A

Activity Recognition for Dynamic Multi-agent Teams

GITA SUKTHANKAR, University of Central Florida
KATIA SYCARA, Carnegie Mellon University

This article addresses the problem of activity recognition for dynamic, physically-embodied agent teams. We define team activity recognition as the process of identifying team behaviors from traces of agent positions over time; for many physical domains, military or athletic, coordinated team behaviors create distinctive spatio-temporal patterns that can be used to identify low-level action sequences. This article focuses on the novel problem of recovering *agent-to-team* assignments for complex team tasks where team composition, the mapping of agents into teams, changes over time. We suggest two methods for improving the computational efficiency of the multi-agent plan recognition process in these cases of changing team composition; our proposed approach is robust to sensor observation noise and errors in behavior classification.

Categories and Subject Descriptors: I.5 [**Pattern Recognition**]: Miscellaneous

General Terms: Algorithms, Experimentation, Performance

Additional Key Words and Phrases: activity recognition, plan recognition, multi-agent systems, teamwork

ACM Reference Format:
Sukthankar, G., and Sycara, K. 2012. Activity recognition for dynamic multi-agent teams. ACM Trans. Intell. Syst. Technol. V, N, Article A (January YYYY), 22 pages.
DOI = 10.1145/0000000.0000000 http://doi.acm.org/10.1145/0000000.0000000

1. INTRODUCTION

Proficient teams can accomplish goals that would not otherwise be achievable by groups of uncoordinated individuals. Often when a task is too complicated to be performed by an individual agent, it can be achieved through the coordinated efforts of a multi-agent team over a period of time. In real life, human teams can be found everywhere performing a wide variety of endeavors, ranging from the fun (sports, computer games) to the serious (work, military). Moreover, teams exist in the virtual world as well—in simulations, training environments, and multi-player games.

In this article, we address the problem of *plan recognition for multi-agent teams*, the process of inferring actions and goals of multiple agents from a sequence of observations and a plan library.[1] Although multiple frameworks have been developed for single-agent plan recognition, there has been less work on extending these frameworks to multi-agent scenarios. In the simplest case, where all of the agents are members of one team and executing a single team plan (e.g., players executing a single football play), plan recognition can be performed by concatenating individual agent observations and matching them against the team plan library [Intille and Bobick 1999]. However, this is not possible for many complex multi-agent scenarios that require agents to participate in *dynamic teams* where team membership changes over time [Tambe 1997]. In such scenarios, teams split into

[1]This article is an extended version of the AAAI 2008 conference paper "Hypothesis Pruning and Ranking for Large Plan Recognition Problems".

This research was supported by the following awards: AFOSR FA95500710039, AFOSR FA95500810356, and NSF IIS-0845159.
Author's addresses: G. Sukthankar, Department of EECS, University of Central Florida; K. Sycara, Robotics Institute, Carnegie Mellon University.

DOI 10.1145/0000000.0000000 http://doi.acm.org/10.1145/0000000.0000000

subteams to work in parallel, merge with other teams to tackle more demanding tasks, and disband when plans are completed.

Relaxing the assumption of static team assignment is desirable because it enables the analysis of more complex team tasks. However this makes the team behavior recognition problem substantially more difficult since behaviors are characterized by the aggregate motion of the entire team and cannot generally be determined by observing the movements of a single agent in isolation. For instance, it is relatively straightforward to recognize a group of agents moving into the "huddle" behavior once the other agents in the group are known, but it is difficult to recognize the same behavior given only the movement vector of a single agent, or those from all of the agents in the scene.

In this article, we describe two methods for reducing the number of candidate hypotheses a plan recognizer needs to consider when evaluating spatio-temporal traces of dynamic multi-agent teams. First, our algorithm, Simultaneous Team Assignment and Behavior Recognition (STABR) [Sukthankar 2007], recovers both team assignments and behavior annotations from traces of agent position over time. STABR leverages information from the spatial relationships of the team members to create sets of potential team assignments at selected time-steps. These spatial relationships are efficiently discovered using a randomized search technique, RANSAC [Fischler and Bolles 1981], to generate potential team assignment hypotheses. To prune the number of hypotheses, potential team assignments are fitted to a team behavior model; poorly-fitting hypotheses are eliminated before the dynamic programming phase.

The proposed approach is able to perform accurate team behavior recognition without an exhaustive search over the partition set of potential team assignments, as demonstrated on several scenarios of simulated military maneuvers. Second, if a plan library exists for the team's task, we utilize temporal dependencies encoded in the plan library in combination with agent resource requirements to further reduce the number of candidate plans considered in a search of the plan library. Specifically, we prune plans that violate team and temporal constraints before applying standard depth-first search techniques. By combining these two ideas, we can efficiently recognize plans, team groups, and behaviors from spatio-temporal traces of the agents' movement. In contrast to previous work, our approach is designed to handle the demands of dynamic teams, where team membership changes over time.

2. RELATED WORK

In his seminal work on plan recognition, Kautz defined his framework as a process for determining "which conclusions are absolutely justified on the basis of the observations, the recognizer's knowledge, and a number of explicit *closed-world* assumptions" [Allen et al. 1991]. In general, this union of observations, prior knowledge, and closed-world assumptions characterizes the research efforts on plan and activity recognition.

Much of the early work on plan recognition relied on logical methods, either viewing plan recognition as a specialized type of hypothetical [Charniak and McDermott 1985] or unsound reasoning [Allen 1983]. However, Kautz's event hierarchy framework [1987] combined deductive reasoning with a specific set of assumptions. Two important assumptions that he introduced are the exhaustiveness assumption and the minimum cardinality assumption. The exhaustiveness assumption specifies that the world is limited to the known types of events (plans), hence it is possible to determine that a particular event has taken place by eliminating all other possibilities. The minimum cardinality assumption follows the principle of parsimony, only assuming the minimum number of events that explain the observations. These two assumptions are either explicitly or implicitly made by most current plan recognition frameworks and are also important in our research.

By introducing probabilistic reasoning techniques, plan recognition becomes a process of determining which plans are *likely* rather than which conclusions are *justified*. Charniak and Goldman [1993] developed the first probabilistic model of plan recognition, using Bayesian belief nets for plan inference. Like Kautz's model, Charniak and Goldman's plan language relied on hierarchical action descriptions and did not support sequences of actions; later Huber et al [1994] demonstrated a

method for translating general acyclic plan specifications, including action sequence dependencies, into belief nets,

Kautz's plan recognition algorithm is exponential in the size of the knowledge base. By conceptualizing plan recognition as a variant of a context-free grammar parsing problem, some researchers [Vilain 1990; Lin and Goebel 1991] have developed approaches to reduce the complexity of the problem. Uniting probabilistic reasoning with a grammar parsing approach as was done in probabilistic state-dependent grammars (PSDGs) [Pynadath 1999] improves the expressiveness of the model while maintaining tractable inference. The PSDG model can be represented as a dynamic Bayesian network, which is a commonly used model for activity recognition. Goldman et al. [1999] proposed an alternate representation based on probabilistic Horn abduction specifically to address problems with partially-ordered and interleaved plans that arise when parsing is used for plan recognition.

Specialized search techniques have been developed to reduce the time required for plan recognition; for instance, RESC (Real-Time Situated Commitments) is a real-time approach to tracking the operator hierarchies of a Soar agent [Tambe and Rosenbloom 1995]. RESC uses information from the current world state to determine the validity of the operator hierarchies, commits to a a single interpretation, and backtracks in case of mistaken interpretation; this approach has the advantage that it can be used in real-time opponent modeling and handles reactive behaviors well [Tambe and Rosenbloom 1995]. Rather than committing to a single interpretation, the RESL algorithm marks every plan whose observations matches expectations and maintains it as a possible hypothesis [Kaminka and Tambe 2000]. Avrahami-Zilberbrand and Kaminka [2005] later developed a single-agent plan recognition algorithm which improves on RESL in several ways: (1) more efficient observation matching through the use of feature decision trees (FDTs); (2) the use of temporal structure to rule out inconsistent hypotheses for current state queries.

Multi-agent plan recognition has been developed for both athletic and military domains. To recognize athletic behaviors, researchers have exploited simple region-based [Intille and Bobick 1999] or distance-based [Riley and Veloso 2002] heuristics to build accurate, but domain-specific classifiers. For instance, based on the premise that all behaviors always occur on the same playing field with a known number of entities, it is often possible to divide the playing field into grids or typed regions (e.g., goal, scrimmage line) that can be used to classify player actions. Our algorithm does not rely on the presence of these external landmarks; however if such features exist, they can be incorporated into our framework both to reduce the number of team assignments considered and to potentially improve the behavior recognition accuracy. Previous work in this area typically assumes a known agent-team composition whereas our research focuses on behavior recognition for teams with dynamic composition. For the simple case of static agent-team composition, [Laviers et al. 2009] demonstrated a real-time football play recognition system using a set of support vector machine classifiers.

There has also been work on extending single-agent plan recognition frameworks [Tambe and Rosenbloom 1995; Bui 2003], both to create symbolic [Tambe 1996] and probabilistic [Saria and Mahadevan 2004] multi-agent plan recognition frameworks. These efforts have focused on the use of temporal behavior models and do not extensively utilize spatial information; such models have also been employed to detect teamwork failures [Kaminka and Tambe 2000] and agent-coordination termination [Saria and Mahadevan 2004]. [Banerjee et al. 2010] propose a branch-and-bound search process for multi-agent plan recognition, but the proposed technique does not account for noise in either the observations or the behavior recognition, in contrast to our approach.

