Some Challenges in Tracking Agent Teams

Milind Tambe and Ernesto Brodersohn-Ostrovich
Computer Science Dept and Information Sciences Institute
University of Southern California
4676 Admiralty Way, Marina del Rey, CA 90292
{tambe,ernestob}@isi.edu
http://www.isi.edu/soar/tambe

Introduction

In many multi-agent domains, the interaction among intelligent agents — collaborative or non-collaborative — is both dynamic and real-time. Examples include intelligent tutoring systems that interact with students in real-time(Anderson *et al.* 1990), virtual environments for entertainment(Maes *et al.* 1994; Bates, Loyall, & Reilly 1992), "real-world" synthetic environments for training in traffic(Cremer *et al.* 1994) or battlefield simultations(Tambe *et al.* 1995), RoboCup soccer(Kitano *et al.* 1995), and robotic collaboration by observation(Kuniyoshi *et al.* 1994). In most such domains, agent modeling is crucial for effective collaboration and competition. The key aspect of agent modeling of interest here is *agent tracking* — monitoring other agents' observable actions and inferring their high-level goals, plans and behaviors(Anderson *et al.* 1990; Tambe & Rosenbloom 1995; Rao 1994). In contrast with plan recognition in more static domains(Kautz & Allen 1986), this capability focuses on tracking flexible/reactive behaviors in dynamic environments.

Previous work in agent tracking has mostly focused on tracking individual agents. This paper outlines key challenges that arise in going beyond individuals to tracking agent teams; and presents some initial solutions to adress those challenges[1]. Already, a large number of multi-agent systems, both in synthetic and robotic domains, require that agents understand and track team activities (e.g., many of the domains described above, including RoboCup). The basic challenge in tracking teamwork is that it is more than a group of individual agents acting simultaneously, even if in a coordinated fashion(Grosz & Sidner 1990; Cohen & Levesque 1991; Kinny *et al.* 1992; Jennings 1995). For instance, driving in ordinary traffic is not considered teamwork, despite the drivers' simultaneous activity, coordinated by traffic signs(Cohen & Levesque 1991). Conversely, team activity may not be decomposed and tracked as independent actions of individuals. Thus, for instance, if two children are collaboratively building a tower of blocks(Grosz & Sidner 1990), then they cannot be tracked as building two individual towers of blocks with gaps in just the right places. Similarly, a *wall pass* in Soccer cannot be tracked as individuals' independent actions — no one individual executes the wall pass (for non soccer literates, the next section provides an explanation).

While researchers broadly agree that team activity is not merely coordinated activity, its precise nature is still a topic of active research and debate(Jennings 1995). Nonetheless, this paper will rely on one leading theory of teamwork that is based on *joint intentions*, i.e., joint commitments to joint activities in a shared belief state(Cohen & Levesque 1991). Joint commitment implies that team members have a mutual belief that they are each committed to that activity. Furthermore, a team's jointly intending an activity leads subteams to intend to do their share in that activity, subject to the joint intention remaining valid. Finally, should a team member privately discover that the team's joint activity is achieved, unachievable or irrelevant, it must inform other team members — so this private knowledge becomes mutual knowledge. This communication is an overhead of teamwork, which team members bear so as to save the team's time/resources(Jennings 1995).

While the joint intentions framework attempts to outline ideal behavior for team members, its specified communication requirement can be problematic if such communication is costly, risky or redundant. In Soccer, for instance, a player usually may not communicate the failure to execute a pass to its teammate — he/she may waste an opportunity to score a goal in the process. Thus, as ideal teammembers, agents should balance communication costs and benefits (this is an extension to the joint intentions framework). We will refer to teams that adhere to the extended joint intentions framework as *pure teams*.[2] In pure teams, agents fully adhere to the jointness of the team, and engage

[1] Portions of this paper are based on (Tambe 1996b).

