later in this chapter.

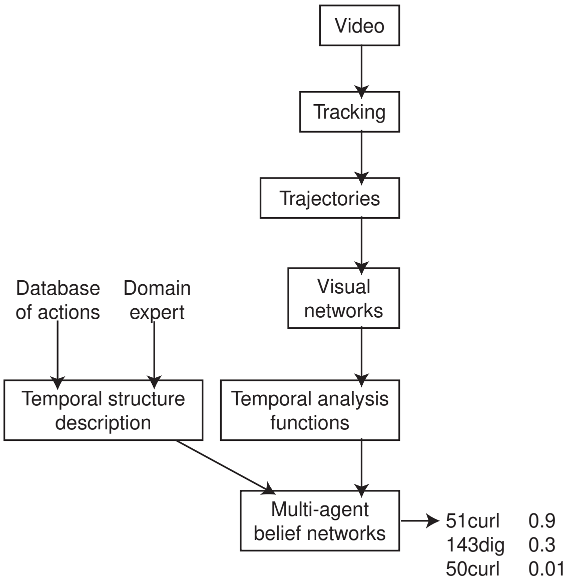


Figure 6-1: The recognition system diagram.

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 large 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 6-1 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. In this work manually-acquired trajectories are used (see Chapter 3). 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 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. The description is used to generate multi-agent belief networks used for computing the likelihood of observing a particular play.

The temporal structure description represents the prototypical scenario of the described action. It 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

One goal of this work is to use prior work in visual object recognition and plan recognition, both discussed in more depth in Chapter 5, to gain insight into the action recognition task. Much of the object recognition work uses image edge segments as primitive input features. Unary and binary constraints on edge segments are then compared with edges of the models in the database (i.e. the model edges). Researchers have found that hypothesized solutions that are consistent over large number of unary and binary checks on edge segment positions are most likely the correct solutions even if higher-order constraints have not been explicitly checked [GLP87]. The question then arises: What are the equivalent of the "model edges" in object recognition for the multi-agent action recognition task?

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. Goals are *defined* by their visual and probabilistic characteristic behaviors, building on work in probabilistic plan recognition [CG93]. The perceptual evidence is evaluated to determine whether an agent has a particular goal at a particular time. For example, the halfback can have the goal of running between the tackle and the guard. To determine if indeed he has such a goal, a recognition system must evaluate the visual evidence, particularly the position of the tackle and the guard and the direction of motion of the halfback. Later, the construction of belief networks that serve as the recognizers of the individual agent goals is described more fully.

There are two reasons that agent goals are selected as the atomic units used for multi-agent action descriptions:

- Agent-based goals are a natural and convenient way to describe ongoing human activity. When watching a football play, someone knowledgeable in the game will typically describe the goals that particular agents are engaged in. For example, a person might say a particular agent has the goal to *block* another agent or to catch a pass (i.e. *catchPass*).

- 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. These observable consequences are of two forms. First, the agent itself may move in a particular way individually. For instance, a receiver may run so as to create a motion pattern of a particular type (e.g. run straight down the field 10 yards, a *streak(10)* action). Second, the agent will change its movement to respond to the contexts established by the other agents as it interacts with them. For many goals, much of this context can be determined by considering the agents in a small spatial-temporal window around the agent of interest. Sometimes the local context is so strong that goals can be detected *before* the primary visual activity associated with the goal. For example, it is possible to detect that the goal of the quarterback is to throw the ball using the relative movement of the quarterback and nearby agents before actually observing the ball leaving the quarterback's possession. Hence, what the system should actually recognize is patterns of motion and contexts (setup by relative agent motions over time) that signal *consistency* with a given goal. Much like the context-based description of natural objects discussed in Chapter 4, goal events can often be detected based on space-time contexts.

6.3.2 Criteria for selecting goals

Unlike methods based upon computational simulation of intentional communication [Pol90b, GK96], 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 (e.g. the quarterback) then detectors for these behaviors are easier to construct, and the application of these detectors is easier to control. Finally, the more certain the behavior is to occur within the current context of the play, the better. Some examples of goal actions for the s51 play are described shortly in Section 6.3.4.

6.3.3 Linking goals with constraints

Given that agent goals are used as fundamental action components, the temporal and logical relationships between the goals can be used to describe a typical example of a multi-agent coordinated action.

Temporal relations

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 has been *observed*[2], whether B has been *observed*, whether A is *before* B, whether B is *before* A, and whether A is *around* B (and therefore, since around will be implemented symmetrically[3], 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 Chapter 5. 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* [All83]. Since Allen's framework does not model uncertainty, any interval that exists is, by definition, *observed*. Here, however, an explicit observed detector is employed to integrate evidence for an action over a temporal window. 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. Ideally, the curves would have distinct cutoffs and plateaus that could be detected using an automatic thresholding algorithm. In reality, the curves are 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 or throwing begins and ends is not clear).

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

[3] 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.

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.

Logical relations

Two logical relationships can be used in the temporal structure description: *xor* and *or*. The semantics are standard: *xor* requires that exactly one subset of a set of goals be observed; *or* requires that some element of a set of goals be observed. Implicit in the temporal structure description is the logical relationship *and*, as will be discussed below. 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 Chapter 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, because all decisions can be critically dependent upon the output of a previous decision. The framework proposed here cannot model that type of plan.

6.3.4 Example

Figure 6-2 shows the temporal description for the s51 example play, diagrammed in Figure 6-3. 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. This slot is necessary because the same play definition can apply to multiple starting position configurations. In the s51 definition, for example, the first object (obj1) always maps to the trajectory labeled as the C object. The third object (obj3), however, can map to more than one starting trajectory. Obj is most likely to map to the RWB trajectory if a trajectory in the dataset has that label. If no such trajectory exists, the obj3 definition could map to a trajectory labeled RTE. The next best match – if there is no RWB and RTE trajectory – is RHB.

Following the agent slot, one or more goal slots are specified. The example indicates

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

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

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

  ;The Right Wing Back (RWB)
  (agentGoal obj3
     (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 offSidelineRightReg
                                        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 offSidelineLeftReg
                                        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))
```

Figure 6-2: A temporal structure description for the *s51* play example with only some actions and temporal relationships specified.

that in an s51 play, obj1 (which will match the trajectory labeled C), should have a goal to 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, blockQBPass, passPatStreaking, passPatCutting, passPatParaLos, catchPass. For each goal type, there

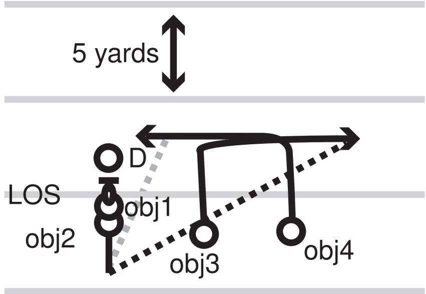


Figure 6-3: The s51 example from Chapter 2.

exists a detector which receives a list of parameters. The detectors are implemented as Bayesian networks that integrate direct visual measurements. Some of these detectors and their parameters are described below:

blockQBPass (obj) *Obj* blocks for the QB as the QB tries to pass the ball.

passPatStreaking (obj distance angle inReg toReg distLOS) An eligible receiver, *Obj*, runs a pass pattern segment that is approximately straight for at least *distance* yards at an angle of approximately *angle* with respect to the line of scrimmage region (losReg). For example, if *angle* = 90, the obj will run perpendicular to the LOS for *distance* yards. This pattern occurs while *obj* is located in region *inReg* and when the *obj* is headed towards region *toReg*. Finally, *distLOS* specifies the approximate distance from the losReg that the motion starts. For example, "passPatStreaking (obj4 5 90 defReg offEndZoneReg 0)" indicates that obj4 will run perpendicular to the LOS for 5 yards in the defReg (defense's end of the field) moving towards the offEndZoneReg (the offense's end zone) starting 0 yards from the LOS.

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*.

passPatParaLos (obj distance inReg toReg distLOSatStart) An eligible receiver, *Obj*, runs a pass pattern segment moving parallel to the losReg for at least *distance* yards while in *inReg* and heading towards *toReg*. This motion occurs at approximately *distLOSatStart* yards from the LOS.

catchPass(obj) *Obj* intends to catch the ball from a pass.

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

```
(agentGoal obj3
   (agent (obj3 (RTE RHB HB FB TB LWB LSB)))
   (cgoal obj3_act1
      (goal obj3_act2 "passPatStreaking (obj3 4 45 defReg
                                        nearRightSidelineReg 0)")
      (goal obj3_act3 "passPatCutting (obj3 70 offSidelineRightReg
                                              freeBlockingZoneReg)")
      (before obj3_act2 obj3_act3))
   (cgoal obj3_act4
      (goal obj3_act5 "blockForQB (obj3)")
      (goal obj3_act6 "blockForBC (obj3)")
      (before obj3_act5 obj3_act6))
   (xor obj3_act1 obj3_act4))
```

Figure 6-4: A temporal structure description for the *s51* play example with only some actions and temporal relationships specified.

label for "blockQBPass (obj1)". Similarly, "(around obj3_act2 obj4_act2)" indicates that either object3's passPatCutting goal occurs around the same time as object4's passPatCutting goal. The meanings of "before" and "around" will be defined shortly. Finally, "(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* is simply a goal comprised of multiple primitive goals. Temporal and logical relations can exist between the compound goal and other primitive goals, as well as between the primitive goals of the compound goal itself.

Since compound goals 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 compound goals represent causally linked individual goals that are tightly coupled. Therefore, it is possible to specify temporal and logical relationships between compound goals. For example, consider the temporal structure description fragment in Figure 6-4. Two compound goals (cgoals) are identified: obj3_act1 and obj3_act4. Within each compound goal, subgoals are listed with appropriate temporal relationships specified (e.g. obj3_act2 before obj3_act3). However, a logical constraint is also specified between the two compound goals, "xor obj3_act1 obj3_act4," indicating that either all the sub-goals of obj3_act1 are observed or all the subgoals of obj3_act4 are observed, but not both.

6.4 Perceptual features

The temporal structure description uses agent goals like *blockQBPass* and *catchPass* to describe a multi-agent action. The input to the system, however, 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 The most basic detectors are for movement and angular movement properties of single agents (e.g. velocity, angular acceleration). Many of these detectors also operate on groups of agents (e.g. *velocity(offensePlayers)*).

Trajectory curvature These detectors analyze a single trajectory curvature and allow curvature inspection over particular temporal windows of the trajectory (e.g. *curvatureTraj(obj)*) They also measure properties with respect to points of locally maximal curvature along a single agent's trajectory (e.g. how long ago was a point with high curvature?).

Categorization and visibility All objects have types and supertypes, and these detectors can confirm an object is of a particular type and also check if particular objects are visible in the scene (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 These detectors check for distance relationships between objects and spatial regions. Some use a deictic perspective. Detectors can also use individual objects or collections of objects (e.g. closestAgent(obj,defenseObjs)).

Relative orientation of objects These detectors check if objects and groups of objects are oriented in particular ways with respect to other objects, groups of objects, or regions (e.g. *facingOneOf(obj1,defense)*).

Measurement of change These detectors detect some type of spatial change (or orientation change) greater than some specified value between the current time and some specified time change. Some of these detectors detect change with respect to some region (e.g *compareDistanceMoved (obj timePT distance)* checks if obj has moved distance since timePT).

Trajectory properties These detectors operate over the entire trajectory for some object observed up to frame currentTime and measure properties such as if the trajectory passes between particular spatial regions (e.g. *enteredReg(obj decTime region)* checks if obj entered region within decTime).

Spatial relationships between objects The spatial relationship detectors can compare individual and collections of objects (e.g. *behind(obj,linemen)*).

Spatial relationships during intervals of time These detectors examine spatial relationships between objects or objects and regions during intervals of time (e.g. *closestApproach (obj, QB, decTime)* returns the closest distance obj came to QB in the interval from currentTime to decTime).

Path interception estimation Some detectors use the current time and velocity and acceleration estimates to predict when two objects, an object and a region, or the orientation of two objects, will intercept in the future (e.g. *intercept (obj1 obj2 incTime)* determines if obj1 will intercept obj2 before incTime assuming current trajectories stay constant).

Temporal extent checks These detectors compare some given domain-dependent time point to the current time (e.g. *timeElapsed((throwTime))* returns the time elapsed from throwTime if that time has been computed, otherwise nil).

Figure 6-5 and Figure 6-6 list a representative set of feature detectors. Each detector takes a set of arguments, which are usually either objects, groups of objects, 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 [Zad78] using overlapping piecewise linear functions.[4] Figure 6-7 shows the piecewise discretization functions for the *velocity* and *facing* evidence functions. In this work, these functions are manually specified by the domain expert.[5] 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. The interpretation of the membership curve values as relative likelihoods and how they are used by the visual networks are described in Section 6.5.

Figure 6-8 shows a screenshot of an interface used to view the output of feature detectors for a particular play example. Only a small subset of the features computed are shown. The

[4]In this work, 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. See Section 7.5.

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

accelerationMaxVect (objs) Value of accelerationX (objs) and accelerationY (objs) that is farthest from zero. States: accelFastPos accelPos none accelNeg accelFastNeg.

angularChange (objs) Abs(angularAcceleration (objs)). States: none change fastChange.

backupReg (obj region) Determine if obj velocity is in opposite direction of shortest path from obj to region. States: true false.

closestDistance (objs1 objs2) Distance from objs1 to closest agent in objs2. States: lessp5 less1p5 less2p5 less5p5 less9p5 less14p5 more14p5.

distanceReg (objs region) Distance from centroid of objs to region. If there are multiple regions, the distance to the closest region is used. States: lessp5 less1p5 less2p5 less5p5 less9p5 less14p5 more14p5.

agentsWithinDistBetweenObjsReg (obj1s objs2 region) Number of agents in objs2 within d radius of any objects in objs1 where d is the closest distance between any agents in objs1 and region. States: n0 n1 n2 n3 n4 more4.

facingDist (objs1 objs2) Centroid angle and position of agents in objs1 facing towards centroid position of objs2 taking into account the distance between objs1 and objs2. States: facing oriented notFacing.

facingOneOf (objs1 objs2) Centroid angle and position of agents in objs1 facing towards at least one of objs2. States: facing oriented notFacing.

facingSameDir (objs1 objs2) Centroid angle of objs1 facing in the same direction as centroid angle of objs2. States: facing oriented notFacing.

facingAngToReg (objs region angle) Centroid of objs is facing perpendicular (when angle 90) or parallel (when angle 0) to the line segment of angle closest to objs. States: aligned partlyAligned notAligned.

Figure 6-5: Some perceptual feature detectors.

shaded bars indicate the output of the feature detectors over time. Not shown is the degree of membership value associated with each output state at each time. Which perceptual features are computed is determined by the goals listed in the temporal structure description. The next section explains how these goals are modeled.

6.5 Visual networks

Previous work, discussed in Chapter 5, demonstrates that agent goals can be represented in a probabilistic framework using Bayesian belief networks [CG93, HDW94, PW95]. These networks are applied instantaneously to each image frame. The Bayesian networks use

curvatureTrajAbs (obj) Abs(curvatureTraj (obj)). States: straight curved curled.

enteredReg (obj decTime region) Check if object entered region between currentTime and (currentTime - decTime). States: true false.

leftOfReg (objs1 objs2 region) Objs1 is leftOf objs2 with respect to the direction of the vector from objs2 to the closest point on region. States: true false.

between (objs1 objs2 objs3) Objs1 is between objs2 and objs3. States: true false.

moveAngToReg (obj1 region decTime angle) Check if obj moving at approximately angle orientation to region (where 0 is parallel and 90 is perpendicular) from currentTime to (currentTime - decTime). States: true false.

compareDistanceMoved (objs timePT distance) Difference between given distance and the distance the objs centroid has moved from the timePT to currentTime. States: less2 less4 less6 less8 less10 less12 less16 less20 less24 more24.

interceptSpace (obj1 objs2 objs3 incTime) Obj1 is moving towards crossing the dividing line between objs2 and objs3 within incTime timesteps. Computed using obj1's velocity. Obj1 must go between every pair from objs2 and objs3. States: observed maybeObserved notObserved.

possibleToIntercept (obj1 obj2 timesteps) Determine if obj1, moving in the most direct path at "running" speed can intercept obj2 before (currentTime + timesteps) given obj2 maintains its current velocity. States: observed maybeObserved notObserved.

Figure 6-6: Some more perceptual feature detectors.

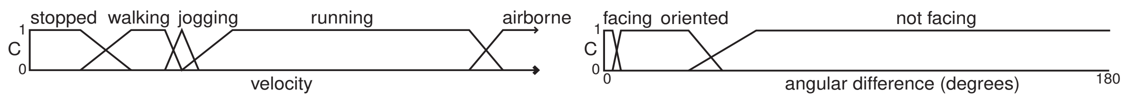


Figure 6-7: Two examples of the piecewise-linear functions used to compute a degree of membership value by perceptual detector functions computing object motion and object-to-object facing angle.

the output of the visual feature detectors, so they will be referred to as *visual networks*. Belief networks, which are reviewed in Appendix E, can be designed so that they can represent uncertainty in evidence, goal models, spatial reasoning, and temporal reasoning. The networks are used as building blocks for recognizing multi-agent activity by using their outputs to check for temporal consistencies such as the visual evidence for goal A is *before* the visual evidence for goal B.

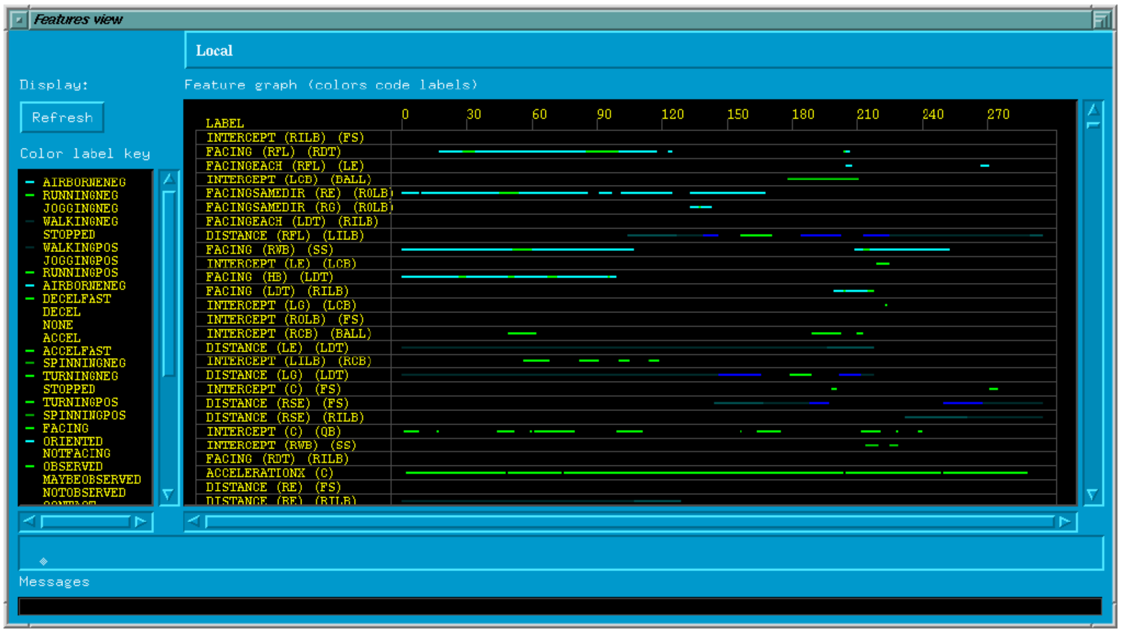


Figure 6-8: Screenshot of an interface used to display perceptual feature output. Only a small percentage of features is shown.

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 is 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 [Jen96] 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. Some issues related to learning networks will be discussed in Section 7.5.4.

Figure 6-9 shows two networks, *catchPass* and *playInProgress*. The networks consist 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

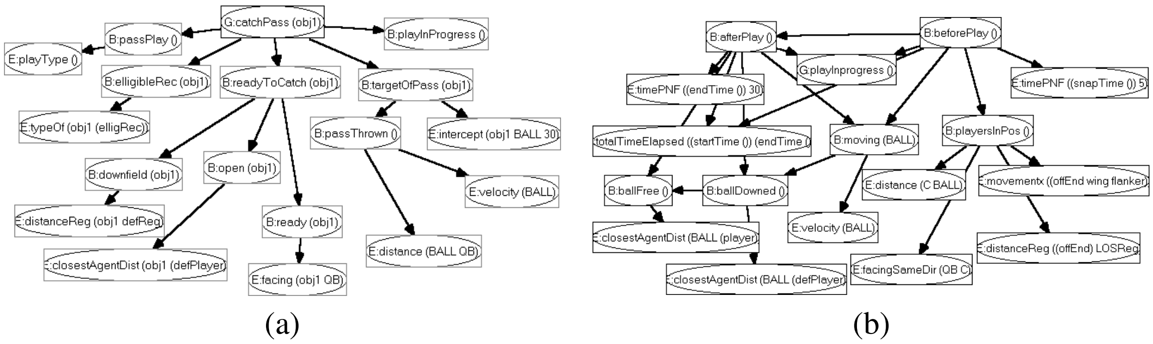


Figure 6-9: Two visual networks: (a) the *catchPass* goal network (left) and (b) the *playInProgress* belief network.

corresponding probability value), and represents an internal state of the agent or some external state in the world at the time when the network is evaluated. Belief nodes can also represent the belief that an agent has a particular goal; these nodes are not attached directly to evidence nodes. This distinction forces the network designer to explicitly represent two types of uncertainty: (1) the uncertainty of the detectors given noisy data, modeled in the conditional probabilities $P(evidence|belief)$, and (2) the uncertainty resulting from the use of an approximate model, which is modeled in the conditional probabilities $P(belief|beliefInGoal)$. Each visual network has a designated *main belief node* (e.g. *catchPass* and *playInProgress*).

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. Appendix F briefly describes the visual networks used by the system.

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 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 6-9a), 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 and makes it manageable.

The same virtual evidence computation described in the previous section can also be used to use continuous-valued output from visual networks as evidence in visual networks. The evidence from the output of the *playInProgress* visual network is entered into the *catchPass* network in Figure 6-9a using this method. For example, evidence from an external network, such as the *playInProgress* evidence node in Figure 6-9a, is incorporated into a network such as *catchPass (obj1)* as follows. If the *playInProgress* network cannot evaluate and returns NIL, no evidence is entered for the node. If the *playInProgress* network returns a high certainty of a particular state that exceeds a predetermined threshold for *playInProgress*, evidence is entered directly into the *catchPass* network (e.g. if *observed* = .99 and *notObserved* = .01 and *threshold(playInProgress)* = .85 then *observed* = 1.0 is entered into *catchPass*). Finally, if *playInProgress* evaluates below the threshold, the beliefs are treated as direct evidence and the probabilities are converted to likelihood evidence[Pea88] (e.g. if *observed* = .8 and *notObserved* = .2 and *threshold(playInProgress)* = .85 then the evidence that *observed* is 4 times more likely than *notObserved* will be entered into the *catchPass* network).

In practice, nested visual networks were required to manage the complexity of knowledge engineering the visual networks for the play recognition system. Appendix F breaks the implemented visual networks into "generic action detectors" that detect basic motions or agent relationships in a small space-time windows (e.g. movingBackwards(obj), stayingBetween(obj1 obj2 obj3), runningStraight (obj distance), runningToward (obj1 obj2)) and football-specific detectors (e.g. catchPass (obj), dropback (obj)). Implementing the football-specific detectors without use of smaller, generic network components that could be inserted as evidence proved impractical. One change to a generic action detector would require changes in nearly all of the football-specific networks. Consequently, the football-specific detectors were constructed to use generic network outputs as evidence. 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; these assumptions are discussed further in Chapter 7.

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.

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. To avoid the computational cost of evaluating all the networks and then iterating to convergence, in this system, the knowledge engineer is encouraged to minimize the number of interconnected networks. Loops that remain are handled by having the evidence computation algorithm avoid recomputing any networks that have already been computed during the current iteration.

6.5.3 Evidence quantization

Many of the evidence detectors measure naturally continuous quantities (e.g. velocity); however, although it is possible to design Bayesian networks with continuous variables [Ole93], developing such networks requires the knowledge engineer to provide multi-variable continuous distribution functions. Practical networks typically use discretely-valued nodes because of the modeling difficulty of specifying continuous conditional probability functions that model real world concepts with non-smooth transitions. Quantized values can be entered into a network as continuous evidence, keeping the knowledge engineers task manageable without ignoring continuous evidence. To maintain continuous valued information with discrete states, whenever possible evidence is entered as "virtual" likelihood evidence (see [Pea88]). The likelihood is obtained using the degree of membership values obtained for each of the node's states (see Section 6.4 and Figure 6-7). Virtual evidence is discussed in the next section.

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 6-10. 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 the heuristic functions in Appendix G that compare the activation levels of each goal over time, 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. Figure 6-10 shows the certainty values for the *before* 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 functions in Appendix G.

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

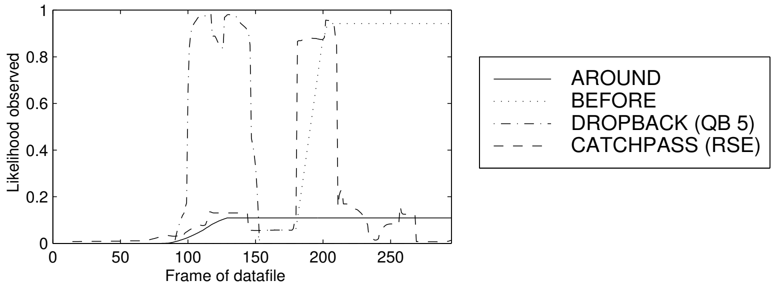


Figure 6-10: Goal likelihood curves returned by the networks "dropback (QB 5)" and "catch-Pass (RSE)" superimposed with the corresponding temporal curves for "dropback (QB 5) before catchPass (RSE)" and "dropback (QB 5) around catchPass (RSE)".

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 6-11 shows an example of a multi-agent network for recognizing *part* of the action description shown in Figure 6-2 for the s51 play example. The network is similar to the visual networks described previously, but it has some special structure, described below, to facilitate recognition of group action by multiple agents.

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 observed 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

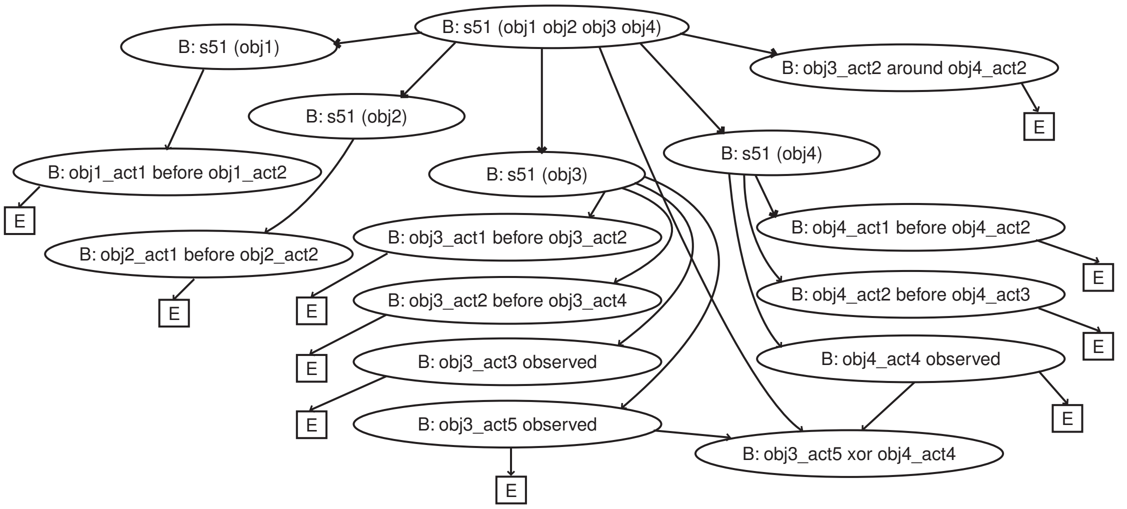


Figure 6-11: The multi-agent network structure for the s51 play temporal structure description in Figure 6-2.

Figure 6-2), 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).

The network shown in Figure 6-12b is an example network for the play description in Figure 6-12a. This example demonstrates some additional multi-agent network properties. If a goal does not have a 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.