Due to the difficulty of acquiring reliable location data for multiple entities, much of the research has been evaluated in simulation; however improvements in sensor technology such as the microwave position system described in [Beetz et al. 2005] should make real-world deployments possible in the future. Video recognition systems for analyzing 2-4 agent military teams have been demonstrated by [Hoover et al. 2005; White et al. 2009], but neither group examined the issues of scaling to large team sizes or utilizing plan libraries. The related problem of crowd analysis (with hundreds of people) has been explored by the computer vision community (e.g., [Ali and Shah

2008]), but such techniques focus on the flow of the aggregate rather than recognizing behavior of a subset of individuals. Scovanner and Tappen (2009) present an approach for learning motion models from video that leverages information about whether pedestrians belong to the same movement group. However this method simply identifies and tracks pedestrian groups without performing behavior classification or plan recognition. As tracking and object recognition techniques continue to improve, we expect that video analysis of large dynamic teams for applications such as surveillance or after-action reviews will be a compelling use of the proposed methods.

3. PROBLEM FORMULATION

We use the term **agent** to denote an independent entity capable of physical motion and action, such as humans, simulated entities in a virtual environment, or robots. A **team** is defined as "a set of agents having a shared objective and a shared mental state" [Cohen and Levesque 1990]. **Teams** and **subteams** are created when subsets of agents coordinate to independently pursue a separate objective; members of a subteam can rejoin their original team after finishing their objective or band with other agents to form different teams. Let $\mathcal{A} = \{a_0, a_1, \ldots, a_{N-1}\}$ be the set of agents in the scenario. We require that an agent only participate in one team at any given time; thus a **team assignment** is a set partition on $\mathcal{A}$. An agent that is not currently a member of any team is known as a **singleton**, and is unrestricted in its motion choices. Observations of the agents are in the form of **spatio-temporal traces**, a time series of the agents' physical positions.

From a complete set of observations over all the agents, we recognize the following team characteristics:

formation. a static layout describing the relative positions between the agents such as a column or a wing.

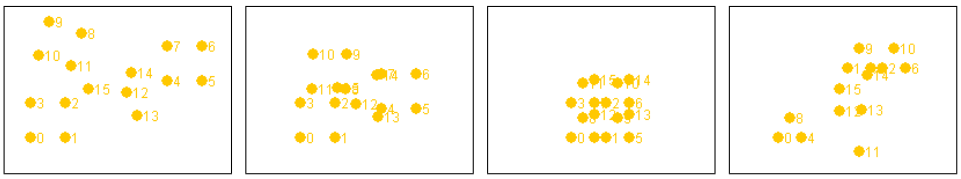
team behavior. a short, observable segment of coordinated movement and action executed by an agent team. Team behaviors are (1) non-atomic and (2) involve multiple agents. The agents in a team are constrained to move according to a set of **team behaviors**, $\mathcal{B}$. The subset of behaviors available to a given team is specified by the domain and can depend on the number of agents in the formation. Example team behaviors in our military domain include maneuvers such as wheel, pivot, buttonhook, stacked, and bounding overwatch.

plan. an ordered sequence of team behaviors describing a recipe used by an agent team to achieve a goal. During the course of a single plan, the original team can split or merge into different subteams to execute behaviors. Plans can be abandoned during execution, if necessary. An example military plan is building clearing, where the agent team moves through a map and neutralizes enemy combatants.

3.1. Dynamic Team Behaviors

In the course of a scenario, agents (either singletons or subsets of disbanding teams) can assemble into new teams; similarly, teams can disband to enable their members to form new teams or to operate as singletons. Thus the team assignment is expected to change over time during the course of a scenario. The team assignments over time and the behavior executed by each team are hidden from our system. We assume that our input consists only of a spatio-temporal trace, which is a sequence of noisy observations of the 2D position of each agent through time, $\mathbf{a}_i(t) \in \Re^2$. Our approach can be used with traces collected from a variety of position sensors (e.g., vision, handheld devices); the majority of our experiments were performed in a simulation of a physical environment where human subjects move avatars around a map. Although we do not have to contend with position error, there is a significant amount of variability in the manner that the human subjects execute behaviors and plans, so recognition remains a non-trivial problem.

We illustrate this with an example: Figure 1 shows several frames from a simple scenario with 16 agents. In Figure 1(a), 12 of the agents are arrayed in three teams of four agents in a square formation, $(\{a_0, \ldots, a_3\}, \{a_4, \ldots, a_7\}, \{a_8, \ldots, a_{11}\})$, with the remaining four agents as singletons. In Figure 1(b), the squares are converging towards the central area and the formations are starting to



(a) $t = 0$ (b) $t = 24$ (c) $t = 50$ (d) $t = 80$

Fig. 1. (a) An example scenario with three teams of 4 agents, $(\{a_0, \ldots, a_3\}, \{a_4, \ldots, a_7\}, \{a_8, \ldots, a_{11}\})$ and four singleton agents $(a_{12}, \ldots, a_{15})$; (b) teams maneuver while maintaining formation and converge to central area; (c) the three teams disband and regroup into four teams of 3 agents; (d) the various teams scatter as units. The interleaving of agent formations, the presence of singletons and observation noise (suppressed here) makes the team assignment and behavior recognition challenging.

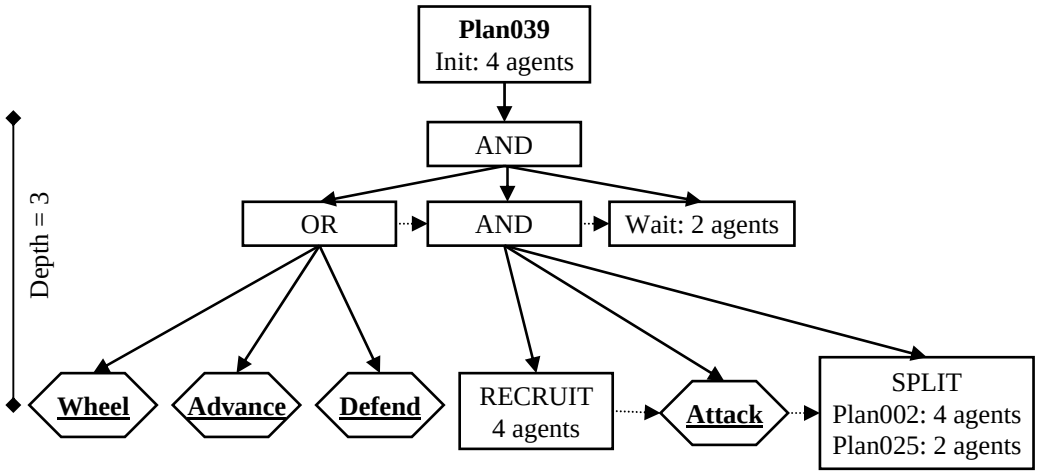


Fig. 2. Example plan tree. The top node lists the plan library index and the number of agents required (4) to start execution of this team plan. Hexagonal nodes denote directly observable behaviors; square nodes are effectively invisible to an external observer and must be indirectly inferred. The **RECRUIT** node indicates when additional agents are needed to continue plan execution. The **SPLIT** node denotes where the plan requires multiple subteams to execute subplans (002 and 025) in parallel. At the end of the plan, any remaining agents are released for recruitment by new plans.

interleave. In Figure 1(c), the squares are disbanding and those are regrouping into four groups of three, arrayed as triangles. Finally, in Figure 1(d), the triangles are moving away from the central area. For illustration purposes, observation noise is not shown in this figure.

3.2. Multi-agent Plans

In certain domains (e.g., military), the agents may be following a plan, an ordered sequence of team behaviors describing a recipe used by an agent team to achieve a goal. In these cases, we would like to be able to identify all the plans currently being executed by the team from a library, containing the set of possible plans. Since the team can split or merge into different subteams, our multi-agent plan representation needs to correctly model dependencies in parallel execution of plans with dynamic team membership.

Each plan is modeled as a separate AND/OR tree, with additional directed arcs that represent ordering constraints between internal tree nodes. Observable actions are represented as leaf nodes. These nodes are the only nodes permitted to have sequential self-cycles; no other cycles are permitted in the tree. Figure 2 shows a small example plan tree.

Additionally all plans are marked with an *agent resource requirement*, the number of agents required for the plan to commence execution (additional agents can be recruited during subsequent stages of a plan). For our military team planning domain, most leaf nodes represent observable multi-agent behaviors (e.g., movement in formation) and thus require multiple agents to execute. Note that the agent resource requirement specified in the top level node does not represent the maximum number of agents required to execute all branches of the plan, merely the number of agents required to *commence* plan execution.

We use two special node types, **SPLIT** and **RECRUIT**, to represent the splitting and merging of agent teams. A **SPLIT** node denotes that the following portion of the plan can be decomposed into parallel subtasks, each of which is handled by its own subteam. The node specifies the composition of each subteam and their tasks (which are simply plan trees). Any agents not allocated to a subteam will continue to execute the original plan until released. Merging teams are represented by **RECRUIT** nodes. **RECRUIT** nodes are a mechanism for teams to acquire more members to meet an agent resource requirement; if no agents can be found, plan execution blocks at the **RECRUIT** node until sufficient agents (released from other tasks) become available. **SPLIT** and **RECRUIT** are not directly observable actions and must be inferred from changing team sizes in observable leaf nodes. Since different observed actions can vary in duration, we do not assume strong synchronization across plans based on atomic action duration.

[Kaminka and Bowling 2002] developed the concept of *team coherence*, the ratio of total agents to the number of active plans, to represent the possibility of team coordination failures; they demonstrate that plan recognition can be used as part of a scalable disagreement-detection system to detect the existence of incoherent team plans. Here, we represent such teamwork failures as plan abandonment; if the agents reconcile their differences and resume coordination, it is detected as a new plan instance, rather than a continuation of a previous team plan.

4. APPROACH

In this section, we describe our techniques for identifying formations, behaviors, teams, and plans from spatio-temporal traces. Note that we are not proposing a new classifier for identifying spatio-temporal traces, but rather a framework for combining the results of multiple classifiers.