[2] One interesting issue here is whether this definition is sufficient to establish the pureness of a team — specifically, team members do not appear to share risks, resources and rewards.

in their task knowing other members will not act selfishly. The pureness of the team degrades as selfishness enters the picture; and in a fully impure team, each agent is on its own.

Focusing first on pure teams, the basic challenge is to track the team's joint intentions. Previous approaches(Anderson *et al.* 1990; Rao 1994; Tambe & Rosenbloom 1995), that focus on tracking individual agents, fail to express and track such joint team activities. In particular, these approaches are based on *model tracing*(Anderson *et al.* 1990), which involves executing an agent's model, and matching the model's predictions with observations. However, an individual's model simply does not express a team's joint goal and activities. For instance, in Soccer, one individual's model simply cannot express a wall-pass — a wall-pass mandates the joint involvement of two or more players.

Real-time, dynamic domains create additional novel challenges. The key challenge in real-time is ambiguity resolution. In particular, disambiguation often requires time; and yet, for timelineness in interaction, a tracker (tracking agent) must frequently resolve such ambiguity in real-time. The presence of multiple agents increases the combinatorics of the search space of disambiguation, however, since combinations of agents' activities have to be disambiguated. Further challenges include:

1. In a dynamic environment, teams may dynamically split into subteams or merge into teams; and subteams may be unable to fully synchronize their activities. Tracking must accomodate such team/subteam split and merge.
2. Recognizing that a group of individuals forms a pure team. This problem can be arbitrarily difficult. For instance, Searle(Searle 1990) contrasts a group of people that start running towards a shelter when it rains and a outdoor ballet where dancers converges on the shelter as part of their choreography. Externally individuals' movements are identical; yet, in one case, they are acting independently, and in the ballet case they form a team!
3. Tracking must also address impure teams, where agents occassionally act selfishly.

The key idea in addressing the above challenges is the adoption of a team perspective. In model tracing terms, this implies executing a team's model, which predicts the actions of the team and its subteams (rather than separate models of individual team members). Team models explicitly encode the joint goals and intentions required to track a team's joint mental state, and they are uniformly applicable in tracking even if an agent is a participant in a team, rather than a non-participant. Team models also facilitate real-time disambiguation. In particular, given their explicitly encoded jointness, recognizing one team-member's actions helps them to quickly disambiguate other members' actions. For instance, if one member of a two-member team is tracked as engaging in a wall-pass, the other agent must necessarily be engaged in the wall pass (at least in a pure team). Furthermore, by abstracting away from individuals, team models avoid the execution of a large number of individual agent models. To track with such team models in real-time dynamic environments, we build on RESC(Tambe & Rosenbloom 1995), an approach for tracking individual agents in such environments.

The new approach, RESC$_{team}$, is aimed at real-time, dynamic tracking of pure teams. It does not address impure teams; and does not address the challenge of recognizing teams. In particular, we assume that teams are either known in advance or easy to detect via simple tests (e.g., agents' physical proximity). In many real-world situations, as in games, teams are indeed known in advance.

One interesting issue raised is whether RESC$_{team}$ is any different from applying RESC to a single individual controlling multiple components. In particular, the issue is whether a team model is any different from the model of a single individual with multiple components. While this observation is to a degree accurate, one key here is to recognize that individuals in a team, unlike components controlled by a single mind, can diverge in their beliefs, and thus may not always be synchronized in their actions. Furthermore, RESC or other agent tracking techniques have previously not focused on such multi-component tracking. Finally, recognizing that a team must be modeled as a unified team model for tracking is itself one key contribution of this paper.

In the remainder of this paper, we first present three different examples of real-world teamwork that illustrate some of the above challenges. We then present details of the team models, and some initial experimental results. The description below assumes as a concrete basis, agents based on an experimental variant of the Soar architecture(Laird, Newell, & Rosenbloom 1987; Newell 1990). This variant architecture, while faithful to most of Soar's basic problem-solving principles — e.g., problem-solving occurs by applying an operator hierarchy to a state — makes it easier for agents to execute own actions while tracking multiple other agents in parallel(Tambe & Rosenbloom 1996).