```
(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))
```

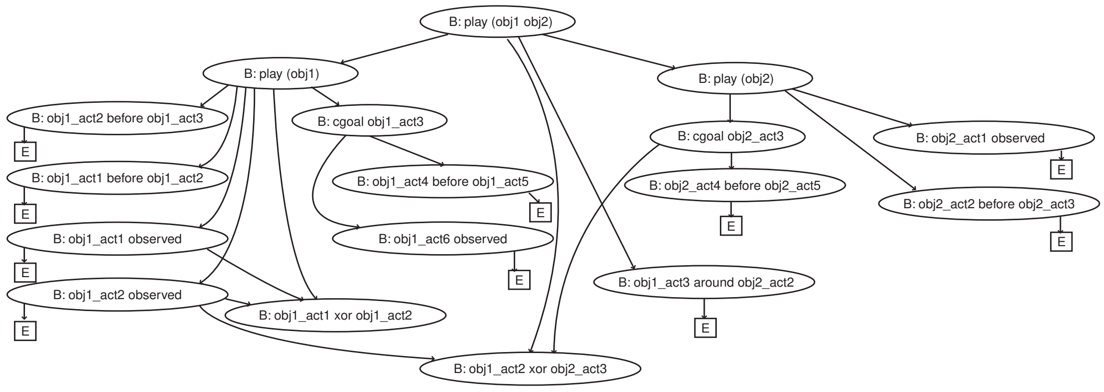


Figure 6-12: A temporal structure description and the multi-agent network structure generated.

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 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 Chapter 7.

The network shown in Figure 6-11 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. Performance will be discussed in Chapter 7.

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 multi-agent network structure is generated using a set of rules specifying how to insert and connect nodes for a temporal structure description so that the node-link structure described in the previous section are obtained.

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 [Hub96], although with networks of a different structure. The conditional probabilities themselves are estimated by the knowledge engineer.

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).

Huber [Hub96] and others have used the idea of automatic construction of networks using rule-based expert systems [Hol88, NHH95], first-order logic [GC90], Horn clauses with associated conditional probability tables [Bre92], and construction of networks based on sensitivity analysis pruning [Bre92, PC93, HHNK95]. All these methods rely on prespecified conditional probability tables and (to make the techniques practical) a relatively small set of primitives that can be 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 6-2, the *agent* slot of the *agentGoal obj3* description indicates that object *obj3* can possibly match with a trajectory if the trajectory has one of the labels *(RWB RTE RHB HB FB TB LWB LSB)*. This list is a preference ordering. *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

One multi-agent network exists for each play in the database, and the networks are evaluated simultaneously as data is observed. Figure 6-13 shows an example of output curves for the football play detection system, showing the likelihood values for each play over time.[6] As time progresses and the play begins, the likelihood of the correct play rises above the other play types. Currently the system is computing the certainty 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.

The next chapter describes the recognition performance of the system and discusses the

[6] The algorithm's recognition performance is discussed in the next chapter.

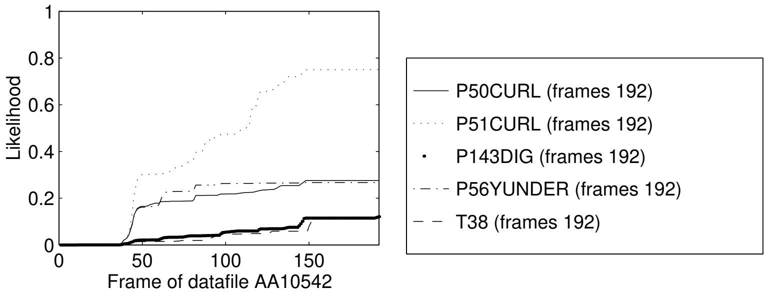


Figure 6-13: Some likelihood curve outputs for the p51curl play.

strengths and weaknesses of the representation proposed in this chapter.

Chapter 7

Recognition results and evaluation

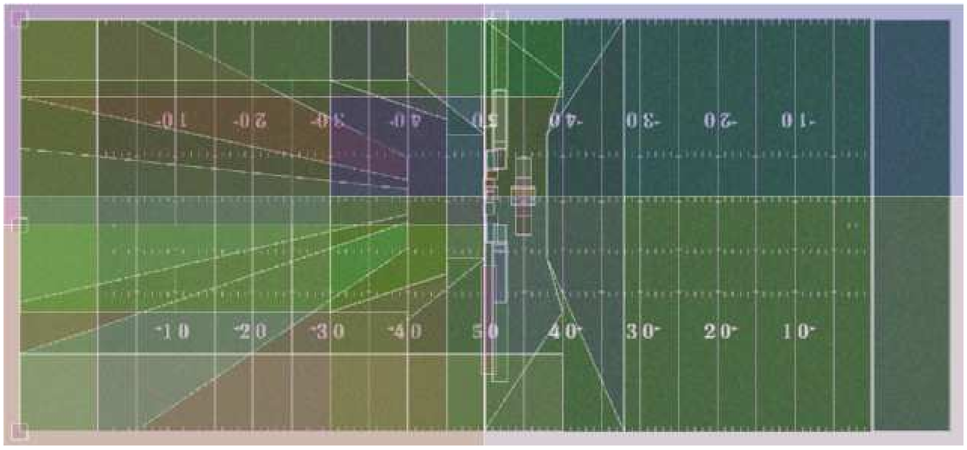


Figure 7-1: Football specific field regions. Some regions are defined based upon the starting position of the ball in each play.

This chapter describes the results of applying the representation for multi-agent action recognition developed in Chapter 6 to a database of play examples. The performance of the algorithm is discussed. The algorithm is then analyzed in terms of the desirable multi-agent representation criteria raised in Chapter 2 and Chapter 4. Finally, the chapter concludes with a brief discussion on the scope of the proposed recognition algorithm by considering how it might be applied to the two non-football domains introduced in Chapter 2.

7.1 System components and knowledge engineering

The representation for action recognition developed in Chapter 6 is 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 Chapter 6. Football-specific regions of the field were specified and are shown in Figure 7-1. 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; these were described in Chapter 6 and are listed in Appendix F. The knowledge engineering process for visual networks is a time-consuming one. In general, Bayesian networks, despite their desirable properties as

Figure 7-2: Interface showing output from visual goal networks used for testing.

a representation for uncertainty, are tedious and time-consuming 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 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.[1] Therefore, networks were tested by running each network for each player in a few example plays and using the interface shown in Figure 7-2 to visually check that the detector generated reasonable outputs when applied to any player object. This testing was also used to set a threshold value for each network. The threshold indicates a value above which the network's goal is considered to be definitely detected.[2] 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, about half of the networks are specific to the football domain.

Other knowledge utilized by the system includes the conditional probability values used during multi-agent network construction and the rules used to generate the multi-agent network structure from the temporal structure description. The estimates in the conditional probability tables were designed to softly weight evidence and are discussed further in Section 7.4. The structure of the networks was described in Section 6.7.

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.

[1] See Section 8.2 for information about an online system that has been developed to obtain such data.
[2] This value is used to set the few global variables used by the system, such as *ballThrown*. See Section 6.5.2.

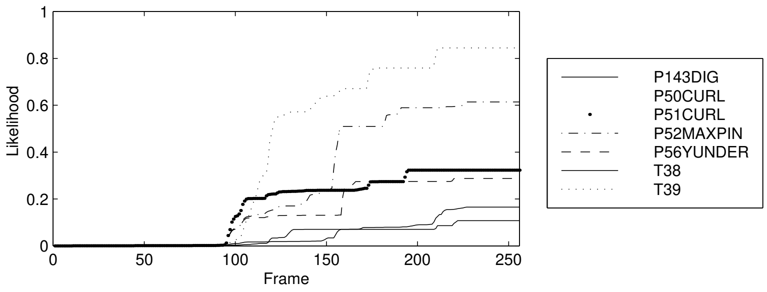


Figure 7-3: Result of running 7 play detectors on a t39 play example. Shown is the likelihood of each play having been observed at frame t considering all evidence from frames $0 - t$.

7.2 Recognition performance

Figure 7-3 shows the likelihood value obtained by evaluating the multi-agent network at each frame for 7 play models on an example of a t39 play. Here, the desired behavior is achieved because uncertain evidence of temporal relationships between goals is sufficient to cause the t39 play detector's likelihood value to quickly rise above the other plays shortly after the play action begins at frame 90.

Chapter 3 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. Chalkboard diagrams of each play type appear in Appendix C.

Figure 7-4 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). 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 probabilities 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 will only be addressed by strengthening the models of play action by including more action components and temporal relations per agent. The weaker the model, the more easily it is matched by some incorrect instance. More detailed

Name	p143dig	p50curl	p51curl	p52maxpin	p54maxcross	p56yunder	p63up	p63upa	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	-	-	-

Figure 7-4: Likelihood values when the recognition system is run for each of 10 play models over a data set of 29 examples.

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 7-4 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 undisputed measure of similarity to use when comparing the differences between two plays. The examples in the table must be examined individually to understand why the algorithm performs as it does.

The "file 20" example in Figure 7-4 (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 p52maxin. Figure 7-5 has a p52maxpin and p56yunder play diagrams approximately 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. Both plays, when executed perfectly, are similar when the "optional action" of one player is not taken into account. 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

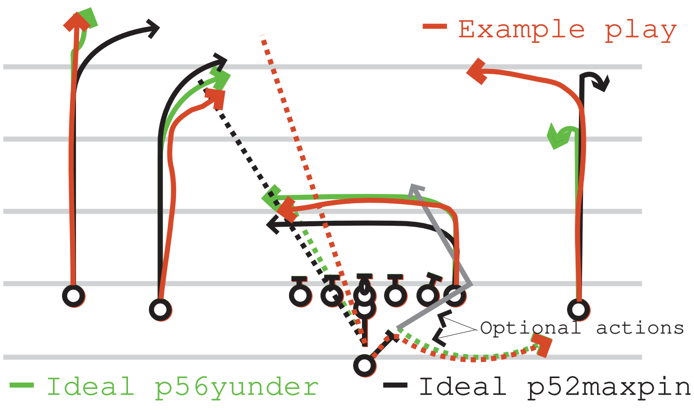


Figure 7-5: P56yunder and p52maxpin play diagrams with one p56under example play approximately overlaid. The system returned likelihoods of .64 for p56yunder and .76 for p52maxin.

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.

The performance of the algorithm is promising, but some caveats are necessary. Given the limited available data set, the same data was used for training and testing. Training consisted primarily of reverse engineering the play descriptions, using a much larger set of plays on video prior to developing the recognition algorithm. However, visual goal networks and ultimately temporal structure descriptions were developed using the data that has been used for testing. If a much larger data set were obtainable, a useful experiment would be to apply the existing multi-agent play networks to that data.

There are a number of problems with the representation, which are discussed in Section 7.5. Despite these issues, however, the algorithm is performing well on a real-world, multi-agent recognition task. The next section considers why.

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 7-6 (repeated here from Chapter 6 for convenience). These nodes are structured as a naive Bayesian network [DH73]. 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

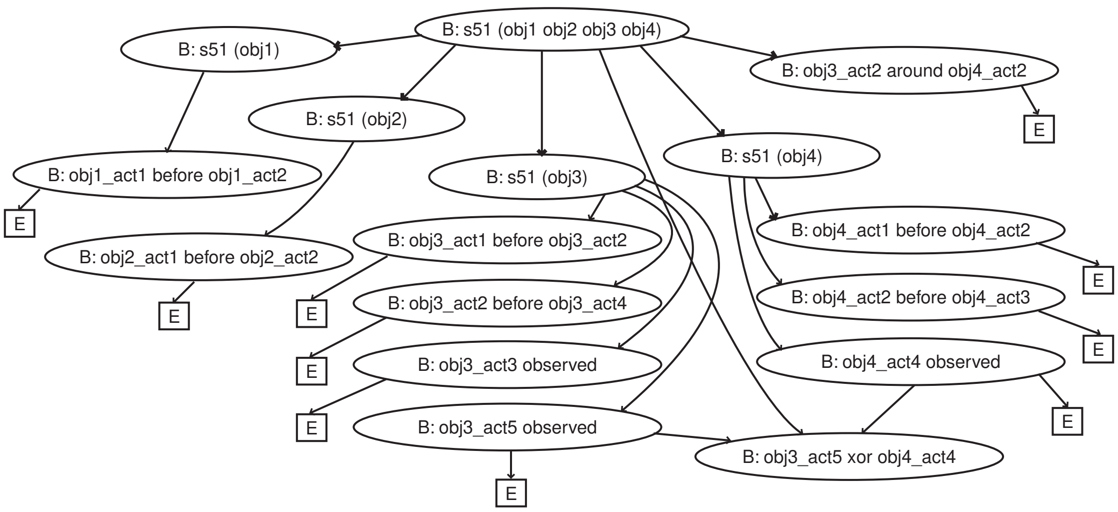


Figure 7-6: The multi-agent network structure for the s51 play temporal structure description in Figure 6-2.

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), where the class is the value of the compound action and the attributes are evidence of the goals of that particular agent. The network, however, is not quite a hierarchical naive Bayesian network. For example, binary connections between some nodes model logical relationships. Further, the nodes themselves are not simply object attributes. Most nodes represent a *relationship* between two agent goals. These nodes capture attribute dependencies.

7.4.1 Capturing important attribute dependencies

Naive Bayesian classifiers make strong (and often clearly incorrect) independence assumptions that each class attribute is independent given the class. They also assume that all attributes that influence a classification decision are observable and represented. For these reasons, they are sometimes assumed to perform poorly in real domains. On the contrary, experimental testing has demonstrated that naive Bayesian networks are surprisingly good classifiers, despite their strict independence assumptions between attributes and the class. In fact, simple naive networks have proven comparable to much more complex algorithms, such as the C4 decision tree algorithm [LIT92, CN89, JL95, DP96]. One theory is that the low variance of the classifier can offset the effect of the high bias that results from the strong independence assumptions [Fri97]. The probabilities encoded in the classifiers are typically learned from large data sets.

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* [FGG97]. 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 [FGG97, KP99].

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. Generally, both network structures assume significant independence but they attempt to model "important" dependencies. For instance, as discussed in Chapter 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 independence but specifying some important low-order dependencies is a promising approach for modeling real-world domains.

7.4.2 Network computability

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. Evidence can be propagated in time linear to the number of nodes [Pea88] because the size of the largest clique is just 2 regardless of the number of nodes (see Appendix F for a discussion of Bayesian networks, clique size, and propagation complexity). 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 7-6 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.

The logical relations break the singly-linked tree structure. Figure 7-7 illustrates the problem when two branches of the tree are linked by a logical relation. The logical relationship bridges the two paths and forces connections along the path to be triangulated in such a way that rapidly increases the clique size. Tree paths linked by logical relations lead to a minimum maximal clique size of 4, as illustrated by the example in Figure 7-7 that is 5 nodes deep. This structure is computationally manageable because the complexity is $O(2^N)$ (where the 2 results from each node in the multi-agent network being binary and N is the maximal clique size of 4). Of course, the complexity can increase further depending upon whether either side of the tree path is connected to another logical relation – these new links can sometimes increase the maximal clique size. In practice, however, the multi-agent

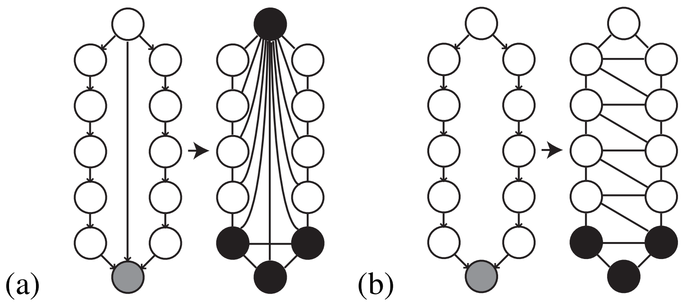


Figure 7-7: Original and triangulated graphs for tree chains linked by a logical relationship (gray nodes). (a) A logical relationship linked directly to the class has maximum clique size of 4. Nodes for one of the cliques is filled in. (b) Not connecting the logical relation directly to the class reduces the maximum clique size to 3.

networks are typically only 3 levels deep and have just a few logical relations that span across agents.

As illustrated by Figure 7-7b, logical relationships can be enforced with a smaller clique size if the logical constraint node does not link back to the main node, as they do in the multi-agent networks. The network, however, then cannot explicitly weight the importance of the logical relationship in determining the class. In the multi-agent networks, the logical relations can be considered both as a constraint influencing other attributes and as an attribute as well. In practice, the multi-agent networks generated from the temporal structure description have maximal cliques sizes of 4.

Intuitively, logical relationships are computationally expensive because Bayesian networks are not performing a directed, explicit search of a space using an efficient search heuristic. Instead, the entire space is "searched" when evidence is propagated according to Bayesian rules. This is a desirable property for combining uncertain evidence from multiple sources because each bit of evidence can appropriately influence the probability of any state in the graph (depending upon the structure of the graph of course). However, the more that states depend upon particular key decisions (i.e. global variables), the larger the search space becomes. Consequently, the entire space is considered each time new evidence is propagated through the network. Networks where evidence is only provided at leaf nodes have no mechanism by which to effectively prune large regions of the search space.

Network structures previously used in object recognition research were discussed in Chapter 5. In this thesis, a network structure has been selected that captures some important logical relations (deemed "important" by the domain expert) but that primarily softly integrates uncertain information. The network is a top-down, part-of network, where logical relations link back to class nodes. Object characteristics (or in this case, goal characteristics) are not stored in node states, so there are no comparisons on links. Node state size and link complexity are therefore kept low. Links operate as *and* relations for class attributes. Unary and binary temporal comparisons are modeled *within* nodes. The graph stores primarily relational information about an action over time instead of absolute information. For example, goal A before goal B is stored instead of more direct information, like goal A is active 2 seconds starting at 1 second into the play and goal B is active 1.5 seconds starting at 5 seconds into the play. The latter approach requires the network to represent global variable information internally, increasing network complexity. The multi-agent networks, however, do rely on some information being stored outside of the network (e.g. *ballThrown*). Multiple networks have been used to make the knowledge engineering task manageable. Finally, the network structure is convenient because it requires the specification of $P(evidence|state)$ instead of $P(state|evidence)$. DBNs have not been used because the knowledge engineer in this domain is specifying change that can be measured by evidence detectors – the temporal analysis functions. DBNs, however, usually represent changes in temporal state in transitional probabilities on links that are difficult to encode. Further, it is unclear how to encode logical relationships between temporal events in a DBN framework.

7.5 Representational strengths and weaknesses

This section discusses some strengths and weaknesses of the proposed algorithm with respect to issues raised in Chapter 2, Chapter 4, and Chapter 5.

The main strength is that the algorithm is performing well, recognizing a collaborative, multi-agent action in a real-world domain with noisy input. The algorithm demonstrates that using probabilistically weighted, low-order temporal relationships to recognize *coordinated* visual components of collaborative activity permits the recognition of highly structured, *collaborative* multi-agent action. The idea that the networks capture the most important temporal dependencies, which are relationships between individual agent goals, should extend to some other domains of interest. Work in learning using naive Bayesian networks and TAN networks also suggests that this is an approach worth pursuing further, although perhaps with components that are partially learned from data sets to minimize the role of the knowledge engineer.

The algorithm is now discussed with respect to issues raised in previous chapters.

7.5.1 Comparison to object recognition

In Chapter 4, some properties of the object recognition task were identified. This section reviews the multi-agent recognition system with respect to those criteria.

Object recognition models, features, and correspondence In this work the models used for recognition are not geometric; they are temporally ordered agent goals. The goals are detected using the visual networks. The features are the goal likelihood curves, which are computed using networks that integrate evidence computed from trajectory data. The algorithm does not explicitly check for correspondence. Instead, it finds a *consistent* hypothesis, given low-order temporal relationships between goals.

Therefore, the algorithm is invariant to local spatial and temporal distortions but not temporal *reordering*, which (as discussed later) requires a representation that can model a re-plan.

Occlusion. The equivalent of occlusion in the static object recognition domain is unmodeled interaction between agents. These unmodeled events will disrupt the action of other agents in ways that are explainable only if the action of the deviant agent is also explainable (e.g. the analogous situation in static object recognition is that a system cannot reason about an occlusion situation if it does not have a model of the *occluding* object that is sufficiently strong to explain the observed deviation in the occluded object). The algorithm presented here is robust to small "occlusions" that cause local changes in the trajectories single agents, but the algorithm has no mechanism to model agent occlusion that causes a temporal re-ordering of agent goals – a transformation not modeled by the representation.

Noise. The data used by this system are noisy. The algorithm's probabilistic integration of uncertain information can handle noise in individual agent trajectories and unpredictable but minor behavior modifications of individual agents (e.g. in response to a local defender). Explainable "noise" in agent behavior, caused by an action such as mistake by a player (e.g. falling), is not handled by this representation any differently than random perturbations.

Natural labels. Objects can be labeled at multiple scales and specificities. In this work the hypothesis is made that the action components of high-level, collaborative action are the goals of individual agents. These action components are temporally and logically linked to model multi-agent actions. All representations for multi-agent action will need to make a commitment to some level of description of action. Even representations that are completely learned from data and do not require a knowledge engineer will need a training database of action with semantically meaningful labels. This work advocates labeling individual agent goals. Designers of Bayesian networks know the importance of using meaningful state labels. One of the most important steps when designing a network is to assign the nodes in the network with semantically unambiguous names, so that the modeler is accurate in the way that conditional links are assigned. The same will be true for systems that learn action representations from data.

Hypothesis vs. testing. In geometric object recognition, the task is to find a hypothesis for a match and then perform a full transformation and match to confirm that the hypothesis matches the data. Grimson and Lozano-Pérez [GLP87] use low-order relationships to find good hypotheses. Analogously, here low-order temporal relationships between goals are used to find hypotheses of collaborative action by identifying coordination. Definitively confirming that the hypothesis is "globally consistent," however, may require intentional reasoning about agent interaction. Intentional inference is required to confirm that the observed coordination definitely resulted from team collaboration and not from random agent interactions.

Indexing. This work does not solve the indexing problem. All existing play models are tested against each play example. As shown in Figure 7-3, however, once play action begins the likelihoods of some plays rise substantially above the remainder. The likelihood values could be used to prune the recognition process and reduce the amount of computation required for a large model set.[3]

Feature clustering and focusing. Making object recognition practical requires heuristics for clustering features and finding "focus features" [Gri90]. In this work, no features are explicitly specified as more important than others. An additional node type could be added to the multi-agent networks, however, to bias the multi-agent network result by differentially weighting features. Implicit differential weighting in the multi-agent networks is discussed below.

Match score and good-enough termination. A match score that permits early search termination is also necessary to make geometric object recognition based on searching interpretation trees practical. The multi-agent networks are computing a likelihood value that could be used to prune hypotheses and terminate search for some models before the entire play has been observed. The likelihood value results from the integration of *all* available evidence observed from the start of the action to the current time.

7.5.2 Properties of multi-agent action

In Chapter 2, some properties of multi-agent action were identified. These properties are revisited with respect to the proposed multi-agent action representation.

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. The 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.

[3]Other methods to improve performance could also be explored. For example, some similarity measure between action descriptions could be used to reduce the number of models tested at the beginning of the play. As time passes, models that have performed well may activate other similar models to be tested.

Feature space size. The size of the feature space is large because features of importance are features that *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.

Feature complexity. Feature complexity refers to features that are context-dependent, such as many spatial relationships. In this work, detectors, like *behind*, are approximated as functions that return a continuous certainty value that is then inserted as continuous evidence into a visual goal network. Although the detectors do not incorporate as much contextual information as they could, hard thresholds are not required when the evidence is entered into the visual goal networks, and the networks propagate the uncertain value to related variables.

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.

Explicit search and object assignment. As discussed in Section 7.4.2, networks poorly represent search spaces with many global state variables. Since every model-object to trajectory-object assignment depends upon the assignments made for other players, designing networks that perform action recognition *and* object assignment is extremely difficult, resulting in unmanageably large networks that have impractical clique sizes due to all the cross dependencies in agent labels. Explicit search with heuristics is a more computationally efficient representation. The object assignment step used by the current representation (described in Section 6.8.1) performs an explicit search to match trajectory data to the object description. This dichotomy in the representation – part Bayesian network, part explicit search – may represent a more general problem. Bayesian networks are excellent at integration of information, but they do not replace explicit search. Explicit search with effective search heuristics, however, is probably necessary to develop a representation that goes beyond recognizing "what things look like" and recognizes intentional interaction – "what things are." Action representations that encode intentional concepts will probably require more dependency on explicit search, but the Bayesian network framework proposed here may make the explicit search possible by providing robust, probabilistic inputs.

Relative geometric spatial and temporal knowledge. The most useful features used in both the visual goal networks and the multi-agent networks are features comparing two agent trajectories or two agent goals.

Scalability (in agents). The proposed algorithm was designed for scenes with 11 agents. The idea of using many low-order temporal relationships between agents may scale up to multi-agent actions that involve hundreds of agents (e.g. see [DR98] for a simple example). In such cases, it may be possible to monitor aggregate behavior of the many agents to simplify the problem or to use distinctive clusters of activity to find specific agents in the scenes upon which to focus computation. Scaling the algorithm down to fewer agents results in a greater dependence upon the visual goal networks (see the cooking oatmeal example later in this chapter). In the 1-2 agent case, the representation probably offers little benefit over other existing techniques, like HMM models, because the HMM techniques can be learned from examples if appropriate data sets are available.

Scalability (in classes). The algorithm must also 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.

Intentionality and the difficulty of physical modeling. Examples were discussed in Chapter 2 and Appendix B that illustrate how the actions of players on the team will never be possible to predict using physical models alone.

Availability of evidence. One major advantage of using Bayesian networks is that any evidence, observed at any time, can be incorporated into a network. Evidence that is missing or not computable is considered via the network priors.

Contextual label interdependencies. The multi-agent representation does not provide a mechanism for context-based labeling (discussed in Chapter 4), which requires an explicit search.

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.

7.5.3 Desirable recognition properties

In Chapter 4, desirable properties for a recognition system were discussed.

Probabilistic ranking. The formation labeling system described in Chapter 4 lacked probabilistic ranking rules. The formation labeling system uses networks to softly-weight uncertain information.

More information leads to better search. One of the problems with the formation labeling system is that more ranking rules can degrade the search (see Chapter 4). In the play labeling system, more information will improve the search if approximately equal amounts of detail are added to each play description. This issue is discussed further later in this section. In addition to more information resulting in more efficient search, as more information is added, less effort should be required to find a model hypothesis. The modularity of the multi-agent system achieves this property. Once a sufficiently rich set of visual networks is designed and tested, then it is easy to incorporate that information into temporal structure descriptions. Similarly, the more visual networks that have been constructed, the more quickly new networks can be designed. This modularity, however, is achieved at the expense of making non-Bayesian independence assumptions. Adding a new action involves only creating one new temporal structure description, not modifying existing descriptions, because each multi-agent action is modeled using an independent network.

Separation of search control and rule set. The knowledge engineer does not need to consider the impact of a modeling decision on control directly because the multi-agent recognition system proposed here performs probabilistic evidence propagation instead of traditional search. Indirectly, however, a poor modeling decision (e.g. in a visual network) can increase the complexity of a network substantially and in non-obvious ways. To minimize such problems, the system can alert the knowledge engineer when a Bayesian network with a unmanageable clique size has been designed.

Typical solutions easy to detect. A desirable property in a recognition system is that typical solutions require the shortest time to detect but that less typical solutions can (eventually) be detected. The multi-agent networks require the same time to compute any solution. They do exhibit the property that the likelihood value for good solutions quickly rises above poor solutions when action commences. The networks, however, offer no mechanism by which to detect unusual situations using a longer, more comprehensive search, as the formation labeling system does.

Conniving and tweaking search. Conniving and tweaking search heuristics are not relevant for the multi-agent network framework because networks directly consider all evidence simultaneously.

Maintaining uncertainty. The networks also do not disregard uncertain information. Uncertainty is preserved from evidence detectors through the generation of play likelihood curves using multi-agent networks.

Evaluating partial interpretations. The multi-agent recognition system is causal, using data observed from the start of the play to the current time. The networks permit the observation of partial information. Some observations, however, are particularly important because they influence many features (e.g. the position of the ball).

Trade space for time. The RETE algorithm trades memory space for computation time in rule-based expert system search [For81, For82]. The proposed representation is not amenable to this strategy. The algorithm, however, is highly parallelizable because each multi-agent network can be run independently.

Exploiting Independence. Independence is exploited within Bayesian networks in this representation, but the models of action do not provide any insight or mechanism that might permit automatic exploitation of independence, as mentioned in Chapter 4. The algorithm assumes the availability of a "reset" point, used to initialize each network.

Comparing across class. There is no need for a mechanism that ranks two labels across classes because this system is not attempting to find a consistent labeling of a scene, as the formation labeling algorithm does. Such a ranking might decide that at a given time or time interval, a player is more likely to have the goal to do A than B. Could visual networks be used for this comparison? Perhaps, however, there are problems with comparing the output of two visual networks. The networks are assumed to be independent, and each network can be considering different features. The network designer tunes the networks so that near perfect examples return high likelihoods. The networks are tuned *independently*. Therefore, there is probably too much variability in the numeric outputs to directly compare two networks with similar likelihood values. Networks with large differences in output likelihoods, however, can probably be compared. An analogous problem occurs when comparing multi-agent networks, which is discussed shortly.

Using a priori information. A priori information is easy to incorporate into the proposed framework by simply setting the appropriate evidence or internal state nodes or adjusting the prior probabilities of the multi-agent network main nodes.

Fuzzy spatial and temporal definitions. Information from fuzzy spatial detectors (e.g. *behind*, *near*) is propagated throughout the recognition process, so that subtle evidence is not ignored. The temporal analysis functions preserve the uncertainty output by the visual goal networks when they compute *before, around,* and *observed* relations using intervals without well-defined boundary conditions.

7.5.4 Network structure

This structure discusses some implications of the multi-agent network structure.

How much to label. In the formation labeling system, labels are assigned to objects in the scene, which is a natural assignment to make. Alternatively, labels can be assigned to every 2d image region [SF91]. In the temporal domain, what to label is less clear. In fact, in this work, labels are associated with temporal intervals, where the "intervals" are defined by likelihood curves computed using goal detectors; labels are not "assigned" to coherent objects.[4] The concept of "covering the space" with labels to detect a global, consistent labeling is more difficult to define in the temporal domain. Temporal labels can overlap in time, and some labels will involve multiple agents while others involve individual agents. Therefore, some metric of "fully labeled" is required to extend the formation labeling system to the temporal domain to find a consistent and fully labeled description of activity. One metric might be that each agent has at least some goal assigned at each time, but there are clearly times when multiple goals are active for a player at some time. In this system, detectors are either run or not, but no decision is made on how much of the scene to label with agent goals.

Domain rules intuitive to encode. 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 annoying to extend due to hidden dependencies between rules (see Chapter 4), 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

[4] The analogy might be if labels were assigned in the CONDOR system [SF91] to regions with probabilistic boundaries instead of sharp, deterministic boundaries.

made to keep the descriptions at approximately the same level of descriptive detail to avoid unwanted biases.

Modularity. Implementing the football-specific detectors without use of smaller, generic network components that could be inserted as evidence proved impractical. However, using virtual evidence to insert results of networks into other networks results in incorrectly modeled independencies among variables. Some of the criteria for selecting Bayesian networks over other methods such as fuzzy logic listed in Chapter 5, thereby become suspect. This work has investigated two different representations for knowledge: Bayesian networks and a rule-based system. Both representations require difficult knowledge engineering and make invalid assumptions about independence of variables. Other methods reviewed in Chapter 5, such as fuzzy logic, also make known, invalid assumptions. When modeling a large real domain, will the selection of one method for representing uncertainty over another ultimately have a large impact in the effectiveness of the recognizer? The answer probably depends upon exactly how the systems are utilized. Most likely, a system for a complex task can benefit from multiple representations. To integrate evidence in small space-time windows, Bayesian networks with simple structures (e.g. perhaps TAN networks) seem appropriate for modeling the interaction between a set of variables. Perhaps with an appropriate data set, the networks can be estimated automatically, freeing the knowledge engineer from the difficult task of designing visual networks. An extensional, rule-based uncertainty metric like fuzzy logic might prove sufficient, convenient, and more flexible than network models at the level at which the output of individual goals are integrated and where some traditional search may need to take place.

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. Consider the optional action in Figure 7-5 for the p52maxpin and p56yunder. Here the actions are only "optional" if the player has no reason not to perform them. If the player blocks, for example, then the action would not get performed. In that case, the play with the optional action is not any less of a p52maxpin, for example, because it does not include the optional action. As in Chapter 4 with the formation labeling system, here the best description would be one that considers whether the optional play action is *consistent* with a description for a play given the actions of the player who would otherwise perform the action.

Observing positive evidence. Optional evidence is problematic, in part, because each play detector is probabilistically summing up positive evidence for some class. The play is being observed from the starting frame through the current frame. Temporal relationships like *before* have no "temporal extent" limitations (e.g. the system will detect A *before* B even if A regardless of whether A occurs just before B or ten hours before B). Therefore, given noisy detectors, if the system were to observe a long multi-agent action, eventually the detector's likelihood value would rise as features were randomly observed. Two extensions could be used to alleviate this problem. First, the temporal descriptions could be augmented with quantifier descriptions, such as "just" or "long" before and the temporal comparison functions adjusted accordingly. The second extension is to place a limit on the amount of data a particular network can observe before it is reset or deactivated. An offensive play, for example, almost never lasts more than 15 seconds; therefore, a multi-agent network active for longer than that period of time should no longer integrate evidence information. External networks for applying the multi-agent network classifiers are not explored in this work.

Hierarchy and relative importance. The multi-agent networks implicitly encode the relative importance of the attributes. For example, in Figure 7-6, 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 slightly more in the classification decision than evidence from *xor* nodes. Currently, these conditional probabilities are set by the knowledge engineer during system construction as described in Section 6.7.2. 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.

Learning

The multi-agent networks exploit low-order temporal relationships. Although the system performs well on the current data set, the system would be far more useful if components could be learned from a labeled data set.

Naive and TAN networks can be learned from large, error-free data sets that span the space of possible attribute combinations [FGG97]. In this problem domain, such a data set was not obtainable. As the number of agents in a scene increases, the size of the attribute space exponentially increases. In addition, temporal attributes have uncertain boundaries, requiring that substantially more data be obtained from multiple users if probabilistic intervals are to be determined.[5]

Learning could potentially improve the multi-agent recognition algorithm developed in this work by simplifying the task of knowledge engineering. Each component of the algorithm where learning might be used is discussed briefly below.

$P(E|S)$ **in visual networks.** The evidence detection functions are not perfect indicators of certain states, especially given the noisy data. Therefore, given labeled agent states, $P(E|S)$ could be estimated. Accurate estimates would improve the performance of the visual networks.

Link structure of visual networks and multi-agent networks. Learning the linking structure of a Bayesian network has proven difficult in practice, even with error-free, complete data [Hec95]. The algorithms use a fully labeled data set combined with an iterative search (typically expectation-maximization) over the link structure with fixed random variables to maximize the likelihood produced by the network for positive examples. These methods cannot be applied to the multi-agent networks, which have far too many variables, and with a relatively small and incomplete data sets.

Probability tables in multi-agent network. The probability tables in the multi-agent network could be learned given a complete data set. The values represent the relative weights of the relationships represented in the network. Obtaining such a data set for a multi-agent problem with 11 agents would require thousands of hours of tedious labeling.

Overall, the effort required to obtain a database that is sufficiently large and detailed was beyond the scope of this project. Unfortunately, this problem is common to other multi-agent domains, where in general it will be difficult to obtain databases large enough for learning. These databases will need high-level labels of action (e.g. play names), mid-level labels of action (e.g. blocking) and low-level labels of evidence (e.g. inContact) annotated in order to allow the use of most existing learning algorithms.[6]

---

[5] If multiple users are asked to label the same action, marking start and end points when the action occurs, an activation distribution will provide probabilistic data indicating the action's fuzzy interval for some example.

[6] One attempt has been made to acquire some data to aid in the automatic construction of visual networks. A java labeling interface is currently running over the Internet. It allows football fans to perform small labeling tasks, such as indicating when players begin and end "blocking." If enough Internet users can be found to label small pieces of data, a data set may be acquired of sufficient size to estimate some network priors from data. So far, only small amounts of data have been collected. The system can be seen at http://ongar.media.mit.edu/intille/Demo.html.

7.6 Applying the representation in other domains

The representation has been applied to the football play recognition task, but as described in Section 2.3, there are some characteristics that are unique to this domain. This section briefly considers how the algorithm might be applied to the two multi-agent domains introduced in Chapter 2: a traffic intersection and making oatmeal in the kitchen.

First, consider the traffic scene example described in Chapter 2. Important actions include waiting (for some other vehicle), moving (with respect to the road or some other vehicle), turning (with respect to the road), passing by (another vehicle), giving way (to some other vehicle), stopping, and following. Most of these actions would be described at the visual network level in the current system because they are generally short interactions between two objects (e.g. two cars, a car and a road segment, a car and a pedestrian). A temporal structure description might be created for higher-level actions, however, such as “stopping for gas.” This particular example has been studied by Nagel *et al*, and therefore, it is used as an example here [NKHD95, GN98, HN98]. Their system for recognizing events like “stopping for petrol” that uses situation trees was described briefly in Section 5.5.2. That method and the system in this work share some characteristics. In both systems, low-level evidence is “fuzzily” quantized and then sets of evidence features are used to detect mid-level single-agent actions like “turning” and “stopping.” Nagel *et al* use a fuzzy metric temporal logic [SN] to integrate uncertain information; here, Bayesian networks are used.

Longer temporal actions are recognized using different mechanisms. Nagel *et al* define a “situation graph” that hierarchically represents ordered sequences of sub-actions. As evidence is observed, the graph is traversed. The representation in this thesis would represent an action like “stopping for petrol” as a temporal structure description with one agent and *before* and *around* relationships specified for the goals of that single agent. If some actions are more critical than others, the knowledge engineer would indicate these actions in the temporal structure description, and the actions would be weighted appropriately in the conditional probabilities.

As the action takes place, the multi-agent network (here with one agent) would return the likelihood of observing the action. The representation developed here may prove to be somewhat less brittle than the FMTL situation graph framework because uncertainty propagates over the entire action description. The situation graph, however, explicitly represents the hierarchical organization of a structured action, and at any time it can indicate precisely which part of the action has been observed and which part should be observed next. An interesting possibility for future research is to consider using visual goal networks as primitives in a situation graph framework and using low-order temporal relationships as evidence to propagate situation graph reasoning.

The remainder of this section discusses how the multi-agent network representation would be used to recognize multi-agent action for the cooking oatmeal task described in Section 2.1.2. A temporal structure description for this action might appear as follows:

```
(goalTeam oatmeal "Collaborative cooking oatmeal"
```

```
  (agentGoal obj1 (agent (obj1 (person-1 person-2)))
     (goal obj1_act1 "open (obj1 cabinet)")
     (goal obj1_act2 "getObject (obj1 bowl)")
     (goal obj1_act3 "getObject (obj1 oatmeal)")
     (goal obj1_act4 "fillObject (obj1 bowl)")
     (goal obj1_act5 "moveTo (obj1 sink)")
     (goal obj1_act6 "manipulate (obj1 faucet)")
     (goal obj1_act7 "open (obj1 microwave)")
     (goal obj1_act8 "putSomethingDown (obj1 microwave)")
     (goal obj1_act9 "pickSomethingUp (obj1 microwave)")
     (before obj1_act1 obj1_act2)
     (before obj1_act1 obj1_act3)
     (before obj1_act2 obj1_act4)
     (before obj1_act3 obj1_act4)
     (before obj1_act4 obj1_act5 obj1_act6 obj1_act7 obj1_act8)

  (agentGoal obj2 (agent (obj2 (person-2 person1)))
     (goal obj2_act1 "moveTo (obj2 refrigerator)")
     (goal obj2_act2 "open (obj2 refrigerator)")
     (goal obj2_act3 "getObject (obj2 milk)")
     (goal obj2_act4 "putSomethingDown (obj2 counter)")
     (goal obj2_act5 "open (obj2 cabinet)")
     (goal obj2_act6 "getObject (obj2 bowl)")
     (goal obj2_act7 "moveTo (obj2 (closestPerson))")
     (goal obj2_act8 "put (milk (closestBowl))")
     (goal obj2_act9 "put (milk (2ndClosestBowl))")
     (before obj2_act1 obj2_act2 obj2_act3 obj2_act4 obj2_act5
     obj2_act6 obj2_act7))

   (before obj1_act8 obj2_act7)
   (before obj1_act9 obj2_act8)
   (before obj1_act9 obj2_act9))
```

Note that in this particular case, there are many ways in which individual agents could perform the subtask of the action. Therefore, it might be possible to recognize this typical behavior by simply recognizing that actions happened in the right order (checking for low-order temporal consistency), regardless of which agent performed them. Therefore, a description that might work as well and be significantly easier to specify might encode the group action using visual networks applied to every agent. For example:

```
(goal act1 "moveTo (anyAgent refrigerator)")
(goal act2 "open (anyAgent refrigerator)")
(goal act3 "getObject (anyAgent milk)")
(goal obj4 "getObject (anyAgent butter)")