To completely understand the actions of the agents, we would like to recover the following information at every time step:

— an agent-to-team assignment $a_i(t) \mapsto \mathcal{S} \subset \mathcal{A}$,
— a team-to-behavior assignment $S_j(t) \mapsto b \in \mathcal{B}$,
— a behavior-to-plan assignment (assuming the existence of a plan library).

Our procedure is summarized as follows:

(1) Create an initial guess about agent-to-team assignment based on static formations.
(2) Use our proposed algorithm, STABR, to generate agent-to-team assignments and team-to-behavior assignments, based solely on an analysis of the spatio-temporal traces without the plan library.
(3) Use our pruning technique to rapidly eliminate plans that violate plan dependencies.

4.1. The STABR Algorithm

Ideally, one may wish to consider every legal agent-to-team assignment and team-to-behavior assignment at every time-step and then select the sequence that best matches the observed data. However, a straightforward implementation of this idea is computationally infeasible. The pool of potential agent-to-team assignments grows very quickly with the number of agents; this is equivalent to the number of partitions of a set, and is given by the *Bell number* of the set [Rota 1964]. The number of team assignments in the 16-agent example shown in Figure 1, $B_{16} > 10^{10}$. Clearly, examining every potential team assignment at even a single time-step is infeasible. And naively evaluating all

of the possible combinations of partitions over the entire spatio-temporal sequence further increases the complexity in an exponential manner.

Fortunately, a closer examination of the problem reveals structure that can be exploited to generate a computationally-feasible solution. The key observations behind our algorithm are summarized as follows. First, at each time-step, the relative positions of the agents in a team is constrained by the spatial configuration of the formation. Even though it may not be possible to unambiguously determine from a single time-step that an observed subset of agents is arrayed in a particular formation, one can profitably employ a static analysis of agent positions to generate hypotheses of valid team assignments and behaviors. Second, although an analysis of the motion of a single agent may not be sufficient to infer its behavior, an examination of the aggregate movement of several agents in isolation (i.e., a hypothesized team) generates significant information about team behavior. Third, by defining appropriate cost functions for the sequence, one can employ dynamic programming to dramatically reduce the time needed to find good sequences of team and behavior assignments through time. The next section details each of these ideas and describes how they contribute to the design of the STABR algorithm.

STABR analyzes spatio-temporal traces in three stages. First, it performs a static analysis of agent positions at each time-step to identify potential agent configurations that may correspond to known formations; these are used as an initial set of agent-to-team assignment hypotheses in later stages. STABR maintains multiple potentially-conflicting assignments for an agent, if there is spatial support. Second, STABR examines hypothesized team assignments in isolation and determines whether they have sufficient local spatio-temporal support. Pruning unlikely hypotheses at this stage is crucial since it greatly affects the performance of the last stage. This analysis also enables STABR to determine plausible behavior assignments for each of the surviving hypotheses. Third, these agent-to-team hypotheses are used to generate complete partitions over the agents. In the worst case, this state space could be exponential in the number of surviving hypotheses, underscoring the benefits of pruning. STABR then organizes the states (partitions) over the spatio-temporal sequence in the form of a lattice and employs dynamic programming to identify minimal cost solutions. These correspond to agent-to-team and team-to-behavior assignments that are a good fit to the observed sequence.

4.1.1. Static identification of agent formations. The first stage of the recognition process is to identify potential team assignments, based on static spatial cues. We do this by matching agent positions to pre-specified geometric formation templates; this enables the recovery of more complicated team relationships than the standard approach of clustering agents into teams based solely on proximity (see Results for a comparison of approaches). STABR employs a statistically-robust technique, Random Sampling and Consensus, RANSAC [Fischler and Bolles 1981], to automatically generate and test potential team assignment hypotheses at selected time-steps. For each formation template, agents are drawn uniformly and at random from both the template and the scenario. These point correspondences are used to generate a transform hypothesis to project the remaining template points into the scenario coordinate frame.

We define the set of legal transforms to be the class of similarity transforms (rotation, translation, and scaling); these can be parameterized in homogeneous coordinates as follows:

$$\mathbf{T} = \begin{bmatrix} s\cos(\theta) & s\sin(\theta) & x \\ -s\sin(\theta) & s\cos(\theta) & y \\ 0 & 0 & 1 \end{bmatrix}.$$

Then we apply the static formation recognition scheme described in [Sukthankar and Sycara 2006] to efficiently identify matching transforms; this method is summarized below. The randomly sampled minimal set of point correspondences is expressed in homogeneous coordinates as the 3×3 matrices $\mathbf{A}$ and $\mathbf{B}$ respectively. Since $\mathbf{B} = \mathbf{TA}$, we can recover $\mathbf{T}$ directly using matrix inversion. The match quality of the transform hypothesis $\mathbf{T}$ is assessed by projecting the coordinates of the remaining agents, as given by the template, into the scenario coordinate frame using $\mathbf{T}$. If the

predicted positions are sufficiently close to the observations, the template is accepted as valid and these agents are assigned to a team.

Since RANSAC stochastically searches the space of possible transforms it is not guaranteed to find the best match. However the following formula can be used to determine the number of iterations that are required to find the best match with a specified probability of success [Xu and Zhang 1996]:

$$m = \left\lceil \frac{\log(1-P)}{\log[1-(1-\epsilon)^s]} \right\rceil .$$

P is the target probability (e.g., $P = 0.99$ means the best match is found 99% of the time). s is the number of elements required to define the minimal set ($s = 2$ since a similarity transform requires 2 pairs of point correspondences). ϵ is the expected fraction of outliers in the data set. RANSAC is highly efficient at detecting templates that consist of many agents, since the number of iterations needed to achieve the desired probability is independent of team size. Detecting small teams in a scenario with many distracting agents is a harder problem since the other agents function as outliers. In the example scenario, where 75% of the points are effectively outliers for the square formation, a reliable detection of that template only requires 71 iterations, whereas an exhaustive search through the space would need ${}^{16}C_4 = 1820$ iterations.

4.1.2. Spatio-temporal Classification. There are some cases, particularly for smaller two-soldier subteams, in which static spatial configurations, by themselves, lack predictive power. To recognize these behaviors, our classifiers need to exploit the *temporal* information in behavior sequences in conjunction with spatial information on the position and velocities of team members. Since team behaviors can be executed in a variety of terrains, the classifiers must be robust to deviations in behavior execution caused by the team's response to local terrain features. However, arbitrarily introducing similarity transforms in the middle of a behavior sequence can destroy the spatio-temporal pattern created by the team's movements. To address the problem, we developed spatially-invariant classifiers by transforming our position data into a canonical reference frame defined by the team's motion, and applying a set of Hidden Markov Model classifiers to recognize three statically similar team behaviors (bounding overwatch, buttonhook entry, stacked movement).

4.1.3. Human Data Collection. To train and evaluate our spatio-temporal classifiers, we collected data from pairs of human players using our modified Unreal Tournament game interface to manipulate "bots" through a small urban layout while performing a particular sequence of team behaviors. Note that the subjects were not playing Unreal Tournament, but using Unreal Tournament to execute sequences of commonly used MOUT (Military Operations in Urban Terrain) team maneuvers. To directly monitor the performance of human players, we customized Unreal Tournament (UT) using the game development language *Unrealscript*. Many of the original UT game classes were written in *Unrealscript* and thus can be directly subclassed to produce modified versions of the game (known as mods); for example, Gamebots [Kaminka et al. 2002] is an example of a mod that allows external programs to control game characters using network sockets.

We developed our own `TrainingBot` mod that allows us to save the state of all the bots in the scenario; currently we save each player's ID number, position $(x, y.z)$, and rotation (θ,ϕ) every 0.15 seconds. This information is useful for both offline behavior analysis and for a separate replay mode that allows us to create bots that follow the paths recorded by the original players.

4.1.4. Canonical Representation. Due to the continuous nature of the domain, automatically determining the exact transition points between team behaviors is a difficult problem. While approaching and entering buildings, the players continue moving their bots, changing team behaviors as appropriate for the physical layout. We address this issue by dividing the traces into short, overlapping time windows during which we assume that a single behavior is dominant; these windows are classified independently as described in Section 4.1.5. To recognize team behaviors performed in different physical layouts, it is important for our classifier to be rotationally- and translationally-invariant; we

achieve this by transforming the data in each window into a canonical coordinate frame as described below.

More formally, we define:

— $a \in 1, \ldots, A$ is an index over A agents;
— j is an index over W overlapping windows;
— $t \in 1, \ldots, T$ is an index over the T frames in a given window;
— $\mathbf{x}_{a,j,t}$ is the vector containing the (x, y) position of agent a at frame t in window j.

The centroid of the positions of the agents in any given frame can be calculated as:

$$\mathbf{C}_{j,t} = \frac{1}{A} \sum_{\forall a} \mathbf{x}_{a,j,t}.$$

We describe the configuration of the agent team at any given time relative to this centroid to achieve translation invariance. However, rather than rotating each frame independently we define a shared canonical orientation for all of the frames in a window. This is important because it allows us to distinguish between similar formations moving in different directions (e.g., agents moving line abreast vs. single file). One standard technique for defining a canonical orientation is to use the principal axis of the data points for that window, which can be calculated using principal component analysis (PCA). However for efficiency we have empirically determined that we can achieve similar results by defining the canonical orientation as the displacement of the team centroid over the window: $\mathbf{d}_j = \mathbf{C}_{j,T} - \mathbf{C}_{j,1}$.