Three Domains for Team Tracking

The first domain we examine is air-to-air combat simulations based on a real-world simulator commercially developed for the military(Calder *et al.* 1993). A specific scenario illustrated in Figure 1. Here, a pilot agent **D** confronts a team of four enemy fighters **J**, **K**, **L** and **M**. In Figure 1-a, **D** detects the four opponents turning towards its aircraft, and infers that they are approaching it with hostile intent. In Figure 1-b, the four opponents split up into two subteams, and begin a *pincer* maneuver. That is, one subteam (**J** and **M**) starts turning right, while the other subteam (**L** and

K) starts turning left. Their goal is to trap **D** in the center, and attack it from two sides.

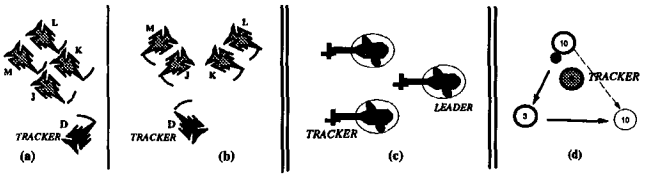

Figure 1: Domains for team tracking: (a-b) air-combat simulation; (c) helicopter simulation; (d) RoboCup simulation. In Robocup, a dashed line indicates a player's movement, while a bold line indicates projected ball trajectory due to a kick.

The key here is an illustration of the basic challenge of team tracking: tracking the jointness of the pincer. The four opponents are not executing independent left and right turns! They are jointly executing a pincer. Yet, an individual agent's model simply fails to express such jointness. Furthermore, the opponents may dynamically select one of many team tactics (pincer is only one of many possibilities) — yet the tracker must disambiguate the tactic in real-time. Here, exploiting jointness is important, since recognizing one subteam's activity aids in recognizing other subteam's activities.

The second domain is teamwork in simulated helicopters(Tambe, Schwamb, & Rosenbloom 1995; Tambe 1996a) (Figure 1-c). Helicopter radio communications are often restricted to avoid detection by enemy. It is thus essential for a helicopter pilot agent to infer relevant information from the actions of its teammates. Consider the simple example of a team given an order to fly to a pre-specified holding point. Here, it is the team that must occupy the holding point; each individual need not do so independently. Indeed, since it is impossible for individual helicopters to remain in formation and also be at the holding point, a pilot must infer the team's presence at the holding area (e.g., based on the teammates' hovering at or near that location). Once again, there is a need to track a team, rather than individuals. The key new issue here is that the tracker is a participant in the team.

The third domain is that of simulated soccer playing agents(Kitano *et al.* 1995). In this scenario two teammates player-10 and player-3 are executing a wall pass to avoid interception of an opponent player-9 (Figure 1-d). After passing the ball to player-3, player-10 moves to a position where player-3 can pass the ball back to it. This constitutes a wall pass, since player-10 essentially treats player-3 as a wall to bounce the ball back to itself. Here, communication is primarily limited to visual contact and tracking; however, it is essential that the player-3 track player-10's intended play for a successful maneuver. It is equally important that player 9 tracks his opponents' teamwork to react accordingly.

RESC: Tracking Individual Agents

The RESC (REal-time Situated Commitments) approach to agent tracking(Tambe & Rosenbloom 1995) builds on *model tracing*(Anderson *et al.* 1990). Here, a tracker executes a model of the trackee (the agent being tracked), matching the model's predictions with observations of the trackee's actions. One key innovation in RESC is the use of commitments. In particular, due to ambiguity in trackee's actions, there are often multiple matching execution paths through the model. Given real-time constraints and the need to react, it is difficult to execute all paths. Therefore, RESC commits to one, heuristically selected, execution path through the model, which provides a constraining context for its continued interpretations. If this commitment lead to a tracking error, it is repaired in real-time. A second key technique in RESC leads to its situatedness, i.e., tracking the trackee's dynamic behaviors. A key assumption here is that the tracker is itself capable of the flexible and reactive behaviors required in the environment. That is, the tracker's architecture can execute such behaviors. Therefore, this architecture is reused to execute the trackee's model to allow dynamic model execution.