⋮

(before act1 act2 act3 act4)
```

A soft-weighting of the temporal relation components may be sufficient to recognize the multi-agent behavior because the activity is highly structured in time. This approach of avoiding agent assignment, however, is only feasible when the number of agents being monitored for action components in the environment is small and the actions each agent performs are distinctive. In football, for example, which agent performs which action is important information that helps discriminate plays.

The temporal structure description designer is required to make many design decisions (especially when two agents are explicitly modeled). The designer must specify which primitive actions to use and which temporal relationships are most important. Unfortunately, the process of doing so here feels much less natural than in the football domain.

The problem is that the cooking oatmeal task is an example of a multi-agent action that can be performed in a large number of ways because which agent does which tasks is not necessarily important. Further, many tasks do not need to be performed in a particular order. Therefore, it is difficult to specify what a "typical" example of coordination will "look like." The components of the description must be observed if this action takes place, but the temporal relationships between observations could occur a large number of ways [Pin99]. This activity is less structured than a football play or a scene at a gas station.

An additional problem with the cooking oatmeal example (and a characteristic not found in the play recognition football domain as mentioned in Chapter 2) is that there is the potential for frequent collaborative re-planning on the part of the agents. Therefore, although a temporal structure description can be created, it requires careful design by a knowledge engineer who needs to encode as much variability in coordination as possible. This task could potentially be difficult to recognize without more explicit reasoning about agent beliefs. How to naturally extend the idea of using low-order relationships to recognition of action such as the cooking oatmeal example is a subject of future work.

7.7 Summary: results and evaluation

The representation developed in Chapter 6 has been applied to the task of recognition of American football plays. The algorithm's performance suggests that using low-order temporal comparisons between agent goal actions is a viable approach for recognizing highly structured, multi-agent action when evidence from the comparisons are softly-weighted using Bayesian graphical network structures. This section discussed some problems with the algorithm and how the algorithm might be applied to other multi-agent recognition domains. Potential future work is discussed in the next chapter.

Chapter 8

Contributions and future directions

8.1 Contributions

The thesis 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 main contributions of this work are summarized below.

- The first contribution is identifying the properties of multi-agent action and exploring how those properties influence the development of a viable recognition algorithm for a real-world, multi-agent recognition task. It has been argued that the multi-agent action recognition task differs from the single-agent action recognition task because the representation must encode *non-geometric* spatial and temporal *relationships*. There are an exponential number of these relationships, and modeling agent interaction using reasoning about "intention" of agents requires intractable modal logic inference. Further, existing intentional reasoning systems make assumptions (e.g. clearly defined temporal intervals) that are impractical for domains with noisy, visual input. These issues were discussed in Chapter 2.

- Although the football play recognition task is just one multi-agent recognition problem with some simplifying assumptions, the football domain has been used to identify general multi-agent recognition issues. In particular, development of a player labeling task in static imagery in Chapter 4 motivated the idea that a large number of relatively simple, context-based comparisons between properties of multiple objects can be used to evaluate an enormously large search space. The same analysis demonstrated the importance of developing a representation that softly weights uncertain spatial and temporal evidence.

- An analogy between static object recognition and multi-agent action recognition introduced in Chapter 5 further motivated the idea that a large set of simple, binary comparisons can be used to recognize structured objects. This idea has been applied to the temporal domain in Chapter 6.

- This work has demonstrated that some 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 proposed representation is used to recognize typical collaborative activity for a real task using real data. The representation, however, does not support reasoning about atypical instances of action. Essentially, the representation proposed here can be used to recognize "what coordinated actions look like" not "what collaborative actions are."

- Chapter 6 introduces a representation to recognize multi-agent action motivated by prior work in static object recognition. The object recognition work suggests that, given a complex object, unary and binary matching constraints can be used to search for an object match in feature data and that higher-order feature comparisons generally escalate computational cost without resulting in significant performance gains [Gri90]. This work extends that observation to the temporal domain and demonstrates a recognition system based on the idea that many low-order temporal matches between an action model and an observed data sequence typically implies a correct match has been obtained for a highly structured, multi-agent action.

- This work demonstrates that 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 some collaborative activity. This shows how modular Bayesian networks can softly weigh uncertain perceptual evidence to compute likelihood activation curves for the goals of individual agents. These networks map from low-level, trajectory features to high-level, semantic concepts used by experts to describe action. They are the building blocks for group action recognition. The goal detectors integrate cues computed within a local space-time region centered on each player and keep the number of features computed manageable by using deictic (i.e. first-person) feature references.

- Specially structured Bayesian networks encoding relationships between goals of individual agents are proposed as a mechanism by which to recognize multi-agent collaborative action. The networks, which typically consist of several hundred binary nodes, are constructed automatically from an intuitive description of activity provided by a domain expert. These networks probabilistically integrate evidence from detectors computing low-order temporal relationships between the agent goals. The networks use a hierarchical, naive network structure, where the low-order temporal dependencies are encoded *within* nodes instead of between nodes (in links) to achieve efficient evidence propagation. This structure permits efficient and exact propagation of uncertainty but still encodes low-order temporal and logical constraints. The test system and results described in this thesis show the network representation is sufficiently strong to detect collaborative multi-agent action for a difficult task from noisy, perceptual data.

- Several contributions of this work relate to the practical issues that must be addressed when developing a recognition system for a complex domain. For example, the use of modular belief networks is proposed as a necessary approximation to make knowledge engineering practical. The networks are generally nested, with domain-independent networks (e.g. relating to object motion and object interactions) used as components in domain-specific networks.

- The network representation proposed in this work models three types of uncertainty: uncertainty in the data and sensing, uncertainty in the action component detectors and their semantics, and uncertainty in the multi-agent action descriptions themselves, resulting from variability in action execution.

- The representation proposed here could be used as a pre-processing step to eliminate some multi-agent action from consideration when a scene is being annotated. The representation is best at distinguishing actions that differ by more than just a small set (e.g. 1-2) criterion (see Chapter 7). The likelihoods computed for two similar plays may be nearly identical due to noise (both in the signal and the scene). The representation, however, can be used to eliminate other potential plays. The temporal structure description could then be used to construct networks designed to specifically differentiate two plays.

The recognition of multi-agent action is a problem that has not yet been actively explored in the computer vision community. Therefore, the final contribution of this work is in identifying some areas for future work based on the limitations of the representation proposed here.

8.2 Directions to explore

Reasoning about uncertainty. In this work, it has been argued that a representation for multi-agent action must represent uncertainty. Bayesian networks were selected to represent uncertainty over other competing methods (e.g. fuzzy logic) because they propagate all information using Bayesian inference. Unlike fuzzy logic propagation, networks designed with precise conditional probability values and an appropriate network structure will not produce results inconsistent with first-order logic [RN95]. Unfortunately, to use Bayesian networks in practice, approximations such as overly-strong independence assumptions (resulting from modularity) are required. Conditional probabilities are often estimated. The structure of the networks are typically simplified from the "ideal" structure considerably. Modeling real-world situations *without* these approximations requires impractically large networks. Consequently, it is difficult to argue that Bayesian networks as they are applied for most real-world modeling tasks are necessarily the only appropriate or sufficient mechanism for representing uncertainty. It may be that low-order constraints could be represented adequately in a fuzzy logic framework and that the logical framework would provide additional benefits (e.g. perhaps allowing better representation of optional actions (see Chapter 7)). Additional research on the components of a multi-agent representation will help clarify which properties in the representation of uncertainty are most critical.

Encoding negative information. The play recognition likelihood curves shown in Chapter 7 are monotonic. Monotonicity occurs because the multi-agent networks have binary nodes with conditional probabilities that weight positive evidence but do not weight negative evidence. Weighting negative evidence properly requires nodes with trinary instead of binary states in order to differentiate a negative observation from a situation when no data has been observed (e.g. the states might be $\{$*observed, notObserved, noEvidence*$\}$ instead of simply *observed* and *notObserved*, as in the framework described here). In the system proposed in Chapter 6, however, each node in the multi-agent networks uses all observed data from the start of observations to the current time. Consequently, detectors more complex than the temporal analysis functions that have been proposed are required to differentiate *notObserved* and *noEvidence* states. In fact, these detectors will probably need to account for the amount of evidence observed and possibly relationships between other detectors in order to make this distinction. Therefore, encoding negative evidence most likely requires a stronger model of the interaction between temporal intervals with ill-defined boundaries.

Detecting "none of the above" actions. Related to the problem of encoding negative information is the problem of detecting "none of the the above" when an example should not match with any of the known action models. The algorithm developed in this thesis will not reduce the likelihood that a play has been observed as more data is observed. Therefore, to classify an example as "none," the likelihood value for all known actions must stay low. In the current system, as described in Chapter 7, this is not always the case. The problem is that many plays look similar, particularly at the start of the play, and therefore, a significant percentage of action components in the temporal structure description will match with many incorrect models. A richer temporal model using some negative information might permit better differentiation of plays by eventually reducing the likelihood for a given play if the end of the play is inconsistent with the model.

Extending the temporal representation. The richness of the temporal representation used here can be extended in several ways. One straightforward extension would be to add qualifiers to the temporal structure descriptions (e.g. A *just before* B or C *about 1 second before* D). Similarly, the knowledge expert might allow the system to differentiate two similar plays by indicating the relative importance of particular temporal relationships (e.g. "The most important characteristic of this action is that A is *before* B"). Relative weights would then influence the conditional probability values automatically assigned to the multi-agent network, so that "important" relationships are weighted more heavily than other relationships. Requiring such detailed temporal structure descriptions places a larger burden on the knowledge engineer. Further, in some cases, these distinctions could make the system more brittle if they are too hard to detect or if the knowledge engineer does not model plays at approximately the same level of descriptive detail. Exploring these tradeoffs is left for future work. An alternative extension, which would require substantial modifications to the system presented here, might use approximate temporal propagation reasoning about interval relationships (e.g. see Pinhanez [Pin99]). Most likely, such an extension would require substantially more computation but would simplify the extension of this work to other domains with less-structured, multi-agent collaboration.

Learning "visual networks." Some mechanism is needed to recognize agent goals using evidence from local space-time windows centered on agents. In this work, visual networks are proposed. In other work, state machines using fuzzy metric temporal logic are used [HN98]. In both cases, the variables, dependencies between variables, and "uncertainties" (i.e. probabilities or fuzzy measures) are provided manually. Given the discussion in Section 7.5 on the compromises necessary to apply uncertainty representations, several mechanisms may be adequate for representing these types of actions. In fact, given a richly labeled data set, it may be possible to automatically construct a robust TAN classifier using deictic features or a probabilistic decision tree classifier. Alternatively, it may be possible to construct a classifier with performance equivalent to that of a visual goal network, using a supervised learning process where a user interactively runs the classifier on examples and iteratively specifies which features are important for recognizing particular goal actions.[1]

Application to interactive environments. As stated in Chapter 2, the structure of a football play somewhat simplifies the problem of recognition of the team action. Other domains of interest with group activity, however, such as interaction in everyday environments like the home, do not have a printed playbook of actions. As indicated by the discussion in Section 7.6, developing temporal structure descriptions can currently require a significant amount of guesswork. In the football domain, the coach generally knows what's important and what's not. Conversely, when describing action in an everyday, cluttered environment, this is not so clear. In fact, the temporal structure will probably need to be weakened somewhat to accommodate greater variability of the unstructured action. The notion of using low-order temporal comparisons between actions, however, is likely to still prove useful in this case. Only the encoding of the description and its conversion to a probabilistic framework for integrating evidence may need to be changed. Unfortunately, the multi-agent interactive environment domain suffers from the same shortage of annotated data problem as the football domain. It may soon prove easier to establish large, shared annotated object trajectory data sets, however, given an emerging interest in interactive environments that recognize the behavior of occupants (e.g. see [Pro98]) and sometimes use tagging systems to track objects.

Feature selection. Agent-based (i.e. deictic) features are one way to manage the size of the feature space in multi-agent domains. Another way is by specifying which agent goals are relevant in the temporal structure description. Another option is to use feature structure suggested by linguistic descriptions of high-level action. For instance, "drop back" is defined in a football dictionary [Foe92] as follows:

[1] The web interface for labeling actions in football plays over the Internet is one attempt to obtain a data set that is sufficiently rich to test automatically generated visual networks.

> To back-pedal. As soon as the QB receives the ball from the center on a passing play, he retreats, looking downfield in the direction of his receivers. Then he stops, "sets up" and prepares to throw.

Just by performing keyword spotting on this description, several features of importance can be identified: QB, receives ball, C, pass play, retreats, receivers, throw, etc. The question is whether these keywords (and possibly one or two-level deep inference about keywords to determine relative importance given a domain) can be associated with concepts in the domain and general concepts about motion in order to drive a supervised learning process or automatically weight components of a description. For example, can a word like "retreat" be linked in some way to the domain-independent concept of "move away from?" "Move away from" might have generic feature detection functions implemented. The algorithm could then identify evidence like *moveAwayFrom(QB,C)*, *moveAwayFrom(QB,receivers)*, and *moveAwayFrom(QB,Ball)* as potentially relevant during the learning process without manual specification.

In summary, future work should focus on extending the idea of low-order temporal relationships to represent multi-agent action. Some method for integrating uncertain information is required, although Bayesian networks alone are probably insufficient; 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.

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 promising framework for recognition of highly structured, multi-agent action from perceptual data and will hopefully motivate other researchers to propose new representations that can represent and identify the interaction of agents in everyday scenes.

Bibliography

[AC87] 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.

[AFFH86] J. Azarewicz, G. Fala, R. Fink, and C. Heithecker. Plan recognition for airborne tactical decision making. In *Proc. Nat. Conf. on Artificial Intelligence*, 1986.

[AFH89] 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.

[Ago90] J.M. Agosta. The structure of Bayes networks for visual recognition. In *Uncertainty in Artificial Intelligence*, pages 397–405. Elsevier Science Publishers B.V., North Holland, 1990.

[AGR88] E. Andre, G. Gerzog, and T. Rist. On the simultaneous interpretation of real world image sequences and their natural language descriptions: the system soccer. In *Proc. European Conf. AI*, pages 449–454, August 1988.

[All83] J.F. Allen. Maintaining knowledge about temporal intervals. *Communications of the ACM*, 26(11):832–843, 1983.

[All84] J.F. Allen. Towards a general theory of action and time. *Artificial Intelligence*, 23:123–154, 1984.

[All91] J.F. Allen. Time and time again: the many ways to represent time. *International Journal of Intelligent Systems*, 6:341–355, 1991.

[Ame98] American football rules. Website, October 1998. http://www4.ncsu.edu/eos/users/f/fdaguet/www/Foot/footRules.html.

[Arm98] Armchair quarterback's guide to football. Website, October 1998. http://24.104.3.96/NFL/League/Info/Rules/index.htm.

[AZNB97] D.W. Albrecht, I. Zukerman, A.E. Nicholson, and A. Bud. Towards a Bayesian model for keyhole plan recognition in large domains. In A. Jameson, C. Paris, and C. Tasso, editors, *User Modeling: Proc. of the Sixth International Conference, UM97*, pages 365–376. Springer Wien New York, Vienna, New York, 1997. Available from http://um.org.

[BC82] R.C. Bolles and R.A. Cain. Recognizing and locating partially visible objects: the local feature focus method. *International Journal of Robotics Research*, 1(3):57–82, 1982.

[BG95] 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.

[BH88] L.B. Booker and N. Hota. Probabilistic reasoning about ship images. In J.F. Lemmer and L.N. Kanal, editors, *Uncertainty in Artificial Intelligence*, volume 2, North Holland, 1988. Elsevier Science Publishers B.V.

[BID+96] A.F. Bobick, S.S. Intille, J.W. Davis, F. Baird, L.W. Campbell, Y. Ivanov, C.S. Pinhanez, A. Schütte, and A. Wilson. The KidsRoom: A perceptually-based interactive and immersive story environment. M.I.T. Media Laboratory Perceptual Computing Section 398, M.I.T. Media Laboratory Perceptual Computing Section, November 1996. Revised September 1997, see also http://vismod.www.media.mit.edu/vismod/demos/kidsroom.

[BID+99] A.F. Bobick, S.S. Intille, J.W. Davis, F. Baird, L.W. Campbell, Y. Ivanov, C.S. Pinhanez, A. Schütte, and A. Wilson. The KidsRoom: A perceptually-based interactive and immersive story environment. *Presence: Teleoperators and Virtual Environments*, August 1999.

[Bis97] J.K. Bishop. Passing route trees. Website (from the Big Jim's Football homepage), July 1997. http://cdl.uta.edu/bigjim/football/trees.html.

[BLM89] T.O. Binford, T.S. Levitt, and W.B. Mann. Bayesian inference in model-based machine vision. In Levitt, Kanal, and Lemmer, editors, *Uncertainty in AI 3*. North Holland, 1989.

[Bob97] 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.

[BOP97] M. Brand, N. Oliver, and A. Pentland. Coupled hidden Markov models for complex action recgonition. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 994–999, Los Alamitos, CA, 1997. IEEE Computer Society Press.

[Bra87] M.E. Bratman. *Intention, plans, and practical reason*. Harvard University Press, 1987.

[Bra90] M. Bratman. What is intention? In P.R. Cohen, J.L. Morgan, and M.E. Pollack, editors, *Intentions in Communication*. MIT Press, 1990.

[Bra92] M.E. Bratman. Shared cooperative activity. In *Philosophical Review*, volume 101, pages 327–341, 1992.

[Bre92] J. Breese. Construction of belief and decision networks. *Computational Intelligence*, 8(4), 1992.

[Car90] S. Carberry. Incorporating default inferences into plan recognition. In *Proc. Nat. Conf. on Artificial Intelligence*, pages 471–478, 1990.

[CG83] P.R. Cohen and M.R. Grinberg. A theory of heuristic reasoning about uncertainty. *The AI Magazine*, pages 17–24, Summer 1983.

[CG93] E. Charniak and R. P. Goldman. A Bayesian model of plan recognition. *Artificial Intelligence*, 64:53–79, 1993.

[Cha91] E. Charniak. Bayesian networks without tears. *AI Magazine*, pages 50–63, 1991.

[Che90] D.M. Chelberg. Uncertainty in interpretation of range imagery. In *Proc. Int'l Conf. Computer Vision*, pages 654–657, 1990.

[CL90] P.R. Cohen and H.J. Levesque. Persistence, intention, and commitment. In P.R. Cohen, J.L. Morgan, and M.E. Pollack, editors, *Intentions In Communication*, pages 33–69. MIT Press, 1990.

[CL91] P.R. Cohen and H.J. Levesque. Teamwork. *Nous*, 25(4):487–512, 1991.

[CN89] P. Clark and T. Niblett. The cn2 induction algorithm. *Machine Learning*, 3(4):261–283, 1989.

[Coo90] G. Cooper. The computation complexity of probabilistic inference using belief networks. *Artificial Intelligence*, 42(2-3):393–405, 1990.

[CSKH97] S. Choi, Y. Seo, H. Kim, and K-S. Hong. Where are the ball and players? soccer game analysis with color-based tracking and image mosaic. In *Proc. Int'l Conf. on Image Analysis and Processsing*, 1997.

[DB97] J.W. Davis and A.F. Bobick. The representation and recognition of action using temporal templates. In *Proc. Computer Vision and Pattern Recognition*, pages 928–934. IEEE Computer Society Press, June 1997.

[Den87] D.C. Dennett. *The Intentional Stance*, chapter True believers: the intentional strategy and why it works. MIT Press, Cambridge, MA, 1987.

[DG93] P. Dagum and A. Galper. Forecasting sleep apnea with dynamic networks models. In *Int'l Conference on Uncertainty in Artificial Intelligence*, pages 64–71, 1993.

[DGH92] P. Dagum, A. Galper, and E. Horvitz. Dynamic network models for forecasting. In D. Dubois, M.P. Wellman, B. D'Ambrosio, and P. Smets, editors, *Int'l Conference on Uncertainty in Artificial Intelligence*, volume 8, pages 41–48, San Mateo, CA, July 1992. Morgan Kaufmann Publishers.

[DH73] R.O. Duda and P.E. Hart. *Pattern classification and scene analysis*. John Wiley & Sons, New York, 1973.

[DHW94] D. Draper, S. Hanks, and D. Weld. A probabilistic model of action for least-commitment planning with information gathering. In *Proceedings of the Tenth Conference on Uncertainty in Artificial Intelligence*, Seattle, WA, 1994. Morgan Kaufmann.

[DKB98] B. Drebs, B. Korn, and M. Burkhardt. A task driven 3d object recognition system using Bayesian networks. In *Proc. Int'l Conf. Computer Vision*, volume 6. IEEE Computer Society, Narosa Publishing House, January 1998.

[DL93] P. Dagum and M. Luby. Approximating probabilistic inference in belief networks is NP-hard. *Artificial Intelligence*, 60(1):141–153, 1993.

[DM90] E.H. Durfee and T.A. Montgomery. A hierarchical protocol for coordinating multi-agent behaviors. In *Proc. Nat. Conf. on Artificial Intelligence*, 1990.

[DP96] P. Domingos and M. Pazzani. Beyond independence: conditions for the optimality of a simple Bayesian classifier. In L. Saitta, editor, *Proceedings of the Thirteenth International Conference on Machine Learning*, pages 194–202, San Francisco, CA, 1996. Morgan Kauffman.

[DR97] M. Devaney and A. Ram. Situation development in a complex real-world domain. In *ICML-97 Workshop on Machine Learning Application in the Real World; Methodological Aspects and Implications.*, July 1997.

[DR98] 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.

[Dru96] M.J. Druzdzel. Qualitative verbal explanations in Bayesian belief networks. *Artificial Intelligence and Simulation of Behavior Quarterly*, pages 43–54, April 1996.

[EH90] C. Elsaesser and M. Henrion. Verbal expressions for probability updates: How much more probable is "much more probable?". In M. Henrion, R.D. Shachter, L.N. Kanal, and J.F. Lemmer, editors, *Uncertainty in Artificial Intelligence*, volume 5, pages 319–328. Elsevier Science Publishers B.V. (North Holland), 1990.

[Elk93] C. Elkan. The paradoxical successs of fuzzy logic. In *Proc. of the Eleventh National Conference on Artificial Intelligence*, pages 698–703, Washington, DC, 1993. IEEE Computer Society, AAAI Press.

[EM93] F. Ennesser and G. Medioni. Finding Waldo, or focus of attention using local color information. In *Proc. Computer Vision and Pattern Recognition*, pages 711–712, New York, NY, June 1993.

[FGG97] J. Friedman, D. Geiger, and M. Goldszmidt. Bayesian network classifiers. *Machine Learning*, 29:131–163, 1997.

[FHKR95] 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.

[FN93] R.E. Fikes and N.J. Nilsson. Strips: a retrospective. *Artificial Intelligence*, 59:227–232, 1993.

[FO96] B.E. Flinchbaugh and T.J. Olson. Autonomous video surveillance. In *Proc. of the 25th AIPR Workshop: Emerging applications of computer vision*, October 1996.

[Foe92] D.P. Foehr. *Touchdown!: A guide to understanding and enjoying football*. Franklin Press, 1992.

[For81] C.L. Forgy. OPS5 user's manual. Technical Report CMU-CS-81-135, Carnegie Mellon University, School of Computer Science, Pittsburgh, PA, 1981.

[For82] C.L. Forgy. RETE: A fast algorithm for the many pattern/many object pattern match problem. *Artificial Intelligence*, 19(1):17–37, September 1982.

[Fri97] J. Friedman. On bias, variance, 0/1 - loss, and the curse-of-dimensionality. *Data mining and knowledge engineering*, 1:55–77, 1997.

[GC90] 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.

[Geo83] M. Georgeff. Communication and interaction in multi-agent planning. In *Proc. Nat. Conf. on Artificial Intelligence*, pages 125–129, August 1983.

[Ger99] N.A. Gershenfeld. *When things start to think*. Henry Hold & Company, 1999.

[Gha97] Z. Ghahramani. Learning dynamic Bayesian networks. In C.L. Giles and M. Gori, editors, *Adaptive Processing of Temporal Information*, Lecture notes in Artificial Intelligence. Springer-Verlag, 1997.

[GJ97] Z. Ghahramani and M.I. Jordan. Factorial hidden Markov models. *Machine Learning*, 29:245–275, 1997.

[GK96] B.J. Grosz and S. Kraus. Collaborative plans for complex group action. *Artificial Intelligence*, 86(2):269–357, 1996.

[GLP87] 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.

[GN96] R. Gerber and H.-H. Nagel. Knowledge representation for the generation of quantified natural language descriptions of vehicle traffic in image sequences. In *Proc. IEEE International Conference on Image Processing*, 1996.

[GN98] R. Gerber and H.-H. Nagel. (Mis?)-using DRT for generation of natural language text from image sequences. In H. Burkhardt and B. Neumann, editors, *Proc. 5th European Conf. Computer Vision*. Springer, 1998.

[Gri90] W.E.L. Grimson. *Object recognition by computer: the role of geometric constraints*. MIT Press, Cambridge, MA, 1990.

[GS86] B.J. Grosz and C.L. Sidner. Attention, intentions, and the structure of discourse. *Computational Linguistics*, 12(3):175–204, 1986.

[GS90] B.J. Grosz and C.L. Sidner. Plans for discourse. In P.R. Cohen, J.L. Morgan, and M.E. Pollack, editors, *Intentions In Communication*, pages 417–444. MIT Press, 1990.

[GS93] J. Gordon and E.H. Shortliffe. A method for managing evidential reasoning in a hierarchical hypothesis space: a retrospective. *Artificial Intelligence*, 59:43–47, 1993.

[GSC+95] Y. Gong, L.T. Sin, C.H. Chuan, H.J. Zhang, and M. Sakauchi. Automatic parsing of TV soccer programs. In *Proc. Int'l Conf. on Multimedia Computing and Systems*, pages 167–174, May 1995.

[GSRL98] W.E.L. Grimson, C. Stauffer, R. Romano, and L. Lee. Using adaptive tracking to classify and monitor activities in a site. In *Proc. Computer Vision and Pattern Recognition*. IEEE Computer Society, 1998.

[GTS94] W. Gilks, A. Thomas, and D. Spiegelhalter. A language and program for complex Bayesian modelling. *The Statistician*, 43:169–78, 1994.

[Had91] P. Haddawy. Time, chance, and action. 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., 1991.

[Ham89] K. Hammond. *Case-Based Planning: Viewing Planning as a Memory Task*. Academic Press, New York, 1989.

[HD91] M. Henrion and M.J. Druzdzel. *Uncertainty in Artificial Intelligence*, chapter Qualitative propagation and scenario-based approaches to explanation of probabilistic reasoning. Elsevier, 1991.

[HD93] M.J. Huber and E.H. Durfee. Observational uncertainty in plan recognition among interacting robots. In *Working notes: Workshop on dynamically interacting robots*, pages 68–75, Chambery, France, August 1993.

[HDW94] 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.

[Hec91] D.E. Heckerman. *Probabilistic similarity networks*. MIT Press, Cambridge, MA, 1991.

[Hec95] D. Heckerman. A tutorial on learning with Bayesian networks. Technical Report MSR-TR-95-06, Microsoft Research, Microsoft Corporation, One Microsoft Way, Redmond, WA 98052, March 1995. Revised November 1996.

[HHNK95] 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.

[HKM+94] 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.

[HL90] J.H.T. Nunes H.J. Levesque, P.R. Cohen. On acting together. In *Proc. Nat. Conf. on Artificial Intelligence*, pages 94–99, Boston, Massachusetts, July/August 1990. AAAI Press/MIT Press.

[HM94] S. Hanks and D. McDermott. Modeling a dynamic and uncertain world I: Symbolic and probabilistic reasoning about change. *Artificial Intelligence*, 66(1):1–55, 1994.

[HN98] M. Haag and H.-H. Nagel. Incremental recognition of traffic situations from video image sequences. In *Proc. of International Conference on Computer Vision Workshop on Conceptual Descriptions of Images*, pages 1–20, 1998.

[Hob90] J.R. Hobbs. Artificial intelligence and collective intentionality: comments on Searle and Grosz and Sidner. In P.R. Cohen, J.L. Morgan, and M.E. Pollack, editors, *Intentions In Communication*. MIT Press, 1990.

[Hol88] S. Holtzman. *Intelligent decision systems*. Addison-Wesley, Reading, MA, 1988.

[Hon94] J. Hong. Plan recognition on the basis of structural and temporal constraints. In C. Backstrom and E. Sandewall, editors, *Current trends in AI planning*. IOS Press, 1994.

[HPF+96] M. Henrion, M. Pradhan, B. Del Favero, K. Huang, G. Provan, and P. O'Rorke. Why is diagnosis using belief networks insensitive to imprecision in probabilities? In *Proceedings of the Twelth Annual Conference on Uncertainty in Artificial Intelligence*, Portland, OR, August 1996.

[HS92] D.E. Heckerman and E.H. Shortliffe. From certainty factors to belief networks. *Artificial Intelligence in Medicine*, pages 35–52, 1992.

[HS93] M. Hild and Y. Shirai. Interpretation of natural scenes using multi-parameter default models and qualitative constraints. In *Proc. Int'l Conf. Computer Vision*, 1993.

[HSA+89] G. Herzog, C.-K. Sung, E. André, W. Enkelmann, H.-H. Nagel, T. Rist, W. Wahlster, and G. Zimmermann. Incremental natural language description of dynamic imagery. In *Wissensbasierte Systeme*, pages 153–162. Springer, 1989.

[HTSN97] M. Haag, W. Theilmann, K. Schäfer, and H.-H. Nagel. Integration of image sequence evaluation and fuzzy metric temporal logic programming. In G. Brewka, C. Habel, and B. Nebel, editors, *Proceedings KI-97, 21st Annual German Conference on Artificial Intelligence*, September 1997.

[Hub96] M.J. Huber. *Plan-based plan recognition models for the effective coordination of agents through observation*. PhD thesis, University of Michigan, 1996.

[IB95] 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.

[Int94] S.S. Intille. Tracking using a local closed-world assumption. Master's thesis, Massachusetts Institute of Technology, August 1994.

[Int98] Introduction to American football. Website (from the rec.sport.college homepage), October 1998. http://www.cae.wisc.edu/ dwilson/rsfc/intro/.

[ISBG99] Y. Ivanov, C. Stauffer, A. Bobick, and W.E.L. Grimson. Video surveillance of interactions. In *IEEE Workshop on Visual Surveillance*, Ft. Collins, CO, 1999.

[JA90] F. Jensen and S.K. Andersen. Approximations in Bayesian belief universes for knowledge-based systems. In *Int'l Conference on Uncertainty in Artificial Intelligence*, pages 162–169, July 1990.

[JCG+97] R.W. Hill Jr., J. Chen, J. Gratch, P. Rosenbloom, and M. Tambe. Intelligent agents for the synthetic battlefield: a company of rotary wing aircraft. In *Proc. of the Conf. on Innovative Applications of Artificial Intelligence*, pages 1006–1012, 1997.

[JCN92] F.V. Jensen, H.I. Christensen, and J. Nielsen. Bayesian methods for interpretation and control in multi-agent vision systems. In *Applications of Artificial Intelligence X: machine vision and robotics, SPIE proceedings*, volume 1708, 1992.

[JCNJ91] F.V. Jensen, B. Chamberlain, T. Nordahl, and F. Jensen. Analysis in HUGIN of data conflict. In P.P. Bonissone, M. Henrion, L.N. Kanal, and J.F. Lemmer, editors, *Uncertainty in Artificial Intelligence 6*. Elsevier Science Pbulishers B.V., 1991.

[Jen89] F. Jensen. Bayesian updating in recursive graphical models by lcoal computations. Dept. of Mathematics and Computer Science Technical Report R-89-15, University of Aalborg, 1989.

[Jen96] F.V. Jensen. *An introduction to Bayesian networks*. Springer-Verlag, 1996.

[JGS97] M.I. Jordan, Z. Ghahramani, and L.K. Saul. Hidden Markov decision trees. In M.C. Mozer, M.I. Jordan, and T. Petsche, editors, *Advances in Neural Information Processing Systems 9*, Cambridge, MA, 1997. MIT Press.

[JJ94] M.I. Jordan and R.A. Jacobs. Hierachical mixtures of experts and the EM algorithm. *Neural Computation*, 6:181–214, 1994.

[JJM85] D.M. McKeown Jr., W.A. Harvey Jr., and J. McDermott. Rule-based interpretation of aerial imagery. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 7(5):570–585, September 1985.

[JL95] G.H. John and P. Langley. Estimating continuous distributions in Bayesian classifiers. In *Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence*, San Mateo, 1995. Morgan Kaufmann Publishers.

[KA86] H.A. Kautz and J.F. Allen. Generalized plan recognition. In *Proc. Nat. Conf. on Artificial Intelligence*, pages 32–37, August 1986.

[Kan92] K. Kanazawa. *Reasoning about time and probability*. PhD thesis, Brown University, Dept. of Computer Science, May 1992.

[KDN93] D. Koller, K. Daniilidis, and H.-H. Nagel. Model-based object tracking in monocular image sequences of road traffic scenes. *Int'l J. of Computer Vision*, 10(3):257–281, 1993.

[KDTN92] D. Koller, K. Daniilidis, T. Thorhallsson, and H.-H. Nagel. Model-based object tracking in traffic scenes. In *Proc. European Conf. Computer Vision*, pages 437–452, May 1992.

[KHN91] D. Koller, N. Heinze, and H.-H. Nagel. Algorithmic characterization of vehicle trajectories from image sequences by motion verbs. In *Proc. Computer Vision and Pattern Recognition*, pages 90–95, 1991.

[Kja92] U. Kjaerulff. A computational scheme for reasoning in dynamic probabilistic networks. In D. Dubois, M.P. Wellman, B. D'Ambrosio, and P. Smets, editors, *Int'l Conference on Uncertainty in Artificial Intelligence*, volume 8, pages 121–129, San Mateo, CA, July 1992. Morgan Kaufmann Publishers.

[Kja95] U. Kjaerulff. A computational system for dynamic time-sliced Bayesian networks. *Int'l Journal of Forecasting*, 11(1):89–, 1995.

[KNO94] H. Kollnig, H.-H. Nagel, and M. Otte. Association of motion verbs with vehcile movements extracted from dense optical flow fields. In J.-O. Eklundh, editor, *Proc. European Conf. Computer Vision*, volume 2, pages 338–347. Springer-Verlag, 1994.

[KP99] 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.

[KSH98] T. Kim, Y. Seo, and K.S. Hong. Physics-based 3d position analysis of a soccer ball from monocular image sequences. In *Proceedings of the Sixth International Conference on Computer Vision*, Bombay, India, January 1998. IEEE Computer Society, IEEE Computer Society Press.

[KYA94] T. Kawashima, K. Yoshino, and Y. Aoki. Qualititative image analysis of group behavior. In *Proc. Computer Vision and Pattern Recognition*, pages 690–693, June 1994.

[Lau92] S.L. Lauritzen. Propogation of probabilities, means, and variances in mixed graphical association models. *J. of the American Statistical Association*, 87(420):1098–1108, 1992.

[Lau96] S.L. Lauritzen. *Graphical Models*. Clarendon Press, 1996.

[LE96] N. Lesh and O. Etzioni. Scaling up goal recognition. In *Principles of knowledge representation and reasoning*. 1996.

[Lev86] T.S. Levitt. Model-based probabilistic inference in hierarchical hypothesis spaces. In *Uncertainty in AI*, pages 347–356. North Holland, New York, 1986.

[LIT92] P. Langley, W. Iba, and K. Thompson. An analysis of Bayesian classifiers. In *Proc. of the Tenth National Conference on Artificial Intelligence*, San Jose, CA, 1992. AAAI, AAAI Press.

[LJC96] J. Liang, F.V. Jensen, and H.I. Christensen. A framework for generic object recognition with Bayesian networks. In P.G. Anderson and K. Warwick, editors, *First international symposium on soft computing for pattern recognition (SOCO '96)*, pages C9–C15, March 1996.

[Loc94] K.E. Lochbaum. *Using collaborative plans to model the intentional structure of discourse*. PhD thesis, Harvard University, 1994. Dept. of Computer Science TR-25-94.