We rotate all of the data in each window so as to align its canonical orientation with the x-axis, using the rotation matrix $\mathbf{R}_j$. Thus the canonical coordinates, $\mathbf{x}'$, can be calculated as follows: $\mathbf{x}'_{a,j,t} \equiv \mathbf{R}_j \mathbf{x}_{a,j,t} - \mathbf{c}_{j,t}$. Our current recognition technique (described in Section 4.1.5) also relies on observations of agents' velocity as a feature which we locally compute as: $\mathbf{v}_{a,j,t} \equiv ||x'_{a,j,t+1} - x'_{a,j,t}||$.

4.1.5. HMM Classification. For each canonically-transformed window in our trace, our goal is to select the best behavior model. We perform this classification task by developing a set of hidden Markov models (HMMs), one for each behavior b, and selecting the model with the highest log-likelihood of generating the observed data. Our models ($\{\lambda_b\}$) are parameterized by the following:

— N, the number of hidden states for the behavior;
— $\mathbf{A} = \{a_{ij}\}$, the matrix of state transition probabilities, where $a_{ij} = Pr(q_{t+1} = j | q_t = i), \forall i, j$ and q_t denotes the state at frame t;
— $\mathbf{B} = \{b_i(o_t)\}$, where $b_i(o_t) = \mathcal{N}(\mu_i, \Sigma_i)$. The observation space is continuous and approximated by a single multivariate Gaussian distribution with mean, μ_i and a covariance matrix, Σ_i, for each state i;
— $\pi = \{\pi_i\}$, the initial state distribution.

For our problem, given A agents in a team, the observations at time t and window w are the tuple:

$$o_t = (\mathbf{x}'_{1,w,t}, v_{1,w,t}, \ldots, \mathbf{x}'_{A,w,t}, v_{A,w,t}).$$

We determine the structure for each behavior HMM based on our domain knowledge. For instance, the stacked behavior can be described using only two states ($N = 2$), whereas we represent the more complicated bounding overwatch behavior using six states connected in a ring. Each hidden state captures an idealized snapshot of the team formation at some point in time, where the observation tuple (in canonical coordinates) is well modeled by a single Gaussian. Rather than initializing the HMMs with random parameters, we use reasonable starting values. These can be polished using expectation-maximization (EM) [Duda et al. 2001] on labeled training data.

To determine the probability, $Pr(o_{1\ldots T} | \lambda_b)$, of generating the observed data with the model λ_b, we employ the forward algorithm [Rabiner 1989] as implemented in the Hidden Markov Model

toolbox [Murphy 2001]. We classify each window segment with the label of the model that generated the highest log-likelihood.

4.1.6. Spatio-temporal analysis of individual teams. The first stage of the algorithm identified, independently for each time-step, a set of hypothetical team assignments. The second stage identifies those team assignments that have significant temporal support, and generates behavior hypotheses for each such team that are consistent with the observed positions. The inability to find a plausible behavior to explain the motion of a hypothesized team is a strong indicator that the hypothesis does not correspond to a real team, but is rather a visual illusion caused by a coincidental configuration of agents.

The behavior recognition proceeds on a team-by-team basis. Each team is independently evaluated over the temporal intervals during which it was detected against the set of HMM classifiers. Thus, we iterate through each behavior and prune those behaviors that fail to match and (most importantly) prune those team hypotheses that cannot be explained by any legal behavior.

4.1.7. Explaining sequences of hypotheses. The final stage of STABR searches the space of team assignment and behavior recognition hypotheses generated by earlier stages for a consistent explanation over the entire spatio-temporal trace. In general, there may be several consistent explanations for the given observed agent movements; for instance, it is always possible to explain any trace as a coincidental convergence of uncoordinated singleton movement (though this would be highly improbable).

For every time slice, STABR generates a list of potential set partitions from the team assignment labels returned by the RANSAC template matching and validated by spatio-temporal behavior analysis. This list of set partitions represents a potential world state for that time slice; each world state contains a team assignment for every agent such that no agent is assigned to multiple teams. Generating a list of consistent world states is exponential in the number of team assignment hypotheses but is dramatically faster than considering the Bell number of total set partitions at that time step. Thus, effective pruning of team assignment hypotheses using spatio-temporal behavior analysis in earlier stages can greatly reduce running time. Any sequence through this set of partitions is both consistent (all agents are assigned to teams and no agent is assigned to multiple teams) and supported by local spatio-temporal evidence. To discriminate between sequences requires knowledge of the higher level plan; in absence of this information it is possible to use a cost function to select a solution that most parsimoniously explains the scenario.

4.2. Utilizing Implicit Temporal Dependencies

In this section, we discuss our technique for automatically recovering and utilizing hidden structure embedded in user-defined multi-agent plan libraries, assuming that a library exists. This hidden structure can be efficiently discovered when the plan library is created, indexed in tables that are stored and updated along with the plan library, and used as part of a pre-processing pruning step before invoking plan recognition to significantly reduce the number of plan libraries considered for each observation trace.

Traditional plan recognition would examine each trace T_i independently, and test each plan from the library $P_r \in \mathcal{P}$ against the trace to determine whether P_r can explain the observations in T_i. We propose uncovering the structure between related traces T_i and T_j to mutually constrain the set of plans that need to be considered for each trace.

Note that we cannot determine which traces are related simply by tracking the observed actions of a single agent through time as that agent may be involved in a series of unconnected team plans. However, by monitoring team agent memberships for traces T_i and T_j, we can hypothesize whether a subset of agents $\mathcal{A}_j$ from T_i could have left as a group to form T_j. In that case the candidate plans P_r and P_s for traces T_i and T_j, respectively, must be able to generate observations that explain

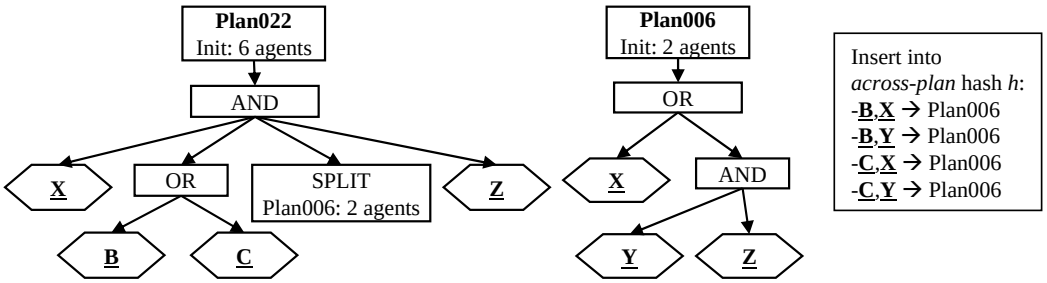


Fig. 3. Example of across-plan relationships captured by hash h. h captures observable behaviors across a team split. In this case, the **SPLIT** node in the parent plan (022) could be preceded by observation **B** or **C**, while the first step in the subplan (006) will generate either **X** or **Y**. Therefore, h will contain four entries, all pointing to Plan006. Observing one of these four sequences is an indication that the system should consider the possibility of a **SPLIT**.

both the final observation of $\mathcal{A}_j$ in T_i (not necessarily the final observation in T_i) and the initial observation of $\mathcal{A}_j$ in T_j.[2]

Similar temporal dependencies also exist between consecutive observations during a single execution trace. For instance, the observation sequence (B_p, B_q) can typically not be generated by every plan in the library, particularly if $|\mathcal{B}|$ is large or when plans exhibit distinctive behavior sequences. These dependencies are implicitly employed by typical plan recognition algorithms; our work generalizes this concept across related execution traces.

4.2.1. Plan Library Pruning. Our method exploits the implicit temporal dependencies between observations, across and within traces, to prune the plan library and to dramatically reduce the execution time of multi-agent plan recognition. Our algorithm for recovering hidden dependencies from the plan library proceeds as follows. First, we construct a hash, h that maps pairs of observations to sets of plans. Specifically, $h : B_p \times B_q \rightarrow \{P_j\}$ iff some parent plan P_i could emit observation B_p immediately before subteam formation and its subplan P_j could emit observation B_q immediately after execution. h can be efficiently constructed in a single traversal of the plan library prior to plan execution. Intuitively, h is needed because the formation of a subteam (i.e., **SPLIT**) is an invisible event; one can indirectly hypothesize the existence of a split only by noting changes in agent behavior. The presence of a **SPLIT** node can also be detected by observing a drop in team size in the parent trace. Specifically, h captures relationships between pairs of plans of the form that an observable behavior in the first plan can be followed by an observable behavior in the second plan (i.e., a subset of agents executing the first plan can **SPLIT** off to execute the second plan). Given a pair of observations, h enables us to identify the set of candidate plans that qualify as subplans for the identified parent plan. This allows us to significantly restrict the plan library for the child trace. Figure 3 illustrates the construction of h for a highly-simplified plan library consisting of two plan trees.

The temporal dependencies that exist between consecutive observations in a single execution trace can be exploited to further prune the set of potential plans. This is also implemented using a hash, g, that maps pairs of potentially-consecutive observations *within* a plan tree to sets of plans, which we also precompute using a single traversal of the plan library. Figure 4 illustrates a simple example with a plan library consisting of two plan trees. Some observable sequences could only have been legally generated by one of those two trees (e.g., **C**,**A**), while others are ambiguous (e.g., **A**,**B**).

[2]For this constraint to hold if plan abandonment is possible, we must assume that abandonment cannot occur *during* subteam formation—it either occurs before subteam formation or after the execution of the subteam's initial observed behavior.