To present a concrete example of RESC, consider **D**'s tracking of **J** in Figure 1-a, assuming **J** is the *only* opponent present. Figure 2-a first illustrates **D**'s operator hierarchy, when it is generating own behavior. The top operator, *execute-mission* indicates that **D** is executing its mission (e.g., defend against intruders). Since the mission is not complete, **D** selects the *barcap* operator (barrier combat-air-patrol) in a subgoal. In service of barcap, **D** applies *fly-racetrack-in* in the next subgoal, to fly in a racetrack pattern for its patrol (and lookout for intruders). All these operators used for generating **D**'s own actions will be denoted with the subscript **D**, e.g., $barcap_{\mathbf{D}}$. $\text{Operator}_{\mathbf{D}}$ will denote an arbitrary operator in **D**'s operator hierarchy. $\text{State}_{\mathbf{D}}$ will denote **D**'s state. Together, $\text{state}_{\mathbf{D}}$ and the $\text{operator}_{\mathbf{D}}$ hierarchy constitute **D**'s model of its present dynamic self, referred to as $\text{model}_{\mathbf{D}}$.

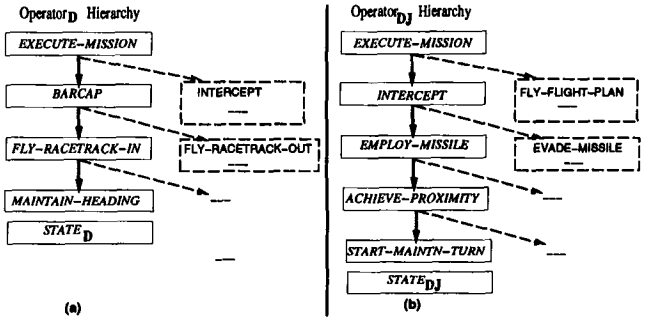

Figure 2: (a) $\text{Model}_{\mathbf{D}}$; (b) $\text{Model}_{\mathbf{DJ}}$. Dashed lines are unselected alternative operators.

To reuse own architecture in tracking, **D** uses a hi-

erarchy such as the one in Figure 2-b to track J's actions. Here, the hierarchy represents D's model of J's current operators (operator$_{DJ}$). This operator$_{DJ}$ hierarchy and the state$_{DJ}$ constitute D's model of J or model$_{DJ}$, used to track J's behavior. For instance, in the final subgoal, D applies the *start-&-maintain-turn*$_{DJ}$ operator, which predicts J's action. If J starts turning right towards D's aircraft, then there is a match with model$_{DJ}$ — D believes that J is turning right to achieve proximity (to employ its missile), as indicated by the higher-level operators$_{DJ}$.

Such architectural reuse provides situatedness in RESC, e.g., operator$_{DJ}$ may now be reactively terminated, and flexibly selected, to respond to the dynamic world situation. As for RESC's commitments, notice that from D's perspective, there is some ambiguity in J's right turn in Figure 1-b — it could be part of achieving proximity or a 150° turn to run away. Yet, D commits to just one operator$_{DJ}$ hierarchy. This commitment may be inaccurate, resulting in a *match failure*, i.e., a difference between model$_{DJ}$'s prediction and the actual action. For example, if J were to actually turn 150°, there would be a match failure. RESC recovers from such failures via "current-state backtracking", which involves backtracking over the operator$_{DJ}$ hierarchy, without revisiting past states (see (Tambe & Rosenbloom 1995) for more details).