[LS88] S. Lauritzen and D. Spiegelhalter. Local computations with probabilities on graphical structures and their application to expert systems. *Journal of the Royal Statistical Society*, 50:157–224, 1988.

[MB92] W.B. Mann and T.O. Binford. An example of 3D interpretation of images using Bayesian networks. In *Proc. of the DARPA Image Understanding Workshop*, pages 793–801, 1992.

[MC93] S.A. Musman and L.W. Chang. A study of scaling issues in Bayesian belief networks for ship classification. In D. Heckerman and A. Mamdani, editors, *Int'l Conference on Uncertainty in Artificial Intelligence*, volume 9, pages 32–39, San Mateo, CA, July 1993. Morgan Kaufmann Publishers.

[McC98] E. McCorduck. *Understanding American football: the simplest, clearest, most detailed guide for spectators*. NTC Publishing Group, 1998.

[MF92] R.C. Munck-Fairwood. Recognition of geometric primitives using logic program and probailistic network reasoning methods. In *Proc. SPIE: applications artificial intelligence X; Machine vision robotics*, pages 589–600, April 1992.

[MH69] 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.

[MH98] R.J. Morris and D.C. Hogg. Statistical models of object interaction. In *Proc. of the IEEE Workshop on Visual Surveillance – VS'98*, pages 81–85. IEEE Computer Society, January 1998.

[MJS96] R. Mann, A. Jepson, and J.M. Siskind. Computational perception of scene dynamics. In *Proc. European Conf. Computer Vision*, pages 528–539, April 1996.

[MN90] M. Mohnhaupt and B. Neumann. On the use of motion concepts for top-down control in traffic scenes. In *Proc. European Conf. Computer Vision*, pages 598–600, April 1990.

[MNN81] H. Marbuerge, B. Neumann, and H.-J. Novak. Natural language dialogue about moving objects in an automatically analyzed traffic scene. In *Int'l Joint Conf. on Artificial Intelligence*, pages 49–51, 1981.

[Moz98] M. Mozer. The neural network house: an environment that adapts to its inhabitants. In *Proc. AAAI Spring Symposium on Intelligen Environments*, Technical Report SS-98-02, pages 110–114, Menlo Park, CA, March 1998. AAAI Press.

[MPB+94] P. Maes, A. Pentland, B. Blumberg, T. Darrell, J. Brown, and J. Yoon. ALIVE: Artificial life interactive video environment. *Intercommunication*, 7:48–49, 1994.

[NA91] K. Ng and B. Abramson. A sensitivity analysis of Pathfinder: a follow-up study. In B.D. D'Ambrosio, P. Smets, and P.P. Bonissone, editors, *Int'l Conference on Uncertainty in Artificial Intelligence*, volume 7, pages 242–248, San Mateo, CA, July 1991. Morgan Kaufmann Publishers.

[Nag88] H.-H. Nagel. From image sequences towards conceptual descriptions. *Image and Vision Comp.*, 6(2):59–74, 1988.

[Nag94] H.-H. Nagel. A vision of 'vision and language' comprises action: an example from road traffic. *Artificial Intelligence Review*, 8:189–214, 1994.

[NB94] A.E. Nicholson and J.M. Brady. Dynamic belief networks for discrete monitoring. *IEEE Trans. Systems, Man and Cybernetics*, 24(11):1593–1610, 1994.

[Neu89] B. Neumann. Natural language description of time-varying scenes. In *Semantic Structures*. Lawrence Earlbaum Associates, 1989.

[NHH95] 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.

[NKHD95] 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.

[Ole93] K.G. Olesen. Causal probabilistic networks with both discrete and continuous variables. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 1993.

[Ora99] ORAD high-tec systems, 1999. An Israeli company producing sports video technology. www.orad.co.il.

[Orp90] P. Orponen. Dempster's rule of combination is #p-complete. *Artificial Intelligence*, 44:245–253, 1990.

[ORP99] N. Oliver, B. Rosario, and A. Pentland. A Bayesian computer vision system for modeling human interactions. In H.I. Christensen, editor, *Proc. of the First Int'l Conf. on Computer Vision Systems (ICVS)*, volume 1542 of *Lecture Notes in Computer Science*, pages 255–272, Gran Canaria, Spain, January 1999. Springer-Verlag.

[PC93] 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.

[Pea86] J. Pearl. On evidential reasoning in a hierarchy of hyptohesis. *Artificial Intelligence*, 28:9–15, 1986.

[Pea88] J. Pearl. *Probabilistic Reasoning in Intelligent Systems : Networks of Plausible Inference*. Morgan Kaufmann Publishers, San Mateo, CA, 1988.

[Pen96] A. Pentland. Smart rooms. *Scientific American*, 274(4):68–76, April 1996.

[PHP+96] M. Pradhan, M. Henrion, G. Provan, B. Del Favero, and K. Huang. The sensitivity of belief networks to imprecise probabilities: an experimental investigation. *Artificial Intelligence*, 85(1-2):363–397, 1996.

[Pin99] C.S. Pinhanez. *Representation and recognition of action in interactive spaces*. Mit media laboratory, Massachusetts Institute of Technology, 20 Ames St., Cambridge, MA 02139, June 1999.

[Pol90a] 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.

[Pol90b] 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.

[Pop94] A.R. Pope. Model-based object recognition: a survey of recent research. Technical report, University of British Columbia, 1994.

[PPMH94] M. Pradham, G. Provan, B. Middleton, and M. Henrion. Knowledge engineering for large belief networks. In *Proceedings of the Tenth Conference on Uncertainty in Artificial Intelligence*, pages 484–490, Seattle, WA, 1994.

[PR95] J. Pinto and R. Reiter. Reasoning about time in the situation calculus. *Annals of Mathematics and Artificial Intelligence*, 1995.

[Pro93] G.M. Provan. Tradeoffs in constructing and evaluating temporal influence diagrams. In D. Heckerman and A. Mamdani, editors, *Int'l Conference on Uncertainty in Artificial Intelligence*, volume 9, pages 40–47, San Mateo, CA, July 1993. Morgan Kaufmann Publishers.

[Pro98] IEEE Computer Society. *Proceedings of the AAAI Spring Symposium on Intelligent Environments*, number SS-98-02 in Technical Report, Menlo Park, CA, March 1998. IEEE Computer Society Press.

[PW95] 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.

[RB92] 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.

[RB94] R.D. Rimey and C.M. Brown. Control of selective perception using bayes nets and decision theory. *Int'l J. of Computer Vision*, 12(2/3):173–207, 1994.

[RE91] P.L. Rosin and T. Ellis. Detecting and classifying intruders in image sequences. In P. Mowforth, editor, *Proc. British Machine Vision Conf.*, pages 24–26. Springer-Verlag, September 1991.

[Rei80] R. Reiter. A logic for default reasoning. *Artificial Intelligence*, 13(1-2):81–132, 1980.

[RJ93a] L.R. Rabiner and B.-H. Juang. *Fundamentals of speech recognition*. Prentice Hall, 1993.

[RJ93b] L.R. Rabiner and B.-H. Juang. *Fundamentals of speech recognition*, chapter A theory and implementation of hidden Markov models. Prentice Hall, 1993.

[RK94] 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.

[RM94] A.S. Rao and G. Murray. Multi-agent mental-state recognition and its application to air combat modeling. In *Proc. Workshop on Distributed AI*, Lake Quinalt, Washington, July 1994.

[RN95] S. Russell and P. Norvig. *Artificial intelligence: a modern approach*. Prentice Hall, Inc., Upper Saddle River, NJ 07458, 1995.

[RS88] 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.

[RTB98] 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.

[SB93] S. Sarkar and K.L. Boyer. Integration, inference, and management of spatial information using Bayesian networks: perceptual organization. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 15(3):256–272, 1993.

[SB94] L. Stark and K. Bowyer. Function-based generic recognition for multiple object categories. *Computer Vision, Graphics, and Image Processing*, 59(1):1–21, January 1994.

[SC96] B. Schiele and J.L. Crowley. Probabilistic object recognition using multidimensional receptive field histograms. In *International Conference on Pattern Recognition*, Vienna, Austria, August 1996.

[Sea90] J.R. Searle. Collective intentions and actions. In P.R. Cohen, J.L. Morgan, and M.E. Pollack, editors, *Intentions In Communication*. MIT Press, 1990.

[SF91] T.M. Strat and M.A. Fischler. Context-based vision: recognizing objects using information from both 2D and 3D imagery. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 13(10):1050–1065, 1991.

[SG92] L.E. Sucar and D. F. Gillies. Expressing relational and temporal knowledge in visual probabilistic networks. In D. Dubois, M.P. Wellman, B. D'Ambrosio, and P. Smets, editors, *Int'l Conference on Uncertainty in Artificial Intelligence*, volume 8, pages 303–309, San Mateo, CA, July 1992. Morgan Kaufmann Publishers.

[Sha76] G. Shafer. *A mathematical theory of evidence*. Princeton University Press, Princeton, N.J., 1976.

[SHJ96] P. Smyth, D. Heckerman, and M. Jordan. Probabilistic independence networks for hidden Markov probability models. Technical report, MIT, 1996.

[Sho76] E.H. Shortliffe. *Computer-Based Medical Consultations: MYCIN*. Elsevier/North Holland, 1976.

[Sho88] Y. Shoham. *Reasoning about change: time and causation from the standpoint of artificial intelligence*. MIT Press, Cambridge, MA, 1988.

[Sho90] Y. Shoham. Agent-oriented programming. Computer Science Dept. Technical Report STAN-CS-1335-90, Stanford University, Stanford, CA 94305, 1990.

[SL87] G. Shafer and R. Logan. Implementing Dempster's rule for hierarchical evidence. *Artificial Intelligence*, 33:271–298, 1987.

[SLJ98] G. Sudhir, J.C.M. Lee, and A.K. Jain. Automatic classification of tennis video for high-level content-based retrieval. In *Proc. of IEEE Workshop on Content-Based Access of Image and Video Databases (CAIVD'98)*, pages 81–90. IEEE Computer Society, IEEE Press, 1998.

[SM72] G.J. Sussman and D.V. McDermott. From PLANNER to CONNIVER – a genetic approach. In *Proceedings of the 1972 AFIPS Joint Computer Conference*, pages 1171–1179, 1972.

[SN] K. Schäfer and H.-H. Nagel. Fuzzy temporal inference of processes. Technical report, Universität Karlsruhe (TH), Institut für Algorithmen and Kognitive Systeme, AM Fassanengarten 5, 76131, Karlsruhe, Germany.

[SP95] T. Starner and A. Pentland. Real-time American Sign Language recognition from video using hidden Markov models. In *Proc. of the International Symposium on Computer Vision '95*, 1995.

[Sri92] S. Srinivas. Generalizing the noisy or model to *n*ary variables. Technical Memorandum 79, Rockwell International Science Center, Palo Alto Laboratory, 444 High Street, Palo Alto, CA 94301, April 1992.

[STW$^+$92] E. Sonenberg, G. Tidhar, E. Werner, D. Kinney, M. Ljungberg, and A. Rao. Planned team activity. Technical note 26, Australian Artificial Intelligence Institute, Victoria, Australia, 1992. Revised 1994.

[Sup] http://www.nextstep.com/stepback/cycle10/126/superpuck.html.

[Tag93] P. Tagliabue. *Official playing rules of the National Football League*. Triumph books, Chicago, IL, 1993.

[Tam96] M. Tambe. Tracking dynamic team activity. In *Proc. Nat. Conf. on Artificial Intelligence*, pages 80–87, August 1996.

[Tam97a] M. Tambe. Agent architectures for flexible, practical teamwork. In *Proc. Nat. Conf. on Artificial Intelligence*, 1997.

[Tam97b] M. Tambe. Implementing agent teams in dynamic multi-agent environments. *Applied Artificial Intelligence*, 1997.

[Taw97] A.Y. Tawfik. *Changing times: an investigation in probabilistic temporal reasoning*. PhD thesis, University of Saskatchewan, Dept. of Computer Science, Saskatoon, Saskatchewan, Canada S7N 5A9, 1997.

[TB92] A.F. Toal and H. Buxton. Spatio-temporal reasoning with a traffic surveillance system. In *Proc. European Conf. Computer Vision*, pages 884–892, S. Margherita Ligure, Italy, May 1992.

[Thi86] R. Thibadeau. Artificial perception of actions. In *Cognitive Science*, volume 10, pages 117–149, 1986.

[TN94] A. Tawfik and E. Neufeld. Temporal Bayesian networks. In *Int'l Workshop on Temporal Knowledge Representation and Reasoning*, pages 85–92, 1994.

[Tor95] Mark C. Torrance. Advances in human-computer interaction: the intelligent room. In *Working notes of the CHI 95 research symposium*, May 1995.

[TR95a] M. Tambe and P.S. Rosenbloom. Event tracking in dynamic, multi-agent environment. *Computational Intelligence*, 1995.

[TR95b] M. Tambe and P.S. Rosenbloom. RESC: An approach for real-time, dynamic agent tracking. In *Int'l Joint Conf. on Artificial Intelligence*, pages 103–110, August 1995.

[Tra99] Trakus, 1999. A company producing football trajectory tracking devices to be used in the NFL. http://www.trakus.com.

[TSB94] T.N. Tan, G.D. Sullivan, and K.D. Baker. Pose determination and recognition of vehicles in traffic scenes. In *Proc. European Conf. Computer Vision*, volume 1, pages 501–506, Stockholm, Sweden, May 1994.

[TSH95] G. Tidhar, M. Selvestrel, and C. Heinze. Modelling teams and team tactics in whole air mission modelling. In *Eight international conference on industrial and engineering applications of artificial intelligence and expert systems*. Gordon and Breach Science Publishers, 1995.

[VD95] J.M. Vidal and E.H. Durfee. Recursive agent modeling using limited rationality. In *Proc. Int'l Conf. Multi-Agent Systems*, pages 376–383, 1995.

[WB95] 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.

[WB98] A.D. Wilson and A.F. Bobick. Nonlinear PHMMs for the interpretation of parameterized gesture. In *Proc. Computer Vision and Pattern Recognition*. IEEE Computer Society, 1998.

[WBC97] A. D. Wilson, A. F. Bobick, and J. Cassell. Temporal classification of natural gesture and application to video coding. *Proc. Computer Vision and Pattern Recognition*, pages 948–954, 1997.

[Web89] J. Weber. Representing and computing temporally scoped beliefs. In *Int'l Joint Conf. on Artificial Intelligence*, 1989.

[Whi90] J. Whittaker. *Graphical models in applied multivariate statistics*. John Wiley and Sons, Chickester, UK, 1990.

[WL92] R. Weida and D. Litman. Terminological reasoning with constraint networks and an application to plan recognition. In *Principles of knowledge representation and reasoning*, 1992.

[Woo81] B. Woolf, editor. *Webster's new collegiate dictionary*. G. & C. Merriam Co., Springfield, MA, 1981.

[You93] S. Young. *HTK: Hidden Markov Model Toolkit V2.1*. Cambridge Univ. Eng. Dept. Speech Group and Entropic Research Lab. Inc., Washington DC, 1993. Can be obtained at: www.entropic.com.

[YYYL95] D. Yow, B.-L. Yeo, M. Yeung, and B. Liu. Analysis and presentation of soccer highlights from digital video. In *Second Asian conference on computer vision*, volume 2, pages 499–503, 1995.

[Zad78] L.A. Zadeh. Fuzzy sets as a basis for a theory of possibility. *Fuzzy Sets and Systems*, 1:3–28, 1978.

Appendix A

American football terminology and actions

Several online [Int98, Ame98, Arm98] and published [McC98, Tag93] are available that describe the basic rules and notation used in American football.

The fundamentals of the game that are required to understand this work are described in this section. Readers familiar with football will only be frustrated by the over-simplifications in the first few sections and should skip to Section A.5.

A.1 The game

The goal of each team in American football is to throw or carry the ball from one end of the field to the 10 yard region at the opposite end called the *endzone*. The defense team tries to prevent this from happening. The game takes place through a series of *plays*, which typically last between six and 15 seconds. At the start of the play, the offensive team assumes a special configuration, called a *formation*. When the offense moves the ball, the play begins and the players run a planned set of coordinated motions that are designed to confuse or outsmart the defense and to result in forward movement of the ball in the direction of the endzone. The offense can either *pass* and then run or just *run* the ball downfield. The play ends when an offensive player with the ball is *tackled*, meaning that the player's forward motion is stopped or the player is physically thrown to the ground. The play can also end if the ball is passed but is not caught before touching the ground by either an offensive or defensive player. The rules about how many plays a team can run and how the ball switches from the offense to the defense are not relevant to understanding this work. For the purposes of this work, each play can be considered an independent event.

A.2 The objects

Every football play involves 22 players. Each player has a designated position based upon how the player is positioned in the starting formation. The starting position, in some cases, limits which actions an agent can and can't perform. Only certain offensive players, for example, are permitted to catch a pass.

The following player position abbreviations stand for offensive players in images shown in this work: C (center), QB (quarterback), LG and RG (left and right guard), LT and RT (left and right tackle), LTE and RTE (left and right tight end), LFL and RFL (left and right flanker), LSE and RSE (left and right split end), LWR and RWR (left and right wide receiver), HB (halfback), FB (fullback), TB (tailback), LWB and RWB (left and right wing back), LHB and RHB (left and right half back), and LSB and RSB (left and right slot back). The C, QB, LG, RG, LT, and RT are required to be in every play in a particular spatial arrangement, shown in Figure A-1. The C, LG, RG, LT, and RT are called *linemen* because they make up the *frontline*. Sometimes additional players are added to the end of the frontline, but it always includes at least these five. The QB always starts behind the C, although sometimes the QB stands back a few yards, and the C always stands next to the ball.

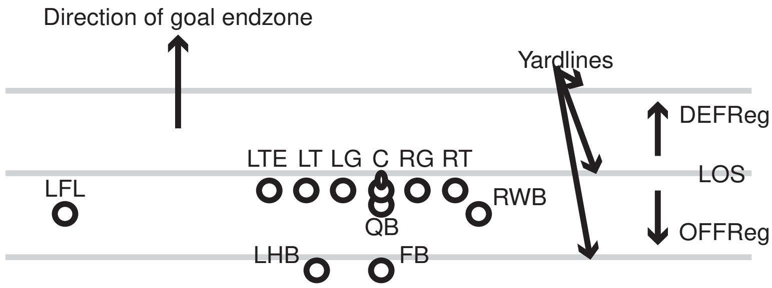


Figure A-1: One offensive starting formation formation.

The examples shown here have been kept simple for clarity. The most important players in the examples are QB (quarterback), C (center), and FB (fullback). The QB typically throws passes or hands the ball to players who will run the ball. The C starts the play by handing or tossing the ball back to the QB and then usually *blocks* for the QB by staying between a defensive player and the ball. The FB will sometimes block for the QB and sometimes will receive a *handoff* or *toss* from the QB and run the ball upfield.

Also important in the examples are four different offensive players called *eligible receivers* because they can catch passes. Usually, they run pre-planned, coordinated trajectories of particular forms to elude nearby defensive agents and then try to catch the ball if it is thrown to them. The FB is also an eligible receiver and sometimes runs one of these *pass patterns.* The receivers in the examples are the LSE and RSE (left and right split end) and LFL and RFL (left and right flanker). The only difference between the split end and flanker position is that in the starting formation the split end is allowed to stand a few yards farther forward than the flanker.

The following player position abbreviations stand for defensive players in examples in this work: LE, RE, NG, LDT, RDT, LLB, MLB, RLB, LOLB, LILB, RILB, ROLB, LCB, FS, SS, and RCB. In some data, the positions of the officials have been tracked and these abbreviations are REF, UMP, SJ, HL, FJ, BJ, and LJ.

The remaining object is the BALL.

A.3 Regions

Figure A-2 shows the dimensions of a football field. The primary unit of measurement is the *yard*[1]. There are two types of regions. Some regions are static, such as the sideline regions. Other regions, here called *relative regions*, are defined based on the starting position of the ball and of the offense and defense players.

The most important relative region is the line of scrimmage (LOS) region, referred to

[1] 1 yard $\approx$ .91 meters

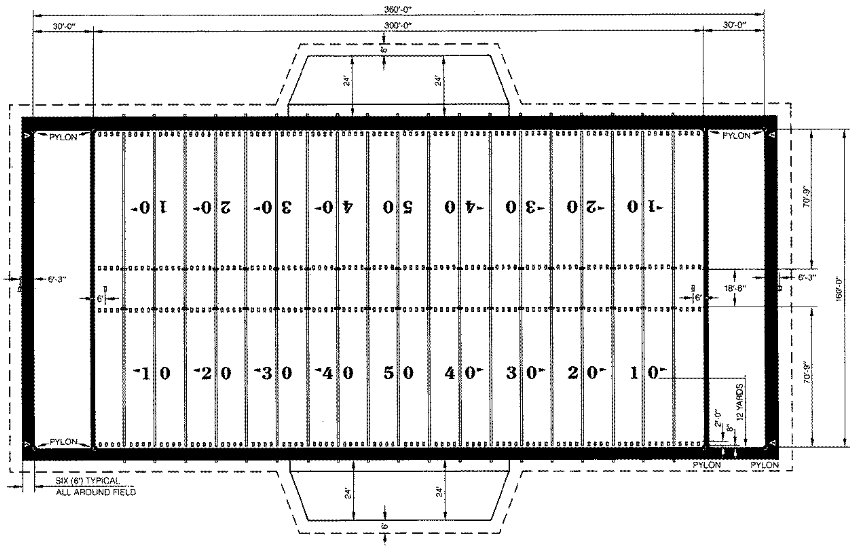


Figure A-2: The dimensions of the football field, from [Tag93].

here as the LOSReg. This is a virtual vertical line that divides the offense and defense at the start of the play, positioned at the center of the ball. In this work, the region of the field on the offense's side of the LOS at the start of the play is called the OFFReg. Similarly, the region of the field on the defense's side of the LOS at the start of the play is called the DEFReg.

The position of players relative to the LOS is often important in football. For example, an offensive player can't pass the ball forward if the ball has already moved from the OFFReg to the DEFReg. Many of the pass patterns (the classes of motion patterns defined by the coach's play definition) are defined relative to the LOS. For example, a player will have the goal to run perpendicular to the LOSReg for 10 yards and then *cut* (i.e. quickly turn at about 90 degree angle) to one direction and run another 5 yards parallel to the LOSReg.

A.4 Plays

A team prepares a repertoire of about 30 plays before a game. The coach will select a play based on the team's current situation and notify the team. Action proceeds as follows. The offensive players get *set* in their starting positions for a brief moment. The offense must then remain completely still until the ball is *hiked*. The ball is hiked by the C player, who either directly hands it to the QB or throws it backwards a few yards to the QB. Although it is possible the ball could be hiked to a player other than the QB, this almost never occurs.

There is one exception to the rule that once the offense is set no players are allowed to move. Up to one offensive team member can move as long as the player's motion is not downfield. Typically the movement is parallel to the LOS region. This player is called the *man-in-motion*.

Although many players are *eligible* to throw a pass, in reality it is almost always the QB. When the QB receives the ball, the offense players start executing the play. The QB will try to *dropback*, by holding the ball and backing up a few yards. During this time some defensive players are trying to tackle the QB in order to prevent the quarterback from throwing the ball. Some offensive players (e.g. the C) will *block* for the quarterback by physically staying between a defender and the ball.

If the play is a running play, the QB will hand the ball to another player (e.g. the FB) who is running by or the QB will toss the ball to a nearby player. The player will then run the ball downfield, usually trying to follow a pre-determined path through the defense, but adjusting the trajectory to avoid defenders and to stay behind offensive players who are blocking. The play ends when the running player is tackled.

If the play is a pass play, after dropping back the QB will look for an open receiver who might be able to receive a pass. The receivers are simultaneously trying to run specified pass patterns. In a coordinated fashion they will change their movement. About this same time, the QB will try to throw the ball to one receiver. If a receiver catches it, he will try to continue running the ball downfield until tackled.

At any time, if a defender catches a pass or if a ball is dropped on the ground by a runner, the defense can pick up the ball and run the ball back the opposite direction.

Plays are specifically designed to look visually similar early in the play to confuse the defense. Therefore, each type of play can start from several different starting formations, and receivers are always running pass patterns even when they know they will not be thrown the ball.

A.5 Actions

The actions used in the examples in this work are briefly described here and labeled using the notation found throughout this paper.

throwPass (OBJ) This is a detector for the action of some player, OBJ, throwing a pass. Typically, the OBJ player is the QB, but there is a group of players called "backs" (in our case, the FB) who can all throw a pass. The pass must be thrown from the OFFReg side of the LOSReg. A pass is usually thrown between two and five seconds after the snap but can be thrown earlier or later. The pass is also usually thrown from the *pocket* region, which is about 5 yards behind the ball's snap position. When the ball is thrown, the BALL object leaves OBJ at a high speed. Other contextual cues that are important are that the play must be in progress (i.e. the ball has been hiked but nobody has been tackled), OBJ must be a player currently in the scene, it must be a pass play, the OBJ must not be getting tackled, the OBJ should have the ball, and the OBJ is not running the ball.

catchPass (OBJ) This is a detector for the action of some player, OBJ, catching a pass. The OBJ player must be an eligible receiver (e.g. FB, LSE, RSE, LFL, RFL) or a defensive player. The passes are usually caught in the DEFReg of the field but sometimes caught near the LOSReg in the OFFReg of the field. Prior to the catch the ball should be traveling at high speed and be following a trajectory that will intercept with the OBJ's trajectory. The BALL then stops suddenly and OBJ is the ballcarrier. Other contextual cues that are important are that the play must be in progress (i.e. the ball has been hiked but nobody has been tackled), OBJ must be a player currently in the scene, the play must be a pass play, and the ball should not have hit the ground.

blockForBC (OBJ) Offensive players block for the ballcarrier (BC) in order to prevent a defensive player from tackling the player with the ball. When OBJ is blocking there is usually physical contact with some defender (or anticipated contact with some defender). The OBJ should be between a defender and the BC or moving between the defender and BC. The OBJ is typically facing the defender. OBJ should not actually be the BC. The play should be in progress.

handoffTo (OBJ1 OBJ2) The ball can be passed between OBJ1 and OBJ2 using a handoffTo action. OBJ1 should be the ballcarrier and then OBJ2 should be the ballcarrier. OBJ1 and OBJ2 should at some point be next to each other (i.e. a handoff is not a toss, which requires the ball travel in the air). Typically OBJ2 is a "back" player (e.g. the FB), and often the OBJ1 is the QB. HandoffTo actions are common about 1-2 seconds into the play, near the pocketReg. Most often, the OBJ2 has some forward motion and moves past the OBJ1.

passPatStreaking (OBJ distance angle inRegion toRegion distLOSatStart) This action is the streak pass pattern. The same pattern can be run from different starting positions and in different ways. The most generic streak pattern is a receiver OBJ running at least distance yards from the LOS at angle to the LOSReg in a relatively straight line. This motion is run in a particular region (i.e. InRegion, which is usually DEFReg) and moving towards a particular region (i.e. toRegion, usually the goal line). Finally, this motion can be a sub-action of a longer component action and therefore the distLOSatStart, or distance from the LOS at the start of the streak, can be specified[2] The OBJ should be an eligible receiver (e.g. FB, LSE, RSE, RFL, LFL) and the play should be in progress. The ball should not have been thrown yet.

passPatCutting (OBJ angle towardsReg locationReg) This action is the cut pass pattern, where an eligible receiver OBJ makes a sudden change of direction in trajectory heading. The change in direction should be about angle degrees. The OBJ turns towards towardsReg while in locationReg, which it typically the DEFReg.

[2]distLOSatStart makes it possible to specify that this action will be recognized even when the receiver first moves downfield doing some motion that's not a streak and then somewhere down field starts a streak pattern.

passPatParaLOS (OBJ distance inRegion towardsReg distLOSatStart) Pass pattern segment where eligible receiver offensive OBJ runs parallel to the losReg for at least distance yards in inRegion heading towards towardsReg and at approximately distLOSatStart yards from the LOS.

passPatCrossing (OBJ distance distOut) Pass pattern segment where eligible receiver OBJ is running at least distance yards across the center of the field (as defined by the horizontal axis where the ball started) at approximately distOut from the LOSReg.

runningBehind (OBJ1 OBJ2) This is the action of OBJ1 running behind OBJ2 with respect to the LOSReg.

A.6 Multi-part actions

Some longer-term composite actions can be constructed using the primitives described above. For example, Figure 2-3 shows typical pass routes used by receivers in football plays[3] The "out 10" pass pattern might be described as follows, "run straight out, perpendicular to the LOS for 10 yards, then make a 90 degree cut to the outside yardline and run a few more yards parallel to the LOS." This action, called out(OBJ, 10 yards) can be described by chaining the action primitives above: passPatStreaking (OBJ, 10 yards, 90 degrees, DEFReg, GOALReg, 0 yards) *then* passPatCutting (OBJ, 90 degrees, OFFLeftSidelineReg[4], DEFReg) *then* passPatParaLOS (OBJ, 2 yards, DEFReg, OFFLeftSidelineReg, 10 yards). All the other patterns can be described in a similar way using a larger set of primitive actions.

A.7 Team actions

A football play is a plan that coordinates the movement of 11 people. Typically plays are written down as diagrams, like the one shown in Figure 2-2 for a play called a *p51curl*. The man-in-motion movement is indicated in gray. The primary pass option is noted by the dotted line. Blocking is noted by a short perpendicular line.

Although the diagram does specify the ideal action when the play is run from one particular formation, there are usually several formations from which the same play can be run. The most characteristic action in this particular example (which remains relatively common between examples) is as follows.

- The P2 player, who is typically a FB, runs forward and blocks to the left or right of the QB, choosing a direction based on defender threats to the QB. Alternatively, if there are no defenders to block, P2 runs a check(3) pass pattern.

[3] Although the concepts they describe are similar, the actual notation used by coaches and football fans varies widely. Here a particular notational scheme has been selected, which was originally obtained from [Bis97].

[4] The left sideline for the offensive team

- The P1 player, some type of receiver, runs a angle(3) pass pattern.
- The P3 player, some type of receiver starting near the RG, runs a shoot(3) pass pattern.
- When P3 crosses with P1, P1 is behind P3 with respect to the LOSreg.
- The P1 and P3 players cut at the same time.
- The QB drops back for five steps.

In some plays, one receiver is designed at *primary* and will receive the ball if all goes well. In this play, P4 is primary, P3 is secondary, P1 is third, and a pass to P2 is unlikely but possible. Once a player catches the ball, the player will run downfield, avoiding defenders, until tackled.

Appendix B

Intentionality case study: SharedPlans

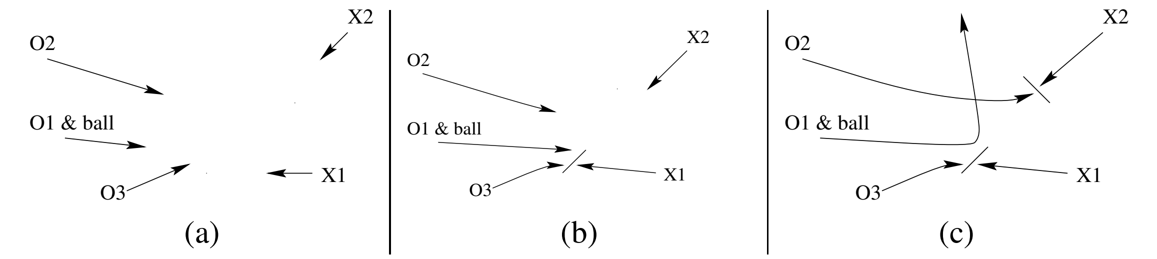


Figure B-1: Three snapshots illustrating action of several players during part of a football play.

This Appendix examines one representation of collaborative behavior and evaluates the representation with respect to visual recognition of multi-agent action. A small example clip from part of a football play is used for illustration. The representation is the SharedPlans mental model proposed by Grosz and Kraus[GK96], which was developed to model speech-act intentionality during person-to-person discourse. The representation was selected because it is currently one of the most comprehensive, general, and well-developed computational theories of intentionality that have been proposed. As the model is applied to the example, the model's representational weaknesses (with respect to its use for visual recognition of multi-agent action) will be noted.

Figure Figure B-1 shows three snapshots of some action during a football play, which will be used for explanatory purposes. There are three offensive players (O_1, O_2, and O_3) and two defensive players (X_1 and X_2). O_1 carries the ball. O_2 and O_3 are the blockers. First O_3 blocks X_1, shown in Figure B-1b, then O_1 slows down and cuts left as O_2 moves past and blocks X_2. O_1 then runs downfield, having avoided both defenders, as shown in Figure B-1c.

B.1 Plans as mental models

The SharedPlans representation of cooperative activity is fundamentally based upon the observations of Pollack[Pol90b]: we can differentiate a "plan to do some act" from a plan "recipe-for-action." STRIPS-like [FN93] plan inference systems try to reason directly about an agent's goal by constructing a recipe-for-action using a library of recipes assumed to be jointly known by the agent, other agents, and the system. Pollack identifies some problems with these approaches. Foremost, a plan to perform recipe actions like `PLANS(Agent, recipe-for-action)` leaves the "state of mind" of the agent unspecified. Pollack argues that such systems cannot reason about invalid plans that agents may hold; further, the systems fail to capture intuitions about the way that beliefs are passed between agents via plan attribution in conversation.

Motivated by these limitations, Pollack considers a plan to be a collection of beliefs and intentions. The fundamental observation is as follows, "For me to have a plan to do β, which consists in the doing of some collection of acts Π, it is not necessary that the performance of Π actually lead to the performance of β. What is necessary is that I *believe* that its performance will do so"[Pol90b]. In other words, a step in a recipe may play a role in a plan only when the agent believes that by doing the step a goal will be enabled.

Pollack defines "having a [single-agent] plan" as follows: An agent A has a plan to do β that consists in doing some set of acts Π, provided that

- A believes that it can execute each act in Π.
- A believes that executing the acts in Π will entail the performance of β.
- A believes that each act in Π plays a role in it's plan.
- A intends to execute each act in Π.
- A intends to execute Π as a way of doing β.
- A intends each act in Π to play a role in his plan.

In some situations, as Pollack has pointed out (see footnote 8 in [Pol90b]), the second condition in her definition of "having a plan" is too strong. In a competitive game like football for example, an agent may try to engage in a recipe for action to a achieve a goal even when the agent is uncertain that each recipe goal can be accomplished. A typical example is when a defender chases an opponent about to make a touchdown and the defender has no realistic hope of catching up and making a tackle. Of course, it is also possible that the recipe actions can contribute to multiple goals which explain the behavior. Perhaps the defender keeps running so that the coach doesn't think he is lazy, or perhaps he keeps running so that he will be on television.

The mental model philosophy impacts visual recognition tasks because Pollack essentially advocates a move from a system-centered representation of plans to an *agent-centered* representation of plans and beliefs. This implies a greater role for agent-centered perception. Take the first condition: A believes that it can execute each act in Π. A acquires this belief through previously known contextual information and perception of the world. If A's perceptual model is flawed or incomplete, A may have beliefs that appear irrational to an omniscient system. Therefore, using the mental model approach to understanding an agent's behavior requires that a recognition system separately model the perceptual input of each agent in the world*from the perspective of the agent.* The implication of this perspective change will be discussed later.

Cohen and Levesque[CL90] have made some additional observations on intentionality, based on the writings of Bratman[Bra90], that are instrumental to any system that tries to use Pollack's belief formulation. Intentionality provides constraints on the reasoning process:

- Intentions pose problems that agents need to find ways to achieve.
- Intentions provide a filter for adapting other intentions, which must not conflict.
- Agents track the success of intentions and will try again if they are not achieved.

- Agents believe their intentions are possible.
- Agents do not believe they will not bring about their intentions.
- Under certain circumstances, agents believe they will bring about their intentions.
- Agents need not intend all the expected side effects of their intentions.

As noted by Bratman[Bra87], intentions modeled as "a kind of persistent goal"[CL90] provide two primary constraints. First, they help to control the computational complexity of decision making in a resource-bounded agent by encouraging stability of decision making once an intention is considered and adopted. Second, future commitment to action using intention makes coordination of future acts possible. A perception system stands to gain from understanding intentionality therefore, because intentional commitment can be used to limit the set of most likely actions at any given time.

Mental plans are a natural representation for modeling discourse because, as Grosz and Sidner[GS86] have argued, discourse streams can be segmented based upon intentional purpose. Since agents engaged in communication are jointly interested in conveying beliefs to one another, they structure their conversation their so that every block of discourse has a coherent intentional focus. Moreover, they insert discourse cues into the conversation that mark when intentional focus has changed. Hence, "discourses are fundamentally examples of collaborative behavior" [GS90]. This collaborative communication allows two powerful assumptions to be made when analyzing discourse that can constrain a reasoning system: reasoning by agents is restricted to the current intentional focus and all of the acts of agents must be explained with respect to the current intentional focus. Lochbaum[Loc94] used these two assumptions in a system that models conversation as partial plan augmentation between two agents – each discourse utterance is assumed to serve the purpose of augmenting a partial shared plan between two agents about the task of current focus.

As Lochbaum points out, however, her assumptions fail to hold in a "keyhole recognition" system where agents are not explicitly using conversational cue mechanisms and where the perceptual stream is not one dimensional[1]. In the football example, for instance, visual input can contain several different interactions occurring simultaneously that need to be parsed into segments using perceptual features. Unlike a discourse, it is not immediately apparent which agents are purposefully interacting with which other agents. Further, most often agents are communicating through the environment, reacting to the visual behavior that they perceive, not high-level conversational acts. The agents may also be inferring the intentions of other agents using visual perception. Consequently, as advocated later in this

[1] A speech stream with only one conversation can be thought of as "one dimensional." A speech stream where multiple conversations are occurring simultaneously and overlaid upon one another is more analogous the problem of video understanding. In this situation, the conversation must be simultaneously parsed into conversation streams as it is undergoing discourse analysis. This has the potential to make the intentional discourse chunking significantly more complex.

Appendix, there appears to be a need for a perceptually-based model that bridges the gap between agent perception and an agent's mental model.

B.2 Collective intention

The SharedPlans model discussed in this section extends the plans as mental model approach to cooperative behavior. The foundation of the method rests on Bratman's[Bra92] three criteria required for "shared cooperative activity" (SCA): mutual responsiveness, commitment to the joint activity, and commitment to mutual support.

Bratman and others have concluded that single agent plans are not sufficient for modeling collaboration[Sea90, Bra92, GK96]. The argument is as follows: given two agents "collaborating," where each agent is committed to the act, one agent can prematurely abandon a joint goal as soon as it determines the goal is unachievable and it can drop the goal *without notifying the other agent*.