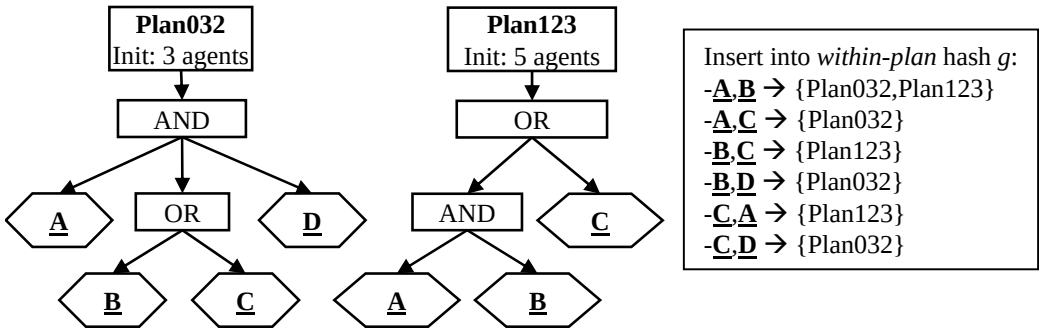


Fig. 4. Example of within-plan relationships captured by hash g. g captures all plans where two observations can be observed in sequence. For instance, the observed sequence of three agents executing **A**,**B** could be generated by either of the plan trees whereas **A**,**C** could only be the result of Plan032. These temporal constraints can significantly prune the set of possible plan hypotheses.

The size of these hash can be $O(|\mathcal{B}|^2|\mathcal{P}|)$ in the worst case since each entry could include the entire set of plans. In practice h and g are sparse both in entries and values. Applying h requires one lookup per execution trace while g requires a linear scan through the observations.

5. RESULTS

We evaluate our methods using several sets of experiments to evaluate the different aspects of our method: 1) formation recognition in simulated 2-D overhead maps of urban areas annotated with the location of MOUT entities; 2) behavior identification from activity traces of two-person human teams performing sequences of MOUT behaviors; 3) the use of STABR to recognize traces in the absence of a plan library; 4) the performance of plan recognition with a plan library. Since there is no single framework that handles all of these elements, we benchmark our methods against two commonly used baselines: 1) agglomerative clustering for identifying agent teams and 2) a depth-first search matching procedure for multi-agent plan recognition.

5.1. Team Template Matching

To test the robustness of our team template matching approach, we add clutter to the maps and distort formations by perturbing the positions of MOUT entities. Figure 5 reports the precision (fraction of correctly-classified results) and recall (fraction of formations that were detected) of our classifier under different conditions of clutter and location perturbation. Note that our approach independently matches each template against the data; thus, all matches that exceed the threshold score are reported as detections. The precision/recall curves are generated by varying this threshold parameter. There is no intrinsic restriction against assigning the same map entity to different templates — this enables us to create templates corresponding to a team and its component sub-teams, and to simultaneously recognize both.

In each experiment, we randomly place fifty MOUT formations on the urban map and report the precision and recall averaged over ten RANSAC searches. The left panel of Figure 5 shows the effects of adding clutter (spurious MOUT entities of the appropriate type) to the map, without increasing the number of RANSAC iterations. The percentage of clutter is measured against the total number of MOUT entities in the formations on the map (thus 100% clutter denotes a 1:1 ratio between spurious and desired MOUT entities). The results show that, as expected, the RANSAC-based approach is very resistant to the presence of spurious entities on the map, and that precision/recall are both very high even at the extreme clutter levels. The right panel of Figure 5 shows precision/recall results for experiments where the locations of each of the MOUT entities were perturbed with iid Gaussian noise. As expected, the performance degrades as noise is added since the spatial

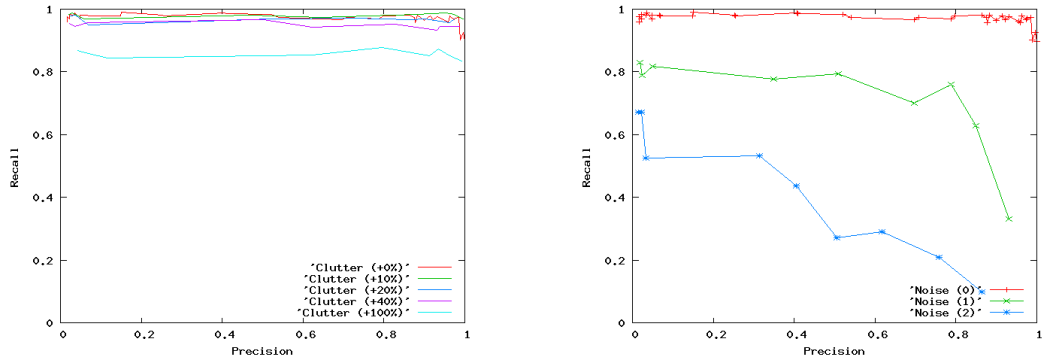


Fig. 5. Precision and recall curves for matching team templates using the RANSAC-based method. Fifty formations were placed on an urban map using randomized similarity transforms. Each run employed 100,000 iterations and the precision/recall averaged over 10 trials is shown. The left panel shows the effect of adding spurious MOUT entities (clutter) while maintaining the same number of iterations. The right panel shows the results of distorting observed formations by perturbing the location of each MOUT entity on the map with iid Gaussian noise.

configuration of the formation ceases to resemble the formation represented by the idealized model. However, we note that the technique is successful at identifying an acceptable number (80%) of the formations under reasonable noise conditions.

5.2. Spatially-Invariant HMMs

To evaluate our spatio-temporal classification method, we developed HMM models for three behaviors typically employed by two-person firing teams during the building clearing task: stacked movement, bounding overwatch, and buttonhook entry (see Figure 6). Note that these three behaviors look very similar in static snapshots and can only be robustly recognized by observing spatio-temporal traces. Position data was simultaneously recorded from two subjects at 0.15 second intervals using our `TrainingBot` mod (see Section 4.1.3). Players executed team behaviors in predesignated sequences, transitioning smoothly from one behavior into the next, adapting each behavior as needed to the local physical layout (turning corridors, entering rooms). The traces were divided into overlapping 20 frame (3 second) windows, which were transformed into a canonical coordinate frame as described in Section 5.2 (illustrated in the inset of Figure 6). The window size was empirically selected base on the observed average speed of the MOUT soldiers.

By using real data collected from human players rather than simulated traces, we can evaluate the robustness of our approach to realistic deviations during behavior execution. Figure 6 (right) shows a raw trace for each behavior; note that consecutive executions of the same behavior exhibit significant variation, particularly noticeable in the bounding overwatch behavior. Each behavior was also performed in a variety of local physical layouts. Table I presents the classification results (confusion matrix) for the three modeled behaviors; the accuracy of the HMM approach is good, particularly for the stacked formation. Buttonhook entry is sometimes confused with bounding overwatch, as may be expected from similarities in the canonical representation shown in Figure 6.

5.3. Recognizing Spatio-Temporal Traces

We evaluate the complete STABR algorithm on a set of scenarios of simulated military formations. The simulator generates traces for the position of each agent, corrupted with iid Gaussian observation noise and emits ground-truth data of the correct team assignments and behavior for the scenario. STABR processes this data and generates a team-assignment and a behavior for each agent, at every time-step. Our evaluation metrics are summarized below:

(1) **team assignment accuracy**: We score, at each time-step, whether the team assignment for each agent is correct. We employ a conservative metric and require the team memberships to match

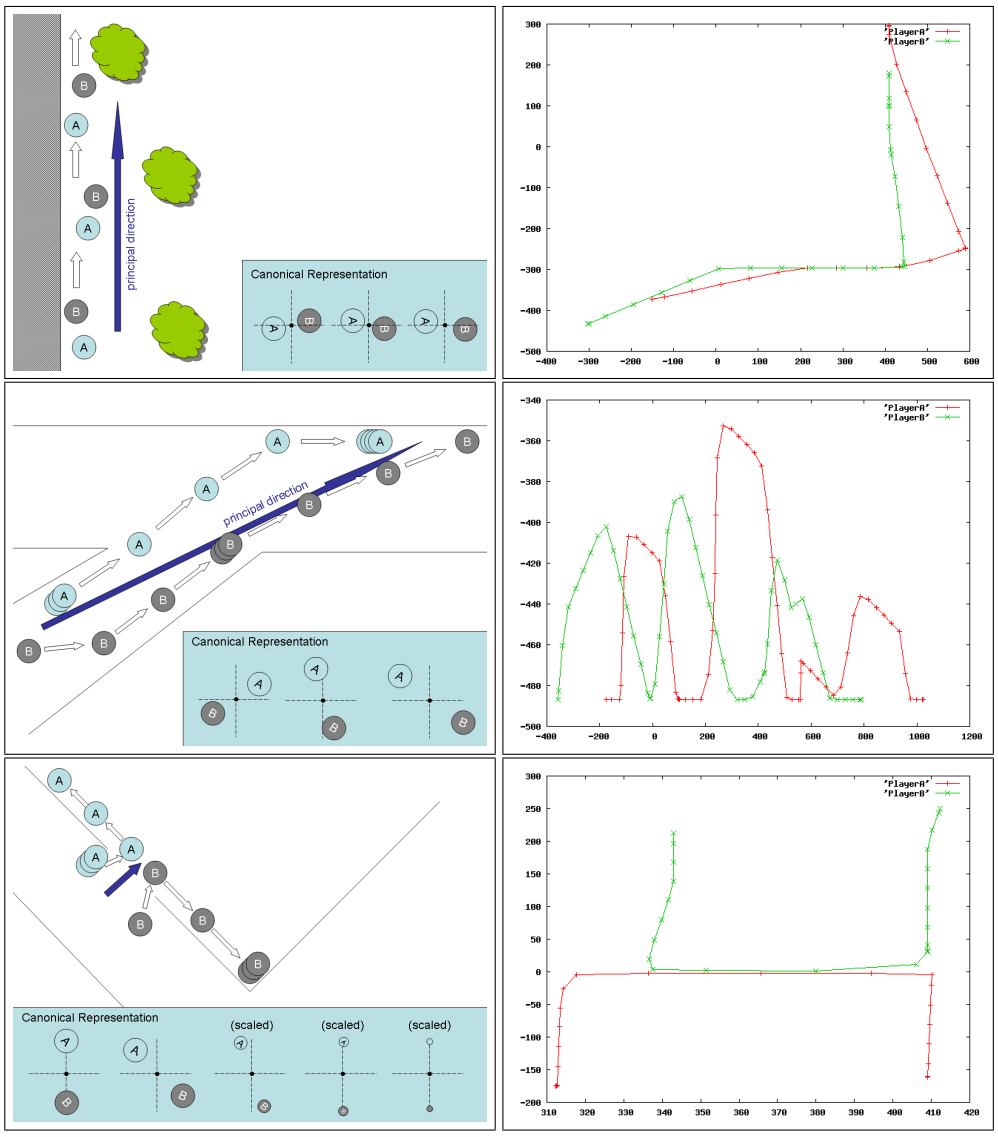


Fig. 6. Team Behaviors: Stacked Formation (top), Bounding Overwatch (middle), Buttonhook Entry (bottom). Schematics for each behavior, along with the canonical representation for several frames, are depicted in the left column. A sample raw trace for each behavior is shown in the right column; the coordinates of the axes are in Unreal Tournament length units.