Tracking with Team Models

To step beyond tracking individuals, and track a team's goals and intentions, team models are put to service. A tracker's model of a team consists of a team state and team operators. A team state is used to track a team's joint state, and it is the union of a *shared* part and a *divergent* part. The shared part is one assumed common to all team members (e.g., overarching team mission, team's participants). The divergent part refers to aspects where members' states differ (e.g., 3-D positions). One approach to define this divergent part is to compute a region or boundary encompassing all individual members. However, given the cost of computing such regions in real-time, the approach preferred in this work is to equate the divergent part to the state of a single *paradigmatic agent* within the team, e.g., the team's orientation is the paradigmatic agent's orientation. (A paradigmatic agent is selected as one in a prominent location.) Thus, a generic team Θ is represented as $\{m_p,\{m_1...m_p...\}\}$, where m_i are some arbitrary number of team members, and m_p is the paradigmatic agent. Θ may have N sub-teams, $\sigma_1...\sigma_N$, each also possessing its own members, state and paradigmatic agents. Thus, $\mathcal{T}$, the team of opponents in Figure 1-b consists of {J,{J,K,L,M}}, with two subteams $\mathcal{S}_1$={J,{J,M}} and $\mathcal{S}_2$={L,{L,K}}. A single agent is considered a singleton team: $\{m_1,\{m_1\}\}$.

A team operator in a team model represents the team's joint commitment to a joint activity. A key aspect of a team operator is the notion of a *role*, which defines an activity that a subteam undertakes in service of the team operator. The *pincer* team operator in Figure 1-b has two roles LEFT and RIGHT. These roles are exhaustive and specify subteams' activities in service of the team operator. Roles also embody a *role coherency* constraint, i.e., there must be one subteam per role for the performance of the team operator. For the team Θ, a team operator with $\mathcal{R}$ roles is denoted as operator$_\Theta$< $\gamma_1,...,\gamma_R$ >. The children of this operator in the operator hierarchy must then define the activities for subteams in these roles. If a team operator has only a single role, that will not be explicitly denoted.

The RESC$_{team}$ approach to track team activity is now specified as follows:

1. Execute the team model on own (tracker's) architecture. That is, commit to a team operator hierarchy and apply it to a team state to generate predictions of a team's action. In doing so, if alternative applicable operators available (ambiguity):
   (a) Prefer ones where number of subteams equals number of roles.
   (b) If multiple operators still applicable, heuristically select one.
2. Check any tracking failures, specifically, match or role failures; if none, goto step 1.
3. If failure, determine if failure in tracking the entire team or just one subteam. If team failure, repair the team operator hierarchy. If one subteam's failure, remove subteam assignment to role in team operator, repair only subteam hierarchy. Goto step 1.

Step 1 reuses tracker's architecture for flexible team model execution, to track dynamic team activity. Step 1(a) selects among multiple operators based on the number of subteams, while 1(b) relies on domain-independent and dependent heuristics for such selection, e.g., one heuristic is assuming the worst about an opponent in an adversarial setting. The commitment in step 1 creates a single team operator hierarchy. With this commitment is RESC$_{team}$ always has a current best working hypothesis about the team activity.

In step 2, tracking failure is redefined in RESC$_{team}$. Match failure — where a team's actions (e.g., orientation) does not match RESC$_{team}$'s current predictions — is certainly a tracking failure. However, in addition, inaccurate commitments in RESC$_{team}$ can also cause *role failure*, a new tracking failure, which may occur in one of three ways due to violation of the role coherency constraint. First, *role overload* failure occurs if the number of subteams exceeds the number of roles in a team operator. Second, *role undersubscribe* failure occurs if the number of subteams falls short of the required number of roles — particularly, if subteams merge together. Third, *role assignment* failure occurs if the number of subteams equals the number of roles, but they do not match the roles. Both match and role failures cause the same repair mechanism to be invoked — current-state backtracking — although in case of role failures, operators with higher (or lower) number

of roles may be attempted next. (Abstract higher level operators are not susceptible to role overload failures, since they may not restrict the formation of subteams.) Step 3 outlines a novel issue in team tracking: in impure teams, a subteam may deviate on its own, and may not indicate a failure of the whole team. We will not address that issue here (but see (Tambe 1996b)).