For example, assume that in the football example shown in Figure B-1a, player O_3 and O_1 are engaged in a collaborative act of neutralizing the threat from X_1. Their plan is that O_3 will block X_1 and O_1 will run by. The problem is that if O_1 and O_3 don't have commitment to mutual support, then O_3 can decide the goal of neutralizing X_1 is not possible and abort the plan without notifying O_1. O_1 will therefore probably get tackled by X_1. Bratman's restrictions, however, ensure that an agent in a collaborative activity won't drop the activity until it can notify the other agent that it is dropping the activity. The agents must arrive at mutual belief that the commitment is impossible[HL90]. A recognition system that knows *a priori* that an activity is a SCA might be able to use this constraint to predict what type of communication should take place between the two agents. The situation in the football example is complicated somewhat, however, by the fact that O_3 is communicating with O_1 *through the environment*. Therefore, by simply changing direction O_3 could be simultaneously aborting his goal and notifying O_1 that the goal has been aborted by assuming that O_1 visually perceives the change. This type of *visual* communication through the environment is common in everyday activity.

B.3 The SharedPlans mental model

The SharedPlans formalism is currently one of the most complete mental models of collaborative behavior, incorporating partial plans and action contracting. Grosz and Kraus adopt Bratman's SCA criteria. To have a collaborative plan for an action, agents must have:

- mutual belief of a partial recipe
- individual intentions that the actions be done and individual intentions that collaborators succeed in doing the identified constituent subactions
- individual or collaborative plans for the subactions

To achieve these constraints, the SharedPlans model formalizes several key properties:

- it uses individual intentions to establish commitment of collaborators to their joint activity
- it establishes an agent's commitments to its collaborating partners' abilities to carry out their individual actions that contribute to the joint activity
- it accounts for helpful behavior in the context of collaborative activity
- it covers contracting actions and distinguishes contracting from the collaboration
- the need for agents to communicate is derivative, not stipulated, and follows from the general commitment to the group activity
- the meshing of subplans is ensured; it is also derivative from more general constraints

For the details of the modal logic formalism, the reader is referred to the paper[GK96]. Here only a few of the operators are described, since the example that follows is intended to highlight issues related to collaborative action recognition, not to demonstrate the full power of the Grosz and Kraus theory.

The most fundamental modal operators are belief (`Bel`) and mutual-belief (`MB`). Three modal operators that relate agents and actions are `Exec`, the ability to perform basic level actions, `Commit`, a commitment to basic level actions, and `Do`, performance of an action. Plans ultimately reduce down to these primitives, but the primitives are still quite abstract for a recognition system based on perceptual observation of visual events. `Bel` and `MB`, for instance, can only be inferred by considering the context and how the agents or group of agents behavior maps to a set of known default or stereotypical behaviors. `Do` can be recovered by observation that an action has taken place, but because agents can `Do` an action inadvertently, `Commit` does not directly follow. If an agent is able to `Do` an act, though, `Exec` must have held true at the action instantiation time. Both the use of context and the knowledge of default models of behavior is critical for inferring these SharedPlan primitives from visual perceptual input.

The four SharedPlans intentional modal operators must also be inferred using context and behavior models: intention-to (`Int.To`), intention-that (`Int.Th`), potential-intention-to (`Pot.Int.To`), and potential-intention-that (`Pot.Int.Th`). Plans can be individual or shared and partial or complete. Shared plans are used to represent collaborative activity. `Int.To` is the basic level action, specifying that an agent is committed to an action and believes that it can perform the action; it is used in the definition of individual and shared plans. `Int.Th` applies only to shared plan situations where one agent does not require a complete recipe for sub actions of another agent but does require that subplans which are known must "mesh." For example, take the football example in Figure B-1. If O_2 and O_3 are blocking for O_1 and O_2 `Int.Th` O_3 will block X_1, O_2 doesn't need to know the recipe-for-action that O_3 will use; however, O_2 must satisfy Bratman's constraints on SCA with respect to O_3 blocking X_1. For example, O_2 won't knowingly conflict with any part of O_3's recipe that is known (say by getting in O_3's path) and O_2 believes O_3 will succeed.

Finally, `Pot.Int.To` and `Pot.Int.Th` are intentions an agent would like to adopt (by changing to `Int.To` or `Int.Th`) but that require deliberation using a reasoning system to ensure the intentions do not violate existing intentions and constraints. Using the same football example, if O_3 `Int.To` block X_1, then O_3 could also `Pot.Int.To` block X_2. However, once a constraint reasoning system determines that given current contextual information O_3 can't block X_1 and have time to block X_2, the `Pot.Int.To` would be dropped due to a pre-existing, conflicting intention.

The next section contains a partial application of the SharedPlans model to the football example. Again, for details of the formalism the reader is referred to the Grosz and Kraus paper[GK96]. The goal of this example is to give the reader a feel for the representation and to raise some recognition-based issues.

B.4 Applying SharedPlans to a football play

The example begins with some definitions. The temporal component of the definitions, which includes the time of the action and the time of the plan adoption, have been omitted for clarity.

- The offense (TEAM = $\{O_1, O_2, O_3\}$), the defense $\{X_1, X_2\}$)
- advance = action: advancing upfield
- play = action: the offensive play
- C_{play} = context for the play. Consists of an intentional component representing the intentional context in which TEAM is doing play,

  O_1: `Int.Th` (O_1, advance(TEAM, BALL), $C_{advanceO_1}$)
  O_2: `Int.Th` (O_2, advance(TEAM, BALL), $C_{advanceO_2}$)
  O_3: `Int.Th` (O_3, advance(TEAM, BALL), $C_{advanceO_3}$)

  plus other constraints on the performance of play. Note that $C_{advance}$ is another context which contains the constraints required for an advance action.

- blk_x1 = action: blocking X_1 player for O_1 (c_blk_x1 = constraints required to blk_x1).
- blk_x2 = action: blocking X_2 player for O_1 (c_blk_x2 = constraints required to blk_x2).
- rball = action: running ball downfield (c_rball = constraints required to rball).
- Recipe for the selected play, chosen by the coach before the play. The recipe calls for a block of X_1 at time 1, a block of X_2 at time 2, and a runball action during the entire play.

  R_{sp} = {{blk_x1(T_1, c_blk_x1), blk_x2(T_2, c_blk_x2), rball(T_{play}, c_rball)}, C_{Rsp}}

- C_{Rsp} = Context in which the recipe for the selected play (R_{sp}) is valid. Includes constraints and intentional context. For instance, a kickoff play recipe-for-action is only valid at certain times.

The SharedPlans formalism is versatile, dealing with full (complete) shared plans (FSPs), where all recipes are fully instantiated, and partial shared plans (PSPs). Cooperative planning that must deal with "hostile" agents like people, broken microwaves, defenders, etc. in a dynamic world requires partial shared plans. Otherwise, the agents will engage in so much mental state replanning that Bratman's goal of using intentional commitment to restrict computational complexity will be lost. In the football example, and probably in most realistic cooperative problems, nearly all plans will be partial. Most low-level actions cannot be planned until just prior to their execution. The tradeoff, of course, is that partial plans provide less constraint for predicting agent behavior based on intention and belief because much plan reasoning is left undone.

For this example, the assumption is made that a shared plan is established just before the play when the position of the defense is unknown, so the offense TEAM must have a partial shared plan for the play action, represented by:

$$\texttt{PSP}(\text{P, TEAM, play, C}_{play}) \tag{B.1}$$

This is the top-level representation of the plan for the entire offensive TEAM "play" action. The PSP is named P. As shown in the initial definitions above, C_{play} has extensive list of constraints and intentional motivations. The SharedPlans model propagates these constraints throughout the expansion of definitions, but it does not specify how a system might use these constraints to do means-ends analysis. In a situation such as the football play, the number of constraints at each step in the plan is large.

The SharedPlans formalism can be used to expand the definition of a partial shared plan. Expression B.1 breaks down into three conditions.

[PSP cond. 0] The first condition is that the TEAM has mutual belief (MB) that all members are committed to the success of the offensive play action. constr(C_{play}) are the constraints from the C_{play} context.

$$\texttt{MB}(\text{TEAM}, (\forall \text{G}_j \in \text{TEAM})\ (\texttt{Int.Th}\ (\text{G}_j, \texttt{Do}\ (\text{TEAM, play, constr}(\text{C}_{play})), \text{C}_{play}))$$

This mutual belief is established in the huddle before the play. This type of constraint is difficult for a recognition system to infer from visual evidence. To do so would require that so much visual evidence of a SCA be acquired by observing player behavior that the only *consistent* interpretation is that this condition must hold. Most likely a recognition system would be initialized with a default rule stating that the players are all committed to performing the play unless it perceives significant cues to the contrary.

[PSP cond. 1] The second constraint is that the TEAM mutually believe that there is a (partial) recipe-for-action for the play, R_{play}. Recipes(play) is the library of potential play recipes, designed by the coach.

$$\texttt{MB}(\text{TEAM}, (R_{play} \in \text{Recipes(play)})), \text{ where } R_{play} \text{ is partial.}$$

Each player's action depends both upon the selected recipe and the actions of the hostile defense agents; hence, some parts of the plan recipe are partial until more information is known. This condition only requires that team members mutually believe there is a recipe, not that they know each action in the recipe. Like PSP condition 0, this condition of mutual belief is established in the huddle before the play. A recognition system would need to assume that football players have an agreed upon recipe unless visual evidence suggests otherwise, since incontrovertible, direct evidence that this mutual belief is established is unlikely.[2]

This mutual belief in a partial recipe restriction is too strong in some situations, like when a play completely falls apart due to some unforeseen circumstance. In such a situation, individual team agents switch from a team-based recipe to individual recipes. These recipes may sometimes conflict, but for the most part if every agent acts reasonably locally, globally effective team-like behavior can result. The players are not explicitly collaborating with well-established mutual belief, but someone viewing the entire scene will often interpret the "emergent" collective behavior as collaboration.

[Expansion of PSP cond. 1] Since TEAM only has a partial recipe, the previous expression expands to the following:

$$\texttt{FSP}(P_{sr}, \text{TEAM}, \texttt{SelectRecGR}(\text{TEAM, play, [partial actions \& constraints } R_{play}]), R_{sr}, C_{sr/play})$$

In other words, there is a recipe (R_{sr}) for finding the full appropriate recipe for the play action given the partial recipe and current constraints. Further, the TEAM has a full-shared-plan (FSP) for finding the full recipe. The FSP is valid in the context of the recipe for selecting a recipe and the context of the play ($C_{sr/play}$). Note that the operator select recipe group (`SelectRecGR`) requires reasoning about complex constraints to generate a viable recipe extension consistent with the current context. The backtracking reasoning process used by

[2] Occasionally, a football team will fail to establish a recipe for a play due to time pressures. In these situations it sometimes appears to the spectator that the team is "confused." Viewers rely upon perceptual cues suggesting that especially uncoordinated behavior is occurring and therefore that the team has not agreed upon a plan. Usually some additional evidence like a quick huddle is required before viewers will feel comfortable with this interpretation, however. Often, unorganized behavior is caused by the defense agents, not lack of plan agreement.

such a procedure would be similar to the processes required by a recognition system that must perceive features and then reason about contextual constraints trying to construct feasible recipes that are most consistent with the current context.

Every subaction added to a partial recipe must be one that the TEAM mutually believes it can perform[3]. Each action can be brought about by an individual or a subgroup. For instance, in the football example there is mutual belief that blk_x1 can be brought about by a subgroup (GCBA) of the TEAM under the constraints from the context of the play constr(C_{play}) and the constraints from the blk_x1 action. Therefore, blk_x1 could be added to a partial recipe. It is not necessary to commit to which of $\{O_2, O_3\}$ will actually perform the eventual block, only that the two players can make the block happen. Similarly, there is mutual belief that action blk_x2 can be brought about by a subgroup of TEAM and therefore added to the partial recipe.

Finally, there is mutual belief that action rball can be brought about (CBA) by individual agent O_1. In this case, no matter what the defense does, the plan specifies that O_1 will perform the action. Of course, implicit in this belief is the assumption that O_2 and O_3 can do their actions unhindered by "hostile" agents. If that turns out not to be the case, replanning will be necessary.

$$
\begin{aligned}
&\text{constr}(C_{sr/play}) \supseteq \\
&\quad \texttt{MB}(\text{TEAM}, \texttt{GCBA}(\{O_2,O_3\}, \text{blk\_x1}, R_{blk\_x1}, \text{constr}(C_{play}) \cup \text{c\_blk\_x1})) \\
&\quad \texttt{MB}(\text{TEAM}, \texttt{GCBA}(\{O_2,O_3\}, \text{blk\_x2}, R_{blk\_x2}, \text{constr}(C_{play}) \cup \text{c\_blk\_x2})) \\
&\quad \texttt{MB}(\text{TEAM}, \texttt{CBA}(O_1, \text{rball}, R_{rball}, \text{constr}(C_{play}) \cup \text{c\_rball}))\ ]
\end{aligned}
$$

As suggested by the example so far, a recognition system that hopes to use a mental model of plans will need ways to infer mutual belief, where contextual constraint will play *the* critical role. For instance, in the SharedPlans model, resource allocation issues are bundled with the rest of the contextual constraints. Since agents don't need details on the recipes used by other agents for actions done by other agents, there is the possibility that the agent will not recognize some resource conflicts. This problem can't be entirely avoided unless agents compute all possible ways that resources could be used, which is computationally prohibitive, or agents communicate everything about what they are doing, which is also an unreasonable expectation. Visually, however, resource allocation is a robust contextual cue. If one object is being used by some agent, we often know for certain it is not being used by another distant agent. A hope is that the resource constraints could be encoded in the representation of collaborative activity in such a way so that when resource allocation can be determined

[3] Another possible option is that the TEAM believes it can contract out the subaction. Contracting is not used in this example, although it is a powerful feature of the SharedPlans formalization.

visually it can drive the recipe system to a restrictive set of interpretations.

The third and final condition of the original PSP, Expression B.1, applies for each subaction in R_{play}. The intend operators require that actions be broken into an action taxonomy of basic-level actions and complex actions (which are constructed of complex and basic-level actions). For many tasks, how such a taxonomy is constructed is an open question. Sometimes, for example, action definitions themselves seem to rely upon intentionality. Take a "point" action. How do we define this action based upon visual perception? Part of the definition could be observation of visual features characteristic of an extended human arm. The problem is that in random daily movement people make pointing-like gestures frequently. Context is required to robustly identify a pointing action. One constraint is that pointing generally requires an object of interest. People usually point *at* something. Still, it is possible to randomly make a "point" gesture toward an insignificant object. Hence, the observer agent must infer that the pointing agent *intended* to point using the context of the situation. A "fake" action in football, where a player moves one way to trick an opponent then moves the other way, is another "intentional" action. The visual features of a fake are ambiguous because they can happen in several types of situations. The observer agent must infer that the agent *intended* to fake in order to assign that action label.

The problem, then, with forming an action taxonomy and using mental models is that we need to recognize actions like point and fake to use SharedPlans to represent intentional relationships, but intentionality seems required to recognize these actions to begin with. The system needs a way to bootstrap itself from visually-basic, non-intentional actions like move-forward, turn-left, and contact-object. The action definition situation is further complicated by the difficulty of determining precise action boundaries in most realistic situations. For example, when observing a football play, when does the quarterback stop moving backwards and start throwing? In the football example in Figure B-1, exactly when does O_3 start "blocking?" When will other agents perceive that O_3 is blocking? Finally, how are the boundaries drawn between an individual action and a collaborative action? Is the act of the quarterback throwing the ball an individual act or a joint act or both, and when can this categorization be determined? These types of questions are indicative of fundamental issues that arise in recognition systems. These systems will need to decompose their visual input into useful primitives prior to performing much mental model plan reasoning.

Now, continuing with the last condition of the expansion of Expression B.1, the first the condition for the block_x1 action is shown below.

[PSP cond. 2 (block_x1 subaction)] Since at the time the play action is adopted it is unknown whether O_2 or O_3 will block_x1, it is a multi-agent partial subaction with subgroup $\{O_2, O_3\}$. This is a multi-agent plan, not an unreconciled plan, because it is agreed that one of the two agents will perform the action but it is not known which one until the defense can be surveyed. The two agents will need to coordinate with one another. The following conditions must hold.

[PSP cond. 2 part 1 (block_x1)] The subgroup has a SharedPlan (SP), named P_{blk_x1}, for the subaction. This condition will be expanded shortly.

$$\mathtt{SP}(\mathrm{P}_{blk\_x1}, \{\mathrm{O}_2, \mathrm{O}_3\}, \mathrm{blk\_x1}, \mathrm{C}_{blk\_x1/play}) \tag{B.2}$$

[PSP cond. 2 part 2 (block_x1)] The TEAM mutually believe that the subgroup has a SharedPlan to do the subaction. Note that the context includes C_{blk_x1} and C_{play} since context constraints are carried through all expansions.

$$\mathtt{MB}(\mathrm{TEAM}, \mathtt{SP}(\mathrm{P}_{blk\_x1}, \{\mathrm{O}_2, \mathrm{O}_3\}, \mathrm{blk\_x1}, \mathrm{C}_{blk\_x1/play}))$$

[PSP cond. 2 part 3 (block_x1)] The TEAM mutually believe the subgroup can bring about (CBAG) the subaction.

$$\mathtt{MB}(\mathrm{TEAM}, \mathtt{CBAG}(\{\mathrm{O}_2, \mathrm{O}_3\}, \mathrm{blk\_x}, \mathrm{R}_{blk\_x}, \mathrm{constr}(\mathrm{C}_{play}) \cup \mathrm{c\_blk\_x}))$$

This condition is strong, requiring all members of the TEAM to believe that the subgroup can bring about the action. Suppose, for instance, that some members of the team may actually be uncertain the group can bring about its action. As long as these skeptical members have recipes for action completely independent of the subgroup and there is no possibility that they will plan any action hindering the subgroup their belief will not affect the collaborative behavior. Such a break into independent subgroups might help simplify a recognition system by reducing the complexity of inter-agent intentional relationships that must be understood. If the player catching the ball and the player blocking are unlikely to influence each other's actions, there is no need to reason about their mutual belief.

[PSP cond. 2 part 4 (block_x1)] Finally, the TEAM mutually believe that all of its members are committed to the success of $\{O_2, O_3\}$.

$$\mathtt{MB}(\mathrm{TEAM}, (\forall \mathrm{G}_j \in \mathrm{TEAM})\ \mathtt{Int.Th}(\ \mathrm{G}_j, \mathtt{CBAG}(\{\mathrm{O}_2, \mathrm{O}_3\}, \mathrm{blk\_x1}, \mathrm{R}_{blk\_x1},$$
$$\mathrm{constr}(\mathrm{C}_{play}) \cup \mathrm{c\_blk\_x}), \mathrm{C}_{cbag/blk\_x1/play}))$$

If all agents have perfect perception, this condition prevents constraint violations since agents won't intend actions that create conflict if they are committed to the success of a subgroup. The problem is that perception isn't perfect, especially in a frenetic, dynamic world like a football play. Some agents may think they are committed to the subgroup action and not plan activities that interfere, but their planning is based upon perception. The agents can make errors which cause them to accidentally interfere with the subgroup. A recognition system, then, cannot assume that an action that looks like interference implies lack of commitment. In fact, given that football players are nearly always committed to the actions of subgroups of team members, especially strong contextual information should be required to justify such an interpretation. Inferring the level of commitment the TEAM has to all subgroup actions from visual data is difficult and sometimes impossible. The information would probably need to be explicitly provided to the system.

There is an condition for the block_x2 subaction which is omitted here because it is analogous to the PSP condition 2 case just shown. There is also another slightly different case for the rball single-agent action, outlined below.

[PSP cond. 2 part 1 (rball subaction)] O_1 will try to perform the rball action, a single-agent subaction, but due to defensive unknowns, O_1 only has a partial plan.

$$\texttt{Int.To}\,(\mathrm{O}_1, \mathrm{rball}, \mathrm{C}_{rball/play})$$

Further, TEAM mutually believe that O_1 intends to rball.

$$\texttt{MB}(\mathrm{TEAM}, \texttt{Int.To}(\mathrm{O}_1, \mathrm{rball}, \mathrm{C}_{rball/play}))$$

[PSP cond. 2 part 2 (rball)] TEAM mutually believe that O_1 can bring about the action.

$$\texttt{MB}(\mathrm{TEAM}, \texttt{CBA}(\mathrm{O}_1, \mathrm{rball}, \mathrm{R}_{rball}, \mathrm{constr}(\mathrm{C}_{play}) \cup \mathrm{C}_{rball} \cup \mathrm{c\_rball}))$$

[PSP cond. 2 part 3 (rball)] The TEAM mutually believe that all its members are committed to O_1's success. BLOCKERS = $\{O_2, O_3\}$.

$$\texttt{MB}(\mathrm{TEAM}, (\mathrm{G}_j \in \mathrm{BLOCKERS})\ \texttt{Int.Th}(\mathrm{G}_j, \texttt{CBA}(\mathrm{O}_1, \mathrm{rball}, \mathrm{R}_{rball}, \mathrm{constr}(\mathrm{C}_{play}) \cup \mathrm{pj}), \mathrm{C}_{cba/rball/play}))$$

A general problem with the SharedPlans formalism when studied from the perspective of recognition is its lack of a perceptually based model. It does not provide constraint on how agent's perception is used or how a system's perception limits inference ability. While it is true that a football team will establish some mutual belief and some partial recipe in the huddle, realistically once action begins, each agent will need to extend or modify recipes based upon the environment. It is impossible to plan for all circumstances in advance, and at some point agents need to adopt recipe changes on their own based only upon what they can perceive. The only confirmation agents have of other agent's current recipes are through perception and communication during the action. How then, is certainty of mutual belief tied to perception? This question is critical for two reasons. First, it can be used to infer when an agent will infer mutual belief. More importantly, if perceptual/contextual descriptions of "collaborative activity" can be outlined, a perceptual system can look for these key features to bootstrap a scene interpretation without having to reason about higher-level intentionality at the outset.

Continuing with the example, the SP (Expression B.2) from the PSP cond. 2 part 2 (block_x1 action) is expanded. This is an example of an unreconciled shared plan – the group does not know which of the agents will ultimately perform the subaction.

The group has a partial shared plan as well as a full shared plan to complete the partial shared plan.

$$\begin{aligned} &\exists\, P_{blk\_x2}, P_{blk\_x1\_Elab}, R_{blk\_x1\_Elab})\\ &\quad \mathtt{PSP}(P_{blk\_x1}, \text{BLOCKERS}, \text{blk\_x1}, C_{blk\_x1}) \wedge \\ &\quad \mathtt{FSP}(P_{blk\_x1\_Elab}, \text{BLOCKERS}, \mathtt{ElaborateGroup}(P_{blk\_x1}, \text{BLOCKERS}, \text{blk\_x1}, C_{blk\_x1}),\\ &\quad R_{blk\_x1\_Elab}, C_{blk\_x1\_Elab/blk\_x1})] \end{aligned} \tag{B.3}$$

Now, given $R_{blk\_x1} = \{\{$ hit_X1_sameside_O1 (T_1, c_hit_X1_sameside_O1)$\}$, $C_{R_{blk\_x1}}\}$, expand Expression B.3. BLOCKERS have mutual belief that all members are committed to the success of the group. The BLOCKERS believe that there is a recipe for blk_x1, but the recipe may be partial. They have a FSP to complete their partial recipe. For each subaction in R_{blk_x1}, the following must hold.

[Unreconciled case PSP expansion] BLOCKERS have not deliberated about the subaction; no decision has been made about which agent(s) will do it. BLOCKERS consider one of its members or a subgroup will do the action.

[Single agent subaction] The BLOCKERS mutually believe that there is a member of the group that can perform the action (CBA).

$$\begin{gathered} \mathtt{MB}(\text{BLOCKERS}, (\exists G_j \in \text{BLOCKERS})\mathtt{CBA}(G_j, \text{hit\_X1\_sameside\_O1},\\ R_{hit\_X1\_sameside\_O1}, \text{constr}(C_{blk\_x1}) \cup \text{c\_hit\_X1\_sameside\_O1})) \end{gathered}$$

The BLOCKERS mutually believe that all its members are considering being committed to the performance of the subaction of that agent.

$$\begin{aligned} &\mathtt{MB}(\text{BLOCKERS}, (\forall G_j \in \text{BLOCKERS})\ \mathtt{Pot.Int.Th}\,(G_j,\\ &\quad (\exists G_k \in \text{BLOCKERS}\ \mathtt{Do}(G_k, \text{hit\_X1\_sameside\_O1},\\ &\quad \text{constr}(\text{hit\_X1\_sameside\_O1}) \cup \text{c\_hit\_X1\_sameside\_O1}),\\ &\quad C_{do/hit\_X1\_sameside\_O1/blk\_x1})) \end{aligned}$$

Agents in the football example communicate during the play primarily through perception of their world. Once the action begins, agents rapidly act based upon the high-level goals established in the huddle and local behavior. When events don't go "according to planned" the agents fall back on individual routines which lead to "emergent" collaborative behavior. For example, if O_2 for some reason can't communicate with O_3 at some point, O_2 may decide to block X_2

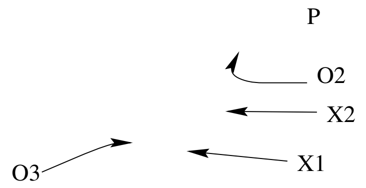


Figure B-2: Movement of several players during a portion of a football play.

using a rule that says "when in doubt, block the closest player." O_3 may then see O_2 heading towards X_1 and follow it's own individual rule saying, "block the closest player who doesn't appear to be blocked or about to be blocked." The resulting behavior will appear collaborative to an outside observer, although from the point of view of the agents, they have just followed a set of simple rules without explicitly agreeing to collaborate. Coordination was built into the action patterns of the players but not explicitly reasoned about during the play, and all communication occurred via observation of the environment by each agent.

If such action rules are shared between agents (i.e. one agent can predict what another agent perceives and which action rules the agent will use), then an agent has the capability to predict what another agent will do in a situation when there is no explicit prior collaborative plan.

B.5 The recognition task

Observation of visual features that are usually linked with collaborative behavior does not necessarily provide strong evidence that agents have established mutual belief of a SCA. Take the clip from a football play, shown in Figure B-2. Suppose two defenders, X_1 and X_2 are observed to be running toward one offensive player, O_3. The system needs to identify if the defenders are collaborating to double-team the offensive player O_3. Assume the system has no prior knowledge about the plans of the two defensive players. Given the situation as shown, one reasonable interpretation is that X_2 is covering O_2, O_2 just cut back, and X_2 has not yet turned in that direction. In this case, there is no collaborative double-team of O_3. However, another interpretation is that X_2 just happened to be near O_2 and is not covering O_2 but moving towards O_3. Without prior knowledge of player intentions, the recognition system has no way of knowing which interpretation correctly captures the intent of X_2. The intent is ambiguous.

One option for the system is to wait for more data. If the system observes that X_2 turns to follow O_2 then it can rule out a "double-coverage" interpretation. However, if X_2 continues to run in the direction of O_3 while X_1 also runs in that direction, the system still needs more information to confirm that X_2 and X_1 are engaged in a SCA. It could be that the play has completely fallen apart and both defensive players have reverted to a simple rule, "follow the defender you consider to be the greatest threat." If there is another defender at position P, then X_2 might move towards O_3 regardless of what X_1 does. This situation, however, will look identical to the case where there is a pre-established plan for X_1 and X_2 to jointly cover O_3.

If the system is forced to make an inference, it should select the interpretation that is the most *consistent* with known information. It might see that X_1 and X_2 are moving towards O_3 and find some evidence of a possible SCA, but if X_1 cuts off X_2 a more consistent interpretation is that there is no mutual belief of double coverage. Only if the system sees evidence of SCA and no evidence that an agent is acting atypically should it infer collaborative mental state information about X_1 and X_2.

In most domains (not just football) it is possible to construct individual rules for agent action that lead to behavior that will appear collaborative to an observer. Consider the cooking example described in Chapter 1. Imagine that two chefs are working in a community kitchen (e.g. a dorm). They don't know each other and don't communicate with one another directly at all while they cook. Further, they have not informed each other of their tasks and established any mutual belief. Assume they are both busy cooking and talking to friends, paying no attention to what each other do. They can both see the same counter-top, where all utensils are stored. Each cook begins making different recipes, and there are limited resources so they need to share equipment. Each cook can see when a utensil is available on the countertop and can take it. If a utensil is not available, the cook will work on some other part of his task until the desired utensil shows up. Or, if the cook has nothing else to do, the cook will just wait a few minutes. For the most part the cooks will naturally, albeit possibly slowly, work around resource conflicts. Now assume a visual recognition system is observing this scenario. Are the agents collaborating? The agents have not established any mutual belief (other than perhaps mutual belief in common courtesy), yet, to the recognition system the visual evidence suggests some task is getting completed in a collaborative fashion. Only if one agent gets stuck without a utensil for an long period of time with nothing else to do will the system be able to infer that collaboration is not taking place. In this case a sophisticated visual detector might notice (using some model of cooking behavior) that one agent is not cooking "optimally;" that is, the agent is doing nothing when the model of cooking and the current context suggest that the agent should be doing something. *Recognizing collaborative activity requires recognizing activity*; therefore, non-intentional models of action are needed to "bootstrap" the recognition of intentional action. These non-intentional action models might recognize *coordination*.

Computational models for intentional reasoning like the SharedPlans model were developed primarily with the goal of modeling high-level discourse between two human agents where conversants are purposefully providing intentional cues during dialogue [GS86]. Systems that recognize intentional action from video must typically do so without the intentional markers common in dialogue. Therefore, coordinated and collaborative behaviors will often be indistinguishable without prior knowledge of the SCAs. Usually that information is not available.

B.5.1 Summary

This case study of applying the SharedPlans model to a simple example from the football domain raised the following issues:

Model of expected communication If a system detects a collaborative key feature, how might it confirm that collaboration has occurred? One way is by having a model of expected communication given known collaboration. The SharedPlans formalism could be used to predict when and how communication is needed to maintain the potential SCA. For example, agents working together tend to become less certain of a collaborator's plan as time passes, increasing the likelihood of communication. Imagine the play in Figure B-2 represents players in a basketball game and X_1 and X_2 are thought to be engaged in the SCA of covering O_3. In this case, the recognition system knows that in order for one agent to drop the covering goal, it must notify the other agent either through speech or through gesture. One gesture might be arm movements. X_1 might decided to end the SCA by pointing at O_2 to indicate to X_2 that the joint coverage goal is over and that X_2 should go cover O_2. The SCA constraints explain why X_1 gestures to X_2 at all instead of simply abandoning the covering goal and moving to cover O_2 himself. Visual features detectors could look for evidence of this communication as conclusive support for the potential SCA. In this work, however, the trajectory data described in Chapter 3 is too coarse to recognize subtle communicative gesture such as hand waves and head glances.

Inter-agent communication Modeling intentionality in multi-agent domains from visual sensor input is difficult in part because agents sometimes explicitly communicate information about their plans. Two agents that are collaboratively interacting, for example, may communicate plan information verbally (as studied in the discourse community [GS86]) or visually. The problem is that to reason about visual communication requires that a system reason about the visual perceptual abilities of individual agents. For example, in the football domain, the QB may have the goal to throw the ball to a player, the LSE, who runs downfield ten yards. However, the QB will adjust his plan if he observes that the LSE has slipped and fallen after running three yards. Interpreting the QB's actions may require making assumptions that the QB observed that the LSE had fallen. Even from the original video signal, and especially from the data trajectories described in the next chapter and used in the work described here, these inferences can be difficult for a person, let alone a computer system, to make with high confidence.

Explanation of "why" vs. recognition of "what" The analysis of the shared plans formalism suggests that explaining how even an apparently "simple" multi-agent interaction was generated requires reasoning about the intentional processes and communications that occur between agents. However, reasoning about intentionality may not be required, in some cases, to recognize *what* has been observed. In Figure B-2, explaining why X_2 decides to cover O_3 requires reasoning about X_2's changing mental state and perceptual ability as well as the other options the agent considered and why each one was not chosen. However, the visual observation of X_2 position over time with respect to O_3 considered simultaneously with additional temporal and spatial contextual constraints, may permit an inference that covering has been observed with a relatively high likelihood without explicit intentional reasoning. Even though the processes by which two players collaborate in a football game can involve different degrees of "intentional" reasoning and communication, a recognition system that does not need to *explain* how the coordination developed may be able to model the visual situations in similar ways.

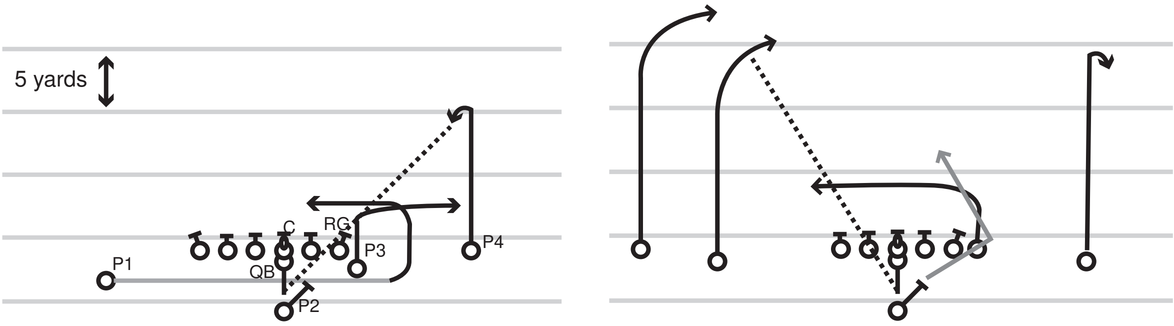
Agent-centered representation The shared-plans representation uses an agent-centered representation that allows modeling of situations in which an agent has an incorrect belief about another agent. In the football example, in order to understand what one agent does it is necessary use perceptual knowledge about the world *as that agent is thought to perceive it*. The recognition system described in Chapter 6 uses agent-based visual features to recognize collaborative activity.

Modeling collaborative "key features" A recognition system should have "collaborative key features" that indicate a high likelihood of "collaborative" activity directly from visual observation of agents and context. Such key features are small sets of other visual features that are highly unlikely to occur without some type of coordination. For example, in Figure B-1b, the system would detect a "potential-block" action between O_3 and X_1 and just a split second afterwards a "potential-cut" action by O_1 at nearly the exact same physical location. While this temporal and physical adjacency is not conclusive evidence that O_1 and O_3 are coordinating their actions, combined with other contextual information (e.g. temporal information connecting the behavior of different agents), it might lead to a reliable indicator of collaborative activity. These key features are likely to consist of temporal relationships between the actions of individual agents.

The observations made when applying the SharedPlans model to the football example in this Appendix have, in part, motivated the development of the multi-agent action representation presented in Chapter 6.

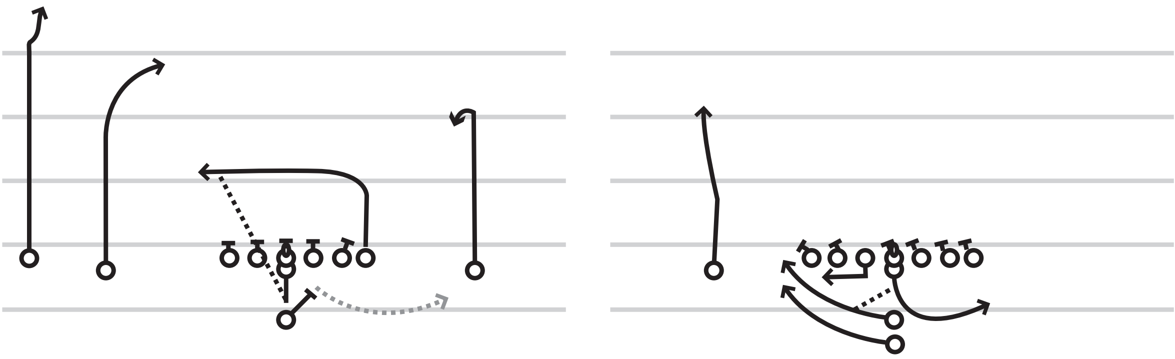
Appendix C

Play diagrams



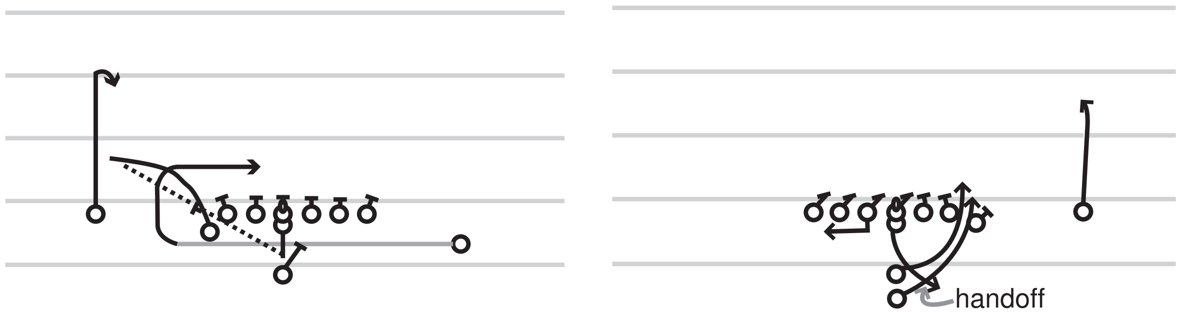
(a) (b)

Figure C-1: (a) p51curl and (b) p52maxpin play diagrams



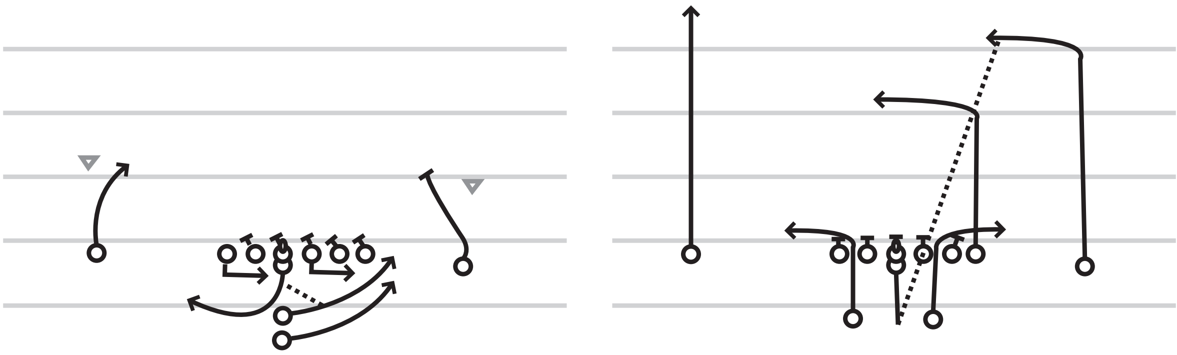
(a) (b)

Figure C-2: (a) p56yunder and (b) t39 play diagrams



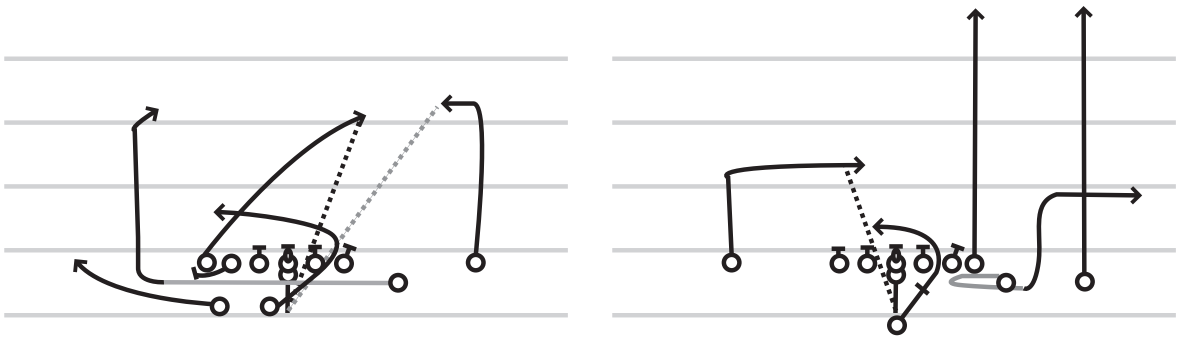
(a) (b)

Figure C-3: (a) p50curl and (b) r34 play diagrams



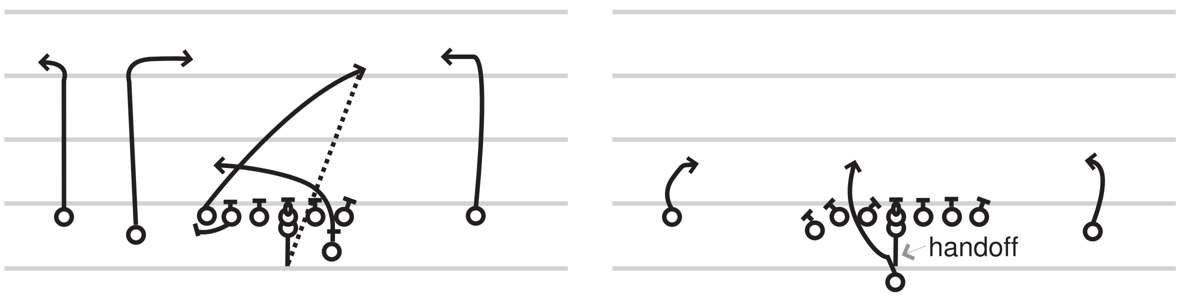
(a) (b)

Figure C-4: (a) t38 and (b) 143dig play diagrams



(a) (b)

Figure C-5: (a) p63up and (b) p54maxcross play diagrams



(a) (b)

Figure C-6: (a) p63upa and (b) s35 play diagrams

Appendix D

Context-labeling rule sets