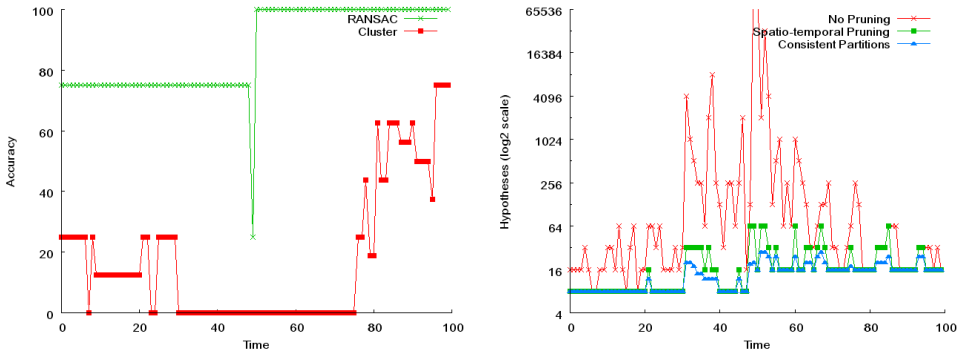
exactly; e.g., the absence of a single agent in a k-member team counts as k errors — one for each of the incorrectly-labeled agents — rather than a single assignment error. Team assignment accuracy is plotted over time (Figure 7(a)) to show results on a particular scenario and averaged over the scenario to generate aggregate results (Table II).

(2) **behavior recognition accuracy**: This measures the quality of behavior recognition and is computed in an analogous manner as team assignment accuracy, using the same conservative metric.

(3) **hypothesis set size**: We examine the number of hypotheses that are considered by STABR during various stages. This enables us to assess the contribution of spatio-temporal pruning.

Table I. Confusion matrix for HMM behavior classification. The ground truth is given in the left column; the classification result is given in the top row. The spatially-invariant Hidden Markov Model approach achieves good accuracy. Buttonhook entry is often confused with bounding overwatch, as may be expected from similarities in the canonical representation as shown in Figure 6.

	stacked	bounding	buttonhook
stacked	90%	10%	0%
bounding	14%	67%	19%
buttonhook	0%	33%	67%



(a) team assignment accuracy (b) #hypotheses

Fig. 7.

(a) Team assignment accuracy for STABR comparing agglomerative clustering with RANSAC on the scenario shown in Figure 1. Clearly proximity-based clustering is ineffective when agent formations are in close proximity. (b) Pruning team assignment hypotheses based on spatio-temporal behavior recognition drastically reduces the number of hypotheses that STABR considers. The number of hypotheses that remain after pruning closely follows the actual size of the consistent partitions.

Each of the following experiments examines a particular aspect of STABR to better understand its contributions.

The first experiment evaluates the benefits of employing the RANSAC-based formation template approach to identifying teams against a standard proximity-based clustering. K-means and agglomerative clustering are two popular unsupervised clustering methods [Duda et al. 2001] that are frequently employed to group agents into teams. Since the former requires that the number of clusters be externally-specified, we chose to compare STABR against the latter. In this experiment, the first stage of STABR is replaced with agglomerative clustering, where groups of proximal agents were aggregated into teams. Figure 7(a) presents the team assignment accuracy for both algorithms on the scenario shown in Figure 1. Agglomerative clustering and RANSAC both perform well when the agent teams are well-separated. However, as the formations begin to interleave, the accuracy of agglomerative clustering deteriorates rapidly. This is because agents that are proximal should frequently be assigned to different teams. The transient drop in accuracy near $t = 50$ corresponds to frames where 12 agents simultaneously transition from three groups of 4 agents to four groups of 3 agents over the span of a few frames; although either assignment would be correct during this interval, the ground truth file arbitrarily selects a single transition point, and STABR's explanation is marked as incorrect. Results on behavior recognition (not shown) mirror those for team assignment, since correctly identifying an agent's behavior generally requires the algorithm to also group it into the correct team.

Table II. Agent team assignment accuracy, averaged over all agents and time for a variety of scenarios. The benefits of STABR over proximity-based clustering are clearly evident. The scenario illustrated in Figure 1 is Scenario D.

	Baseline	STABR
Scenario A	95.8%	97.8%
Scenario B	57.0%	99.3%
Scenario C	36.0%	99.5%
Scenario D	18.3%	98.5%
Scenario E	0.0%	95.0%

Table III. Default plan generation parameters

Parameter	Default
Number of agents $\|\mathcal{A}\|$	100
Plan library size $\|\mathcal{L}\|$	20
Plan tree depth (average)	4
Plan tree branching factor (avg)	3
Number of observable behaviors $\|\mathcal{B}\|$	10
Parallel execution traces (average)	12

Table II summarizes the agent team assignment accuracy for STABR over several scenarios. While proximity-based clustering can handle the simplest scenario, it copes poorly with the interleaved formations in more complex scenarios.

The second experiment studies the contribution of the spatio-temporal behavior recognition, not in terms of accuracy but rather in terms of reducing the number of hypotheses from which world-states need to be generated. Since the execution time of STABR's last stage can grow exponentially with the size of this hypothesis set, it is important to reduce the set of team assignment hypotheses (without jeopardizing accuracy). Figure 7(b) shows (in semi-log scale) the size of the hypothesis set before and after spatio-temporal pruning along with the actual size of the consistent set (which is not actually known until stage 3). As can be seen, the spatio-temporal behavior recognition dramatically reduces the number of hypotheses that need to be considered by the third stage — without adversely affecting accuracy.

5.4. Multi-agent Plan Recognition

To evaluate the performance of our plan library pruning, we follow the experimental protocol prescribed by [Avrahami-Zilberbrand and Kaminka 2005], where simulated plan libraries of varying depths and complexity are randomly constructed. Randomly-generated plans do not reflect the distinctive structure of real-world plans and are therefore a pessimistic evaluation of our method since it relies so heavily on regularities between consecutive observations (both within and between plans). The plan trees are randomly assembled from **OR**, **AND**, **SPLIT**, **RECRUIT** nodes, and leaf (behavior) nodes. Adding a higher percentage of **SPLIT** nodes into the tree implicitly increases the number of execution traces since our simulator (described below) creates a new execution trace for each subplan generated by a **SPLIT**.

Given a plan library and a pool of agents, the execution trace generator simulates plan execution by allocating agents from the pool to plans as they commence execution and blocking plans at **RECRUIT** nodes while agent resource constraints remain unfulfilled. Note that a given plan tree can generate many node sequences; the same node sequence will execute differently based on which other plans are simultaneously drawing from the limited pool of agents.

To evaluate the efficacy of our method, we examine three pruning strategies over a range of conditions. The default settings for each parameter are shown in Table III. To reduce stochastic variation, the following graphs show results averaged over 100 experiments. All of the strategies employed the same depth-first search with backtracking to match execution traces against plan hypotheses.

On average, the across-trace (h) and within-trace (g) hashes are at 19% and 70% occupancy, respectively. The average number of plans hashed under each key is 1.14 and 2.87, respectively. The average wall-clock execution time for the default scenario, on a 3.6 GHz Intel Pentium 4, is only 0.14s, showing that multi-agent plan recognition for a group of 100 agents is feasible.

Since plan recognition methods can return multiple hypotheses for each trace, the natural metrics for accuracy are precision and recall. The former measures the fraction of correctly-identified traces over the number of returned results while the latter is the ratio between the number of correctly-identified traces to the total number of traces. Since all of the methods evaluated here are complete, it is unremarkable that they achieve perfect recall on all of our experiments. Precision drops only when multiple plan trees match the observed trace. In these experiments, precision was near-perfect for all methods, indicating that there was little ambiguity in the generated traces. In a small number of cases (where the observable action vocabulary was small), our method achieved higher precision than the baseline because it was able to disambiguate otherwise identical traces based on parent-child dependencies. However, we do not claim better precision in general over baseline methods since these cases are infrequent; rather, the primary focus of this article is to present a more efficient scheme for team plan recognition that exploits inter-plan constraints.

We perform a set of experiments to evaluate the efficiency of three approaches to team plan recognition:

Baseline (Unpruned). depth-first matching of the observation trace against each plan in the library.
Team Only. prune plan libraries for each observation trace using across-trace dependencies from h before running depth-first matching.
Team+Temporal. prune plan libraries using both within-trace dependencies stored in g, and across-trace dependencies from h, before running depth-first matching.

Figure 8(a) shows how plan recognition time (as measured by the number of leaf node comparisons) scales with growth in library size (number of plan trees). We see that the Unpruned and Team Only approaches scale approximately linearly with library size while the cost for combined Team+Temporal pruning remains almost constant. This is because the set of plan trees that could explain a given *set* of observed traces remains small.