The result of RESC$_{team}$ in tracking the situation from Figure 1-a is shown in Figure 3-a. At the topmost level, *execute-mission*$_{\mathbf{D}\mathcal{T}}$ denotes the operator that **D** uses to track $\mathcal{T}$'s joint mission execution. Since $\mathcal{T}$'s mission is not yet complete, **D** applies the *intercept*$_{\mathbf{D}\mathcal{T}}$ operator in the subgoal to track $\mathcal{T}$'s joint intercept. In the next subgoal, *employ-weapons*$_{\mathbf{D}\mathcal{T}}$ is applied. Following that, *get-firing-position*$_{\mathbf{D}\mathcal{T}}$ tracks **D**'s belief that $\mathcal{T}$ is attempting to get to a missile firing position, and so on. Each operator in this operator$_{\mathbf{D}\mathcal{T}}$ hierarchy indicates **D**'s model of $\mathcal{T}$'s joint commitment to that activity.

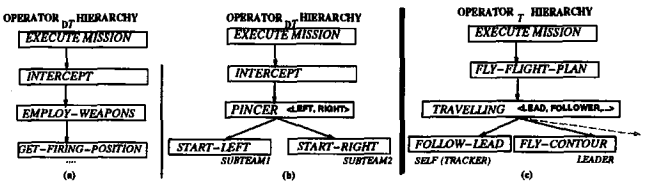

Figure 3: Team tracking in air-combat simulation and helicopter simulation.

When the team in Figure 1-a splits into two subteams, role overload failure causes *employ-weapons*$_{\mathbf{D}\mathcal{T}}$ to fail. After current state backtracking, operator$_{\mathbf{D}\mathcal{T}}$ hierarchy in Figure 3-b results, which correctly tracks the on-going pincer. Here, *pincer*$_{\mathbf{D}\mathcal{T}}$<LEFT, RIGHT> has two roles. The children of this operator specify activities — starting left and right arms of the pincer — for the two subteams formed.

Team models and RESC$_{team}$ provide improved expressiveness required for team tracking. Team operators are expressive, since they explicitly encode a team's joint activity, with roles expressing different activities for subteams. For instance, the team operator hiearchy in Figure 3-b clearly expresses the team's joint pincer, the subteams involved and their roles in it. Team operators also facilitate real-time ambiguity resolution by enforcing jointness. Furthermore, role coherency in team operators adds further constraints, since subteams may only fill unassigned roles. For instance, in Figure 3-b, if one subteam is assigned to the LEFT role of a pincer, the other must also be part of the pincer, and in fact, must fulfill the RIGHT role. Additionally, team models also execute fewer operator hierarchies, e.g., instead of executing four separate operator hierarchies corresponding to four individual opponents, **D** executes only one team operator hierarchy.

Team models and RESC$_{team}$ are applicable for tracking even if an agent is a collaborative participant in the team. To track team activities, a team member executes a team model, using RESC$_{team}$. The result of applying RESC$_{team}$ to the situation in Figure 1-c is shown in Figure 3-c (the tracker happens to be a subordinate in the team). That is, the tracker believes its own team as jointly engaged in *execute-mission*. In service of mission execution, the team is flying a flight plan via a technique called *travelling*. *Travelling* involves a LEAD role, and two other FOLLOWER roles, causing the operator hierarchy to branch out. One key point here is the uniformity of an agent's generation of its own actions and its tracking of its teammates' actions. The tracker executes the *follow-leader* operator branch to *generate* its own behaviors, while executing the *fly-contour* branch to track the leader's flying along the terrain contour. (The third dashed branch is used to track the other follower).