```
(ISA (rectified-candidate ob-candidate db-candidate los-candidate
      isolated-blob-candidate multiple-blob-candidate ball-candidate
      person-candidate)
     label-candidate)

(ISA (offense-candidate defense-candidate)
     person-candidate)

(ISA clineman-candidate
     lineman-candidate)

(ISA (lineman-candidate back-candidate)
     offense-candidate)

(ISA (c-candidate guard-candidate tackle-candidate)
     clineman-candidate)

(ISA (lg-candidate rg-candidate)
     guard-candidate)

(ISA (lt-candidate rt-candidate)
     tackle-candidate)

(ISA (te-candidate se-candidate)
     lineman-candidate)

(ISA (fb-candidate hb-candidate tb-candidate
      wb-candidate sb-candidate fl-candidate qb-candidate)
     back-candidate)

(ISA (fs-candidate ss-candidate le-candidate re-candidate ldt-candidate
      rdt-candidate mlb-candidate lolb-candidate lilb-candidate
      rilb-candidate rolb-candidate)
     defense-candidate)
```

Figure D-1: Example object classification hierarchy used by the formation labeling algorithm.

This appendix contains supplementary examples of the hierarchy rules (Figure D-1), hypothesis generation rules (Figure D-2), ranking rules (Figure D-3), and consistency rules (Figure D-4). The algorithm using these rules is described in Chapter 4.