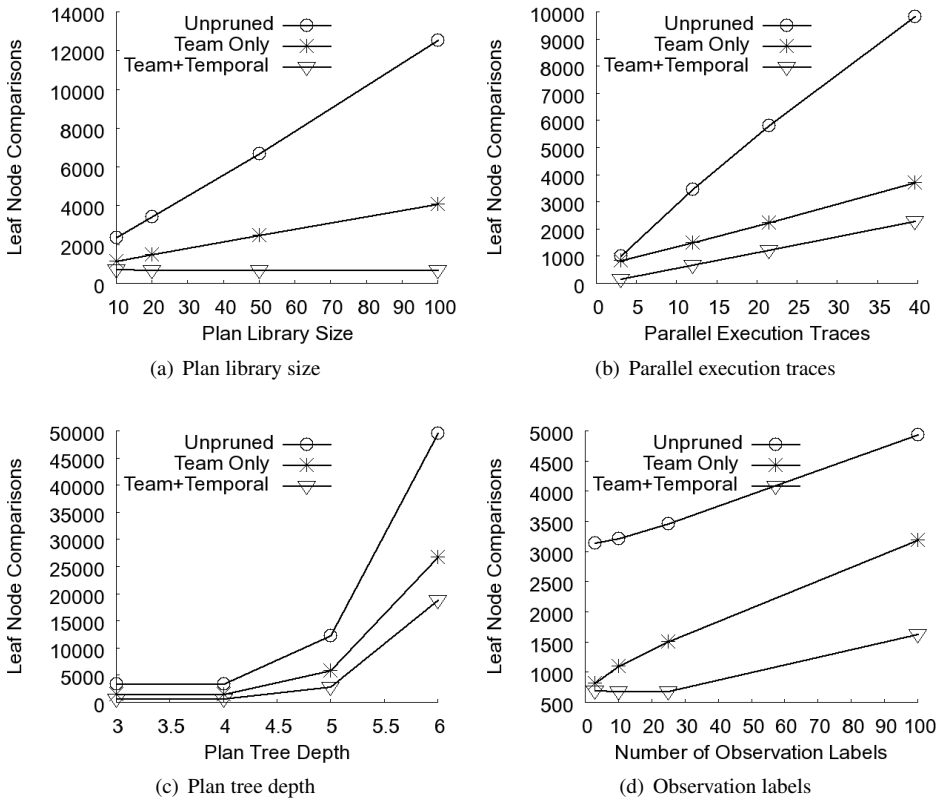
Figure 8(b) examines how the performance of the three methods scales with the number of observed execution traces. It is unsurprising that the time for all of the methods grows linearly. However, pruning significantly reduces cost. In this case, Team+Temporal achieves a consistent but less impressive improvement over Team Only. We see that the pruning strategies enable us to run plan recognition on much larger scenarios.

Figure 8(c) presents the cost of plan recognition against the average depth size of plan trees in the library. Since the number of nodes in a plan tree increases exponentially with depth, we expect to see a similar curve for each of the three approaches. However, we do see a dramatic reduction in cost due to pruning.

Figure 8(d) shows how increasing the number of distinctly-recognizable low-level behaviors (number of observation labels) impacts the cost of team plan recognition. As the number of potential labels grows, it becomes easier to disambiguate sequences of observed actions. Consequently, the benefits of pruning within-trace (using hash g) become increasingly important. This is evident in our results, where Team+Temporal pruning shows clear benefits.

5.5. Robustness to Observation Noise

In some simulation environments, one can collect highly-accurate low-level behavior traces from multiple agents and humans acting in the virtual world. However most real-world activity recognition systems that monitor the activity of humans using cameras [Nguyen et al. 2005], GPS readings [Liao et al. 2004], or wireless signal strength measurements [Yin et al. 2004], report error rates ranging from 5%–40% in accurately classifying behaviors from position data. These error rates pose a challenge for our algorithm since we rely on the existence of temporal dependencies between be-



(a) Plan library size (b) Parallel execution traces

(c) Plan tree depth (d) Observation labels

Fig. 8. Cost of plan recognition, as measured by leaf node comparisons, for different pruning strategies under varying conditions: (a) size of plan library; (b) average number of plans executing in parallel; (c) average depth of plan tree; (d) number of observable behaviors. Pruning using h and g enables dramatic improvements for large plan libraries.

havior observations, across and within traces, to prune the plan library. If these dependencies were corrupted by observation noise, then the pruning algorithm as described above could incorrectly prune correct plans because the noisy observation traces might contain observed transitions that would be "illegal" according to the correct plan. On the other hand, observation failures resulting in fewer behavior transitions being recorded would not adversely affect pruning accuracy since the absence of transitions cannot trigger the deletion of a plan from the hypothesis set.

To address this challenge, we extend our approach by shifting the focus from *pruning* to *prioritization*. Rather than eliminating from consideration those plans that could not legally generate the observed behavior transitions, we order plans based on their likelihood of generating the observed sequences. This likelihood is estimated according to the same criteria employed for pruning—temporal dependencies between observations, both within and across traces. We pre-process the plan library in the same manner, to construct the hashes g (within-trace constraints) and h (across-trace constraints). However, these hashes are employed in a different manner against the observed data. For pruning, the hashes were used to delete plans from the hypothesis set; here they are used to augment the likelihoods of plans that are consistent with the given observation. By assuming conditional independence of observed transitions, we can approximate the log-likelihood of matching a given observation to a particular plan as the sum of independent contributions from each transition. In the absence of additional information from the low-level recognizer, we can treat these contributions as equal. This leads to the following approach for plan ordering. For each observed trace, we accumulate a score that is a linear combination of contributions from observations that are consis-

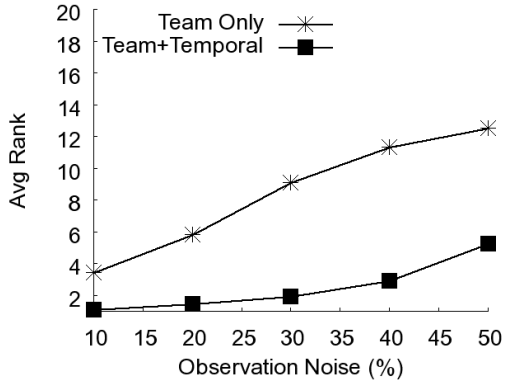


Fig. 9. Average rank of correct plan in conditions of increasing observation noise. The prioritization scheme is effective at ordering plans such that the correct one is within the top 10%.

tent with g and h. The plan library is sorted according to this score (this ordering is specific to each trace), and the behavior recognizer is applied to the plans from most promising to least promising until a suitable match is found.

As with the pruning method, the prioritization approach is agnostic to the choice of behavior recognizer. Although all of the plans in the library can be sent to the recognizer for detailed analysis, in practice we apply the recognizer only to the most promising plans (i.e., the top 10%).

To evaluate the efficacy of our prioritization method, we examine the robustness of the ranking with respect to observation noise. These experiments were conducted with a library with 100 plans (average depth 4). The observation traces were generated as above and then corrupted by iid noise (conditions ranging from 0% to 50% probability of misidentification). A corrupted observation was replaced by a random observation drawn with uniform probability from the set of 10 observable actions.

The observed transitions were used to generate likelihood estimates for each of the 100 plans. The rank of the correct plan (known from ground truth) serves as a measure of the quality of the prioritization. Ideally, one would like the correct plan to be at rank 1; in practice, we would be satisfied if the correct plan appears reliably in the top 10%, since this gives us an order of magnitude improvement over a brute-force matching approach.

Figure 9 summarizes the average results from 100 independent trials for prioritization over a range of noise conditions. We make several observations. First, we note that the prioritization is very effective at scoring the correct plan within the first few ranks (average rank is only 5.2 out of 100 even in extremely noisy conditions). The standard deviations for these results ranged from 1.2 (for 10% noise) to 12.4 (for 50% noise). Thus, in moderately noisy conditions, it is reasonable to expect that the correct plan will fall within the top 10%. Second, we can see that although the across-team constraints alone are fairly effective at ordering the plan library, one can achieve significant improvements by also incorporating within-trace information. This is particularly valuable in high-noise conditions where the chance of corrupting a key observation spanning sub-team formation is non-negligible. Finally, we note that these experiments exploited no additional domain knowledge, such as better sensor models (e.g., confusion matrices for which observations are likely to appear similar) nor indications about which observations might be outliers based on higher-level plan knowledge. These additional sources of domain information can complement our prioritization strategy and further improve performance. This validates our belief that a prioritization-based strategy could significantly improve the efficiency of multi-agent team behavior recognition.

6. DISCUSSION

In military scenarios, group assignment can be quite challenging because modern forces often split into multiple disconnected parts (e.g., far-ranging scouts, small diversion groups, and flanking elements). Recognizing what the force is doing is often possible once it is clear which units are involved. STABR correctly recovers team assignments even in cases of non-spatially contiguous divisions that foil standard clustering approaches to team assignment.

The STABR approach is based on the following intuitions:

(1) Initial agent-to-team assignments can be made on the basis of static spatial cues.
(2) The aggregate agent movement for an incorrect team assignment will generally fail to match any behavior model; this can be exploited to prune poor team assignments thus speeding computation.

The scenarios presented in this article illustrate the operation of STABR in environments that lack the external cues used by other multi-agent plan recognition approaches, such as landmarks, cleanly-clustered agent teams, and extensive domain knowledge. We believe that when such cues are available they can be directly incorporated into STABR, both to improve accuracy and to prune hypotheses. STABR provides a principled framework for reasoning about dynamic team assignments in spatial domains.