The soccer domain appears more dynamic than the above two domains, and raises additional issues. First, although there are eleven players in a team, not all members are active as participants in the on-going play. Indeed, the "active team" size dynamically varies, which changes the types of tactics possible. Figure 4 provides an illustration — it is player-9's perspective of its opponents' team play (from Figure 1-d). The bold lines are active team operators, while the dashed lines are unselected alternatives. Thus, the oppnents' team is launching an *ATTACK* in service of winning a game. The attack can be a *SINGLE ATTACK* or as is the case here, a *ONE-TWO ATTACK*. The dynamic oscillation of the "active team" size is related to the opportunities presented at any point in time. In the current situation, it is the participation of player-3 into the active team that causes the switch to a *ONE-TWO ATTACK*. As part of the *ONE-TWO ATTACK* Player-9 tracks the on-going pass as a single/wall-pass. A wall-pass maneuver is tracked as two immediate adjacent passes, from a sender to a receiver and then back.

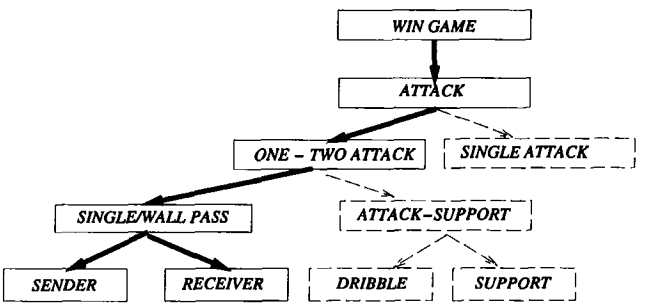

Figure 4: Team tracking: Soccer Hierarchy.

Implementation Results and Evaluation

To evaluate RESC$_{team}$, we have implemented experimental variants of Soar-based pilot agents for

both simulated fighters(Tambe *et al.* 1995) and helicopters(Tambe, Schwamb, & Rosenbloom 1995) (the soccer agents' development is still in preliminary stages) . The original pilot agents have participated in various large-scale exercises, including one involving expert human pilots(Tambe *et al.* 1995). Our experimental pilots (called pilot^{tracker}) incorporate RESC$_{team}$ (contain over 1000 rules). These agents can track teams: opponents' teams in the case of fighter pilots and their own team in case of helicopter pilots.

We have run the pilots^{tracker} agents in several combat simulation scenarios outlined by our human experts. Figure 5-a compares a fighter pilot^{tracker}'s performance when tracking with team models versus when using individuals' models. The scenario in Figure 1-a is used as a basis for comparison, with four agents in the opponents' team. Figure 5-a shows the percent of its total time that pilot^{tracker} spends in acting and tracking. Thus, when using team models in tracking, a pilot^{tracker} spends only 18% of its time is tracking. In contrast, it spends 71% of its time in tracking when using individual agents' models. Basically, individual agents' models *fail* to correctly track the team's pincer. This failure is not simply in terms of inexpressivity, but also, unable to exploit the teams' jointness, they engage in a large unconstrained tracking effort. Thus, they run out of time, before they can each at least individually detect the pincer. Similarly, with team models, pilot^{tracker} spends only 7% of its time in deciding on its own actions(SELF), since it can quickly and accurately track its opponents. In constrast, pilot^{tracker} incorrectly readjusts its own maneuvers when using individual models; hence spends 28% of time deciding on its own actions. Figure 5-b provides similar comparative numbers for a team consisting of three opponents. The key point here is that team models are not wedded to a specific numbers of agents in a team.[3]

Summary and Discussion

Our world is full of collaborative and competitive team activities: in team-sports (soccer, cricket, hockey), an orchestra, a discussion, a coauthored paper, a play, a military alliance, etc. It is only natural that this teamwork is (and will be) reflected in virtual and robotic agent worlds, e.g., robotic collaboration by observation(Kuniyoshi *et al.* 1994), RoboCup robotic (and virtual) soccer(Kitano *et al.* 1995), virtual theatre(Bates, Loyall, & Reilly 1992), virtual battlefields(Tambe *et al.* 1995). If agents are to successfully inhabit such worlds, they must be proficient in understanding and tracking team activity. This paper has taken a step towards this goal by outlining some key questions and presenting some initial solutions. Key contributions/ideas in this paper include: (i) the use of explicit team models for team tracking; (ii) uniform application of team models regardless of an agent's being a participant or non-participant in a team; and (iii) some demonstration of the efficiency and expressivity gained via team models.

[3]Thus, with team models, an agent spends 25% (18% TRACKING + 7% SELF) of time in mental activity, and the rest it waits. Waiting is essential because pilot^{tracker}'s maneuvers take time, e.g., to complete a turn, or reach missile range. When using individual agents' models, most of the cycles are spent tracking.

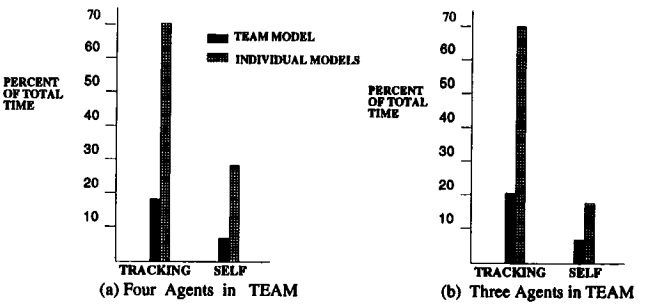

Figure 5: Comparing efficiency of tracking with team and individual models.(a) Four Agents in TEAM; (b) Three Agents in TEAM

While team models could be applied in different approaches to tracking, we presented one specific approach: RESC$_{team}$. Although based on RESC, RESC$_{team}$ includes several additions to address subteam formation(merging) and role assignments. RESC$_{team}$ has been (or is being) applied to three different simulation tasks: (i) air-combat simulation; (ii) helicopter mission simulation; (iii) robocup soccer simulation. We hope these synthetic yet real-world teamwork tasks provide a solid foundation for further investigation of team tracking.

References

Anderson, J. R.; Boyle, C. F.; Corbett, A. T.; and Lewis, M. W. 1990. Cognitive modeling and intelligent tutoring. *Artificial Intelligence* 42:7–49.

Bates, J.; Loyall, A. B.; and Reilly, W. S. 1992. Integrating reactivity, goals and emotions in a broad agent. Technical Report CMU-CS-92-142, School of Computer Science, Carnegie Mellon University.

Calder, R. B.; Smith, J. E.; Courtemanche, A. J.; Mar, J. M. F.; and Ceranowicz, A. Z. 1993. Modsaf behavior simulation and control. In *Proceedings of the Conference on Computer Generated Forces and Behavioral Representation.*

Cohen, P. R., and Levesque, H. J. 1991. Teamwork. *Nous* 35.

Cremer, J.; Kearney, J.; Papelis, Y.; and Romano, R. 1994. The software architecture for scenario control in the Iowa driving simulator. In *Proceedings of the Conference on Computer Generated Forces and Behavioral Representation.*

Grosz, B. J., and Sidner, C. L. 1990. Plans for discourse. Cambridge, MA: MIT Press. 417–445.

Jennings, N. 1995. Controlling cooperative problem solving in industrial multi-agent systems using joint intentions. *Artificial Intelligence* 75.

Kautz, A., and Allen, J. F. 1986. Generalized plan recognition. In *Proceedings of the National Conference on Artificial Intelligence*, 32–37. Menlo Park, Calif.: AAAI press.

Kinny, D.; Ljungberg, M.; Rao, A.; Sonenberg, E.; Tidhard, G.; and Werner, E. 1992. Planned team activity. In Castelfranchi, C., and Werner, E., eds., *Artificial Social Systems, Lecture notes in AI 830.* Springer Verlag, New York.

Kitano, H.; Asada, M.; Kuniyoshi, Y.; Noda, I.; and Osawa, E. 1995. Robocup: The robot world cup initiative