```
Candidate: lg-candidate
Constraints: Clique: los-candidate
             Object: ob-candidate
             Funcs:  (and (on-los OBJ los-candidate)
                          (flanked-both-para-los OBJ los-candidate
                                   ob-candidate))
Candidate: rg-candidate
Constraints: Clique: c-candidate los-candidate
             Object: ob-candidate
             Funcs:  (and (on-los-p OBJ los-candidate)
                          (same-side-los-p OBJ los-candidate
                                   ob-candidate)
                          (right-of-p OBJ c-candidate los-candidate))
Candidate: rg-candidate
Constraints: Clique: los-candidate
             Object: ob-candidate
             Funcs:  (and (on-los-p OBJ los-candidate)
                          (flanked-both-para-los OBJ los-candidate
                                   ob-candidate))
Candidate: rt-candidate
Constraints: Clique: rg-candidate los-candidate
             Object: ob-candidate
             Funcs:  (and (on-los-p OBJ los-candidate)
                          (flanked-left-p OBJ los-candidate
                                   rg-candidate))
Candidate: lt-candidate
Constraints: Clique: lg-candidate los-candidate
             Object: ob-candidate
             Funcs:  (and (on-los-p OBJ los-candidate)
                          (flanked-right-p OBJ los-candidate
                                   lg-candidate))
Candidate: te-candidate
Constraints: Clique: lt-candidate rt-candidate los-candidate
             Object: ob-candidate
             Funcs:  (and (on-los-p OBJ los-candidate)
                          (flanked-right-p OBJ los-candidate
                                   lt-candidate))
Candidate: te-candidate
Constraints: Clique: rt-candidate lt-candidate los-candidate
             Object: ob-candidate
             Funcs:  (and (on-los-p OBJ los-candidate)
                          (flanked-left-p OBJ los-candidate
                                   rt-candidate))
Candidate: qb-candidate
Constraints: Clique: c-candidate los-candidate
             Object: ob-candidate
             Funcs:  (and (behind-p OBJ c-candidate los-candidate)
                          (same-side-los-p OBJ los-candidate c-candidate)
                          (flanked-by-x-perp-p OBJ los-candidate
                                   ob-candidate))
Candidate: qb-candidate
Constraints: Clique: los-candidate
             Object: ob-candidate
             Funcs:  (qb-objs-los)
```

Figure D-2: More example generation rules.

```
; C: Prefer between QB and LOS
Candidate: c-candidate
Constraints: Clique: qb-candidate los-candidate
             Object: nil
             Funcs: (between-cs OBJ qb-candidate los-candidate)

; C: Prefer to left of RG
Candidate: c-candidate
Constraints: Clique: rg-candidate los-candidate
             Object: nil
             Funcs: (quantify (left-of-p OBJ rg-candidate los-candidate))

; C: Prefer to right of LG
Candidate: c-candidate
Constraints: Clique: lg-candidate los-candidate
             Object: nil
             Funcs: (quantify (right-of-p OBJ lg-candidate los-candidate))

; QB: Prefer QBs somewhere near center of mass of offense blobs
Candidate: qb-candidate
Constraints: Clique: nil
             Object: ob-candidate
             Funcs: (near-center-mass-cs OBJ ob-candidate)
```

Figure D-3: More example ranking rules.

```
; No duplicates can be entered for center linemen candidates
Candidate: clineman-candidate
Constraints: Clique: nil
             Object: nil
             Funcs: (no-duplicates-p)
; No duplicates can be entered for the LOS
Candidate: los-candidate
Constraints: Clique: nil
             Object: nil
             Funcs: (no-duplicates-p)
; Each low-level blob can have only one player name assignment
Candidate: offense-candidate defense-candidate
Constraints: Clique: nil
             Object: nil
             Funcs: (no-identical-blob-p)
; No offsides players permitted
Candidate: offense-candidate
Constraints: Clique: los-candidate defense-candidate
             Object: nil
             Funcs: (no-offsides-p)
; Linemen must be on LOS
Candidate: lineman-candidate
Constraints: Clique: los-candidate
             Object: nil
             Funcs: (on-los-p OBJ los-candidate)
; Linemen limited to 7 on LOS
Candidate: lineman-candidate
Constraints: Clique: los-candidate
             Object: nil
             Funcs: (six-or-less-on-line-p)
;C needs to be left of RT (wrt LOS)
Candidate: c-candidate
Constraints: Clique: rt-candidate los-candidate
             Object: nil
             Funcs: (left-of-p OBJ rt-candidate los-candidate)
;C needs to be lined up with FB (wrt LOS)
Candidate: c-candidate
Constraints: Clique: fb-candidate los-candidate
             Object: nil
             Funcs: (flanked-by-x-perp-p OBJ los-candidate
                                         fb-candidate :x 1)
;QB needs to be a reasonable distance from the LOS
Candidate: qb-candidate
Constraints: Clique: los-candidate
             Object: nil
             Funcs: (qb-reasonable-dist-los-p OBJ los-candidate)
```

Figure D-4: More consistency rule examples.

Appendix E

Complexity of Bayesian networks

This appendix briefly describes Bayesian networks and issues related to their computational complexity (for more detailed information, see [Jen96, Lau96, Cha91]).

E.1 Basic notation

Let G notate a *graph* representing a given world of interest, consisting of *vertices* (V) and *edges* (E). Let $V = \{S^1, S^2, ..., S^N\}$ represent an set of *nodes*. Let $e(S^i, S^j)$ represent an ordered, or *directed*, edge in G connecting *parent node* S^i to *child node* S^j. If all edges in G are directed, G is a *directed graph*. $e_u(S^i, S^j)$ indicates an *undirected* edge between S^i and S^j, which is equivalent to G containing both $e(S^i, S^j)$ and $e(S^j, S^i)$. If all edges in G are undirected, G is a *undirected graph*. If G contains both directed and undirected links, the G is a *mixed graph*.

A *path* exists for a given list of N nodes, $(S^1, S^2, ..., S^N)$ when an edge exists between each pair of nodes in the list, $e(S^i, S^{i+1})$ $1 <= i < N$. A path is a *cycle* when an edge exists between the first and last pair of nodes in the list, $e(S^N, S^1)$. When a directed graph, G, has no cycles, it is called a *directed acyclic graph*, or *DAG*. Nodes S^i and S^j are *adjacent* if the graph contains $e(S^i, S^j)$ or $e(S^j, S^i)$. A cycle is a *chordless cycle* if only the successive pairs of node in the cycle are adjacent. A graph is *triangulated* if and only if the only chordless cycles in the graph contain no more than three nodes. The *cliques* of a graph are the largest subgraphs of the graph that are complete, where a *complete graph* is one where there are edges between all pairs of nodes. A graph is *singly-connected* if only one path between any pair of nodes exits.

E.2 Probabilistic graph interpretation

A graph, G can represent a probabilistic distribution. Assume each node, S^i, corresponds to a random variable (or vector of random variables) in the world. Each S^i $(1 <= i <= N)$ consists of M states, $\{s^{i_1}, s^{i_2}, ..., s^{i_M}\}$, where s^{i_k} indicates the kth state of S^i. Each directed edge, $e(S^i, S^j)$ in G represents the conditional dependence of variable S^j on S^i. At each node S^i, with parent nodes $pa(S^i) = (S^j, S^k, ..., S^l)$, a conditional probability table is defined, $p(S^i|pa(S^i))$. For each node without parent nodes, a prior, $p(S^i)$, is specified.

The structure of G then defines a joint distribution of the world variables, denoted as $p(G) = (S^1, S^2, ..., S^N)$. The probability of a particular set of states is denoted by $p(w) = (s^1, s^2, ..., s^N)$.

Each node S models some random variable in the environment. A recognition system, however, must also consider *observations* of states. Let E^i represent the evidence for random variable S^i, where each E^i $(1 <= i <= N)$ consists of M states, $\{e^{i_1}, e^{i_2}, ..., e^{i_M}\}$, where e^{j_k} indicates the kth state of E^j. Although computationally S^i and E^i are treated identically, when analyzing the representational power of different graph structures for visual recognition it is useful to differentiate states of the world from observations of the world.

Examples of networks can be found in Section 5.2.[1]

E.3 Quantized and continuous states

In this work, node variable S^i is assumed to be discrete unless otherwise noted. Any continuous distribution (e.g. distances, angles, size measures) can be approximated with a finite number of discrete states at an additional (albeit sometimes intractable) computational cost.

Here the evidence nodes, E^i, are assumed to be discrete. As described in the main text of Chapter 5, evidence can be entered into a network by instantiating a particular state in the evidence node or by entering likelihood information into the node as "virtual evidence" [Pea88]. Continuous information can be entered into a discrete network using a fuzzy quantization. For example, if $E^i = \{e^1, e^2, e^3\}$, overlapping membership functions are defined for each state, e^i. These are typically Gaussian functions or piecewise linear functions. An observation is then compared against each function and a membership value for each state is returned. This memberships value defines a likelihood that is entered in the network as virtual evidence. Using this methods permits continuous observations to be entered into a discrete network. The discussion below remains unchanged regardless of which evidence instantiation method is used.

E.4 The complexity of exact solutions of graphical models

General algorithms exists for propagating the joint probability in any directed graphical model [Lau96, Jen96] and some mixed graphs [Whi90]. The most general algorithm for a DAG is commonly referred to as the *clique-tree* algorithm [Jen96]. Some graphical propagation algorithms (e.g. forwards-backwards algorithm for hidden Markov models [RJ93a]) have been demonstrated equivalent to the more general clique-tree approach [SHJ96]. The general algorithms are particularly useful because they permit the computational complexity of an arbitrarily-structured graphical model to be evaluated. The detailed discussion and proof of the fundamental concepts is not repeated here (see [Lau96]), but the procedure for determining complexity is briefly described.

Exact propagation in belief networks is NP-hard [Coo90]. Put most simply, the clique-tree algorithm transforms a graph such that all computation can computed locally in the graph and then propagated, without repetition, to other parts of the graph. The structure of the transformed graph sets a lower-bound on the complexity of the propagation. To

[1] In those graphs, unless otherwise indicated, a node representing a particular random variable is valid for all time. S^i_t represents the random variable node S^i at time t and s^i_{t-1} represents the state of S^i_{t-1} at time $t-1$. Hierarchical dependencies between random variables can be made explicit using the following notation. $S^{i_{j_k}}_t$ represents the node and random variable $S^{i_{j_k}}$ at time t, where S^{i_j} is a component of S^i and $S^{i_{j_k}}$ is a component of S^{i_j}.

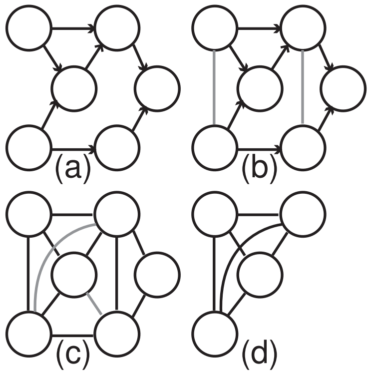


Figure E-1: (a) A directed, acyclic graph G, (b) the moralization of G, (c) the triangulation of the moralized G, and (d) largest clique.

determine the complexity of belief propagation in directed graph G, the graph is first *moralized* by connecting any unconnected parents of each node. For example, if $pa(S^i) = \{S^j, S^k, S^l\}$ and no edge exists between S^j and S^l then undirected edge $e_u(S^j, S^l)$ is added. An example graph and the moralized graph is shown in Figure E-1a and Figure E-1b, respectively, where the links added for moralization are shown in gray. Once all nodes have been moralized, the directed edges are changed to undirected edges. The new undirected graph, G_u, is now *triangulated* by adding by adding edges until the only chordless cycles that exist in the graph have only three nodes. Multiple triangulations of the same graph are possible, but triangulation should be performed so as to minimize the size of the largest clique in G_u.

Although this optimal triangulation is NP-hard, effective greedy heuristic procedures exist for graphs on the order of several hundred nodes[Jen96]. One greedy procedure is as follows. A node is S^i is *eliminated* by *filling-in* (i.e. adding) links so that all of the node's neighbors are pairwise connected and then removing S^i and all of its links. If S^i can be eliminated without adding links, then the node cannot be part of a chordless cycle with greater than three nodes. Further, a graph is triangulated if and only if all of its nodes can be eliminated one by one without any fill-ins [Jen96]. The greedy algorithm to minimize the size of the largest clique is to repeatedly eliminate a node that does not require fill-ins; alternatively, eliminate a node that yields the smallest conditional probability table connected nodes after its fill-ins [Jen96]. Figure E-1c shows the triangulated graph for Figure E-1b, where the links added for triangulation are shown in gray. The size of the largest clique in the moralized, triangulated graph sets a lower-bound on the complexity of belief propagation. Figure E-1d shows one of the largest cliques in the example, which has four nodes. The largest clique can be determined during the triangulation process.

Given the maximal clique size, the complexity of the graph can be approximated as

$O(NM^C)$, where N is the number of nodes [2], M is the maximum number of states in any node, and C is the number of nodes in the largest clique.

Approximate methods (e.g. Gibbs sampling) are available for solving large networks when exact methods are prohibitively expensive [GTS94]. Although some graph structures permit approximate solutions in polynomial time, approximate inference for general networks is NP-hard [DL93]. Fortunately, in practice, the exact solution methods are fast for small networks of around 100 nodes and for much larger singly-connected networks [RN95].

E.5 Relative importance of variable selection, structure, and priors

In addition to specifying the graph structure and the state values, representing a probabilistic distribution graphically requires that each node have a corresponding conditional probability table or prior distribution. Although for some domains these numbers can be estimated automatically from data sets using learning procedures [Hec95], for many real-world domains the knowledge engineers must empirically estimate the values. A common criticism of Bayesian methods is that these estimates are inaccurate and unjustifiably precise. However, several studies have found that it is the structure of a network, not the prior and conditional probabilities, that most affects its representational performance [HPF$^+$96, NA91]. Most well-designed networks exhibit a "flat maximum" property, where small changes to probability changes do not lead to change decisions unless those probabilities are near 0 or 1, which represent "absolute certainty" [NA91]. Further, simplifying from quaternary to binary node states has been shown to have little effect on network performance, suggesting that in some instances binary representations may provide adequate representational power [HPF$^+$96]. These observations – that the structure, not the numbers, provide the representational power – motivate the comparison of graphical representations found in Section 5.2.

E.6 Knowledge engineering

The structure of a Bayesian network controls the network's complexity. Although the structure is, in some cases, dictated by the domain, the knowledge engineer plays a large role in (1) determining which states of the world to represent, (2) determining the linking structure (i.e. which states are causally related to each other), and (3) specifying the prior and conditional probabilities. Knowledge engineers use modeling tricks (e.g. noisy-or and divorcing [Jen96]) to achieve efficient graphical structures. In practice, Bayesian networks are much easier to computationally solve than they are to construct!

[2] To be precise, S is not the number of nodes but the width of a structure called the *clique tree*. For the purposes of this paper, however, assuming N is the number of nodes is a reasonable approximation.

Appendix F

Visual network descriptions

This appendix includes brief descriptions of visual networks used by the recognition system, which are broken into generic action detectors that could easily be applied to other domains and football-specific detectors. The top-node for all the networks is binary.

F.1 Generic action detectors

Many of these networks integrate information from a small window of time. They are often used as evidence nodes in other networks.

stopped (obj) Look at a small window of time to determine if an obj is stopped.

running (obj) Look at a small window of time to determine if an obj is running.

movingBackwards (obj) Obj is moving in direction opposite of facing direction.

movingBackwardsReg (obj region) Obj is moving in direction opposite of shortest direction to region.

possession (obj1 obj2) Obj1 possesses obj2, where obj2 is a possessable object.

inContact (obj1 obj2) Obj1 is in contact with obj2.

stayingBetween (obj1 obj2 obj3) Obj1 is staying between obj2 and obj3, where obj3 is trying to get to obj2.

stayingBetweenClose (obj1 obj2 obj3) Obj1 is staying between obj2 and obj3, where obj3 is trying to get to obj2 and obj1 is either close to obj2 or obj2 is headed towards obj3.

changeMotionDir (obj dt1 dt2 dt3 dangle) Obj abruptly changes motion direction after heading in one direction for some time.

changingMotionDir (obj) Obj curling for some time.

changeMotionStop (obj) Obj abruptly changes motion direction and stops after heading in one direction for some time.

changeFacingDir (obj dt1 dt2 dt3 dangle) Obj abruptly changes motion direction after heading in one direction for some time.

runningStraight (obj distance) Obj is running fairly straight for at least distance.

runningStraightTime (obj frames) Obj is running fairly straight for at least frames.

runningAngled (obj distance angle referenceReg) Obj is running at approximately the given angle with respect the reference region for at least distance.

runningAngledTime (obj frames angle referenceReg) Obj is running at approximately the given angle with respect the reference region for at least frames.

runningCurved (obj distance) Obj is running in an approximately curved motion for at least distance.

runningCurvedTime (obj frames) Obj is running in an approximately curved motion for at least frames.

runningToward (obj1 obj2) Obj2 is running toward obj2.

runningAwayFrom (obj1 obj2) Obj2 is running away from obj2.

runningTowardReg (obj region) Obj is running toward region.

runningAwayFromReg (obj region) Obj is running away from region.

turnClockwise (obj) Obj is turning clockwise.

turnCounterClockwise (obj) Obj is turning counter clockwise.

turnTowards (obj1 obj2) Obj1 is turning toward obj2.

passBy (obj1 obj2) Obj1 is passing by obj2.

following (obj1 obj2 time) Obj1 is following obj2 for at least time.

F.2 Football-specific action detectors

Some of these detectors set special variable values when the detector reaches a threshold criteria for the first time. The threshold criteria is typically observing a certainty above some threshold for a specified amount of time. Many of these are actually called from within the networks in the next section.

ballCarrier (obj) Determine if in local time window, obj is the ballcarrier.

ballRunAcrossLine () The BALL object has been run, not thrown, across LOS in recent time window.

block (obj region) Obj is blocking near region.

blockForBC (obj) Obj is blocking for the ballcarrier.

BlockQBPass (obj) Obj is pass blocking for the QB.

catchPass (obj) Obj is trying to catch a pass.

cutDownField (obj) Obj is suddenly cutting downfield after running a man-in-motion pattern.

dropBack (obj d) Obj, who must be the QB, is dropping back d yards.

fakeHandoff (obj1 obj2) Over short time window, Obj1 is in a position to fake a handoff to obj2.

interceptDefender (obj) Obj is intercepting a nearby defender.

manInMotion (obj) Obj is running a man-in-motion movement.

pullLeft (obj) Obj is pulling left out of the formation.

pullRight (obj) Obj is pulling right out the formation.

receiveBallSnap (obj) Obj is receiving the snap.

receiveToss (obj) Obj1 is receiving a toss from obj2.

runBehind (obj1 obj2) Obj1 is running behind (with respect to the LOS) obj2.

runInFront (obj1 obj2) Obj1 is running in front (with respect to the LOS) obj2.

setInPosition (obj posReg) Obj is set, motionless, in posReg before the play begins waiting for the hike.

snapToQB (obj) Obj is snapping the ball to the QB.

snappingBall () The ball is Sets snapTime variable.

throwPass (obj) Obj is throwing the ball for a pass.

throwingBall () Sets throwTime variable.

playInProgress () The ball has been hiked and is still in play.

elligToRec (obj) Obj is eligible to receive the ball for a pass.

losBallCrossing () The ball is crossing the LOS. Sets losCrossTime variable.

passPatstartingBack (obj) Visual cues characteristic of pass patterns that start from the offensive back positions.

passPatStartingWides (obj) Visual cues characteristic of pass patterns that start from the offensive back positions for some sweeps.

passPatStartingEnd (obj) Visual cues characteristic of pass patterns that start from the tight end or WB positions.

passPatStreaking (obj distance angle inRegion toRegion distLOSatStart) Pass pattern segment where eligible receiver offensive obj runs approximately straight at least distance yards at an angle of approximately angle with respect to the losReg (where 0 is parallel and 90 is perpendicular). This occurs while in region inRegion and when the obj is headed towards region toRegion. Finally, distLOSatStart specifies the approximate distance from the losReg that the motion starts.

passPatStreakingThroughLine (obj distance distOut) Pass pattern segment where eligible receiver offensive obj runs straight through the offensive linemen shortly after snap for at least distance yards to approximately distOut yards from the losReg.

passPatParaLOS (obj distance inRegion towardsReg distLOSatStart) Pass pattern segment where eligible receiver offensive obj runs parallel to the losReg for at least distance yards in inRegion heading towards towardsReg and at approximately distLOSatStart yards from the LOS.

passPatParaLOSAway (obj distance inRegion towardsReg distLOSatStart awayReg) passPatParaLos where obj is moving away from from awayReg.

passPatParaLOSCenter (obj distance inRegion towardsReg distLOSatStart) Pass pattern segment where eligible receiver offensive obj runs parallel to the losReg for at least distance yards while in inRegion moving towards towardsReg at approximately distLOSatStart yards from the losReg at the start.

passPatCutting (obj angle towardsReg locationReg) Pass pattern segment where running eligible receiver offensive obj makes a sharp (e.g. about angle degrees) change in motion in direction heading to towardsReg around locationReg.

passPatCurling (obj towardsReg locationReg) Pass pattern segment where running eligible receiver offensive obj makes a gradual change in motion in direction heading to towardsReg around locationReg.

passPatCuttingStop (obj angle towardsReg locationReg) Pass pattern segment where running eligible receiver offensive obj performs passPatHooking and then stops abruptly.

passPatCurling (obj towardsRegionStart towardsRegionEnd locationReg distanceLOSatStart) Pass pattern segment where eligible receiver offensive obj runs towards towardsRegionStart and then curves towards towardsRegionEnd near locationReg and approximately distancLOSatStart yards from the losReg.

Appendix G

Temporal analysis heuristic functions

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

G.1 ObservedNow

Instantaneous value indicating certainty that some goal is being observed at the given time.

Assume:

- Current time, t
- Goal, g
- 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$(g)_t = \frac{S}{w_g}$

G.2 Observed

The maximum value for ObservedNow for frames $0 - t$.

- Observed$(g)_t = argmax($ObservedNow$(g)_0, \ldots,$ObservedNow$(g)_t)$

G.3 BeforeNow

Instantaneous value indicating certainty that some goal, $g1$ is observed as before some other goal, $g2$ at the given time.

- Current time, t
- Goal1, $g1$
- Goal2, $g2$
- Minimum before-gap-width, w = 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})

G.4 Before

The maximum value for BeforeNow for frames $0 - t$.

- BeforeNow$(g)_t = argmax($BeforeNow$(g)_0, \ldots,$BeforeNow$(g)_t)$

G.5 AroundNow

Instantaneous value indicating certainty that some goal, $g1$ is observed as around some other goal, $g2$ at the given time.

- Current time, t
- Goal1, $g1$
- Goal2, $g2$
- 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$(g)_t = \frac{S}{w}$

G.6 Around

The maximum value for AroundNow for frames $0 - t$.

- AroundNow$(g)_t = argmax($AroundNow$(g)_0, \ldots,$AroundNow$(g)_t)$

This work is dedicated to anyone who actually read this far...