The chicken-and-egg problem of simultaneous team assignment and behavior recognition is conceptually similar to other AI problems, such as image segmentation/object recognition in computer vision. During the image segmentation phase, pixels are assigned to objects that are then classified by an object recognition algorithm. The choices made by segmentation affect the quality of the object recognition; thus one can favor segmentations that generate recognizable objects. In the same way, STABR favors team assignments that produce recognizable behaviors.

Although STABR was designed specifically for the analysis of spatio-temporal traces, we believe that STABR can also be applied to a broader class of problems, where spatial information does not govern team structure. For instance, agents could be assigned to teams based on observed inter-agent communication patterns in conjunction with role templates that represent functional relationship between agents. In such domains, it may be necessary to relax the restriction on team membership to allow an agent to simultaneously belong to multiple teams. This change would simplify the process of generating valid world states since it removes the need for consistency checking at the expense of increasing the number of potential hypotheses that need to be considered.

Our multi-agent plan recognition approach identifies characteristics of the plan library that *compress* the number of potential explanations. The benefits of implementing this as an automatic preprocessing step include the following:

(1) By automatically recovering this hidden structure, we remove some of the burden of plan library authorship from the user.
(2) Pruning and prioritization of the plan library works with a variety of plan recognition algorithms.
(3) Prioritization of plans improves efficiency of plan recognition in the presence of observation noise.

Although there is some amount of hidden temporal structure in single-agent plan libraries, when plans involve the formation of teams, additional structure is created by the enforcement of agent resource requirements.

7. CONCLUSION

In cases where the agent's team composition remains consistent over time, multi-agent plan recognition is no more difficult than single agent plan recognition since the team can usually be treated as a single agent. This article describes several research contributions that can be used to improve the efficiency of a multi-agent plan recognizer in cases where the team composition changes over time:

Formation identification. Many team behaviors in physical domains exhibit distinctive spatial configurations among agents, and between agents and static objects. We present an efficient method for recognizing such team formations. Our method is robust to noise in agent position and high degree of clutter (other agents). Unlike template matching methods that must search exhaustively over a discretized parameter space, our approach only considers those hypotheses that are consistent with a minimal subset of agents, enabling it to scale easily to large agent teams in arbitrary spatial layouts.

Behavior recognition for dynamic agent teams. In complex scenarios, the membership of agent teams changes over time: teams assemble to accomplish specific tasks, create subteams as needed, and disband into individuals. We present an approach for simultaneously recovering agent-to-team assignment and team behavior for such tasks. We efficiently hypothesize team assignments where the spatio-temporal traces for those agents (over a limited time window) match models of known team behavior.

Efficient plan recognition for dynamic agent teams. When team behaviors are generated by higher-level plans, our goal is to match observed agent activity to specific trees in a given plan library. Naive approaches to this problem can be extremely expensive since the number of hypotheses grows combinatorially. A particular challenge is that the actions of splitting and merging in dynamic teams are unobservable and can only be inferred through indirect means. We present efficient pruning techniques based on across-plan and within-plan action constraints generated by static analysis of plan libraries. This enables us to perform plan recognition for large dynamic teams.

REFERENCES

ALI, S. AND SHAH, M. 2008. Floor fields for tracking and high-density crowd scenes. In *Proceedings of European Conference on Computer Vision*.

ALLEN, J. 1983. Recognizing intentions from natural language utterances. In *Computational Models of Discourse*, M. Brady and R. Berwick, Eds. MIT Press.

ALLEN, J., KAUTZ, H., PELAVIN, R., AND TENNENBERG, J. 1991. *Reasoning About Plans*. Morgan Kaufmann Publishers, Chapter 2, pp.121–148.

AVRAHAMI-ZILBERBRAND, D. AND KAMINKA, G. 2005. Fast and complete symbolic plan recognition. In *Proceedings of International Joint Conference on Artificial Intelligence*.

BANERJEE, B., KRAEMER, L., AND LYLE, J. 2010. Multi-agent plan recognition: Formalization and algorithms. In *Proceedings of the 24th AAAI Conference on Artificial Intelligence (AAAI-10)*. 1059–1064.

BEETZ, M., KIRCHLECHNER, B., AND LAMES, M. 2005. Computerized real-time analysis of football games. *Pervasive Computing 4*.

BUI, H. 2003. A general model for online probabilistic plan recognition. In *Proceedings of International Joint Conference on Artificial Intelligence*.

CHARNIAK, E. AND GOLDMAN, R. 1993. A Bayesian model of plan recognition. In *Proceedings of National Conference on Artificial Intelligence*.

CHARNIAK, E. AND MCDERMOTT, D. 1985. *Introduction to Artificial Intelligence*. Addison Wesley.

COHEN, P. AND LEVESQUE, H. 1990. Intention is choice with commitment. *Artificial Intelligence 42*.

DUDA, R., HART, P., AND STORK, D. 2001. *Pattern Classification*. Wiley & Sons, Inc.

FISCHLER, M. AND BOLLES, R. 1981. Random sample consensus: A paradigm for model fitting with application to image analysis and automated cartography. *Communications of the ACM 24,* 6.

GOLDMAN, R., GEIB, C., AND MILLER, C. 1999. A new model of plan recognition. In *Proceedings of Uncertainty in Artificial Intelligence*.

HOOVER, A., MUTH, E., AND SWITZER, F. 2005. Clemson University MOUT Project.

HUBER, M., DURFEE, E., AND WELLMAN, M. 1994. The automated mapping of plans for plan recognition. In *Procceedings of Uncertainty in Artifcial Intelligence (UAI)*.

INTILLE, S. AND BOBICK, A. 1999. A framework for recognizing multi-agent action from visual evidence. In *Proceedings of National Conference on Artificial Intelligence*.

JACQUES JR., J., BRAUN, A., SOLDERA, J., MUSSE, S., AND JUNG, C. 2007. Understanding people motion in video sequences using Voronoi diagrams. *Pattern Analysis and Applications 10*, 321–332.

KAMINKA, G. AND BOWLING, M. 2002. Towards robust teams with many agents. In *Proceedings of International Conference on Autonmous Agents and Multi-agent Systems*.

KAMINKA, G. AND TAMBE, M. 2000. Robust agent teams via socially attentive monitoring. *Journal of Artificial Intelligence Research 12*, pp.105–147.

KAMINKA, G., VELOSO, M., SCHAFFER, S., SOLLITTO, C., ADOBBATI, R., MARSHALL, A., SCHOLER, A., AND TEJADA, S. 2002. Gamebots: A flexible test bed for multiagent team research. *Communications of the ACM 45,* 1.

KAUTZ, H. 1987. A formal theory of plan recognition. Ph.D. thesis, University of Rochester.

LAVIERS, K., SUKTHANKAR, G., MOLINEAUX, M., AND AHA, D. 2009. Improving offensive performance through opponent modeling. In *Proceedings of the AAAI Conference on Artificial Intelligence for Interactive Digital Entertainment*. 58–63.

LIAO, L., FOX, D., AND KAUTZ, H. 2004. Learning and inferring transportation routines. In *Proceedings of National Conference on Artificial Intelligence*.

LIN, D. AND GOEBEL, R. 1991. A message passage algorithm for plan recognition. In *Proceedings of National Conference on Artificial Intelligence*.

MURPHY, K. 2001. The Bayes Net Toolbox for Matlab. *Computing Science and Statistics 33*.

NGUYEN, N., PHUN, D., VENKATESH, S., AND BUI, H. 2005. Learning and detecting activities from movement trajectories using Hierarchical Hidden Markov Models. In *Proceedings of Computer Vision and Pattern Recognition*.

PYNADATH, D. 1999. Probabilistic grammars for plan recognition. Ph.D. thesis, University of Michigan.

RABINER, L. 1989. A tutorial on Hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE 77*.

RILEY, P. AND VELOSO, M. 2002. Recognizing probabilistic opponent movement models. In *RoboCup-2001: Robot Soccer World Cup V*, A. Birk, S. Coradeschi, and S. Tadorokoro, Eds. Springer Verlag.

ROTA, G. 1964. The number of partitions of a set. *American Mathematical Monthly 71*.

SARIA, S. AND MAHADEVAN, S. 2004. Probabilistic plan recognition in multiagent systems. In *Proceedings of International Conference on Automated Planning and Scheduling*.

SCOVANNER, P. AND TAPPEN, M. 2009. Learning pedestrian dynamics from the real world. In *Proceedings of International Conference on Computer Vision*.

SUKTHANKAR, G. 2007. Activity recognition for multi-agent teams. Ph.D. thesis, Carnegie Mellon University.

SUKTHANKAR, G. AND SYCARA, K. 2006. Robust recognition of physical team behaviors using spatio-temporal models. In *Proceedings of Workshop on Modeling Others from Observations (MOO 2005)*.

TAMBE, M. 1996. Tracking dynamic team activity. In *Proceedings of National Conference on Artificial Intelligence*.

TAMBE, M. 1997. Towards flexible teamwork. *Journal of Artificial Intelligence Research 7*, 83–124.

TAMBE, M. AND ROSENBLOOM, P. 1995. RESC: An approach to agent tracking in a real-time dynamic environment. In *Proceedings of International Joint Conference on Artificial Intelligence*.

VILAIN, M. 1990. Getting serious about parsing plans: A grammatical analysis of plan recognition. In *Proceedings of National Conference on Artificial Intelligence*.

WHITE, B., BLAYLOCK, N., AND BÖLÖNI, L. 2009. Analyzing team actions with cascading hmm. In *The 22nd International FLAIRS Conference,*. 129–135.

XU, G. AND ZHANG, Z. 1996. *Epipolar Geometry in Stereo, Motion, and Object Recognition*. Kluwer.

YIN, J., CHAI, X., AND YANG, Q. 2004. High-level goal recognition in a wireless LAN. In *Proceedings of National Conference on Artificial Intelligence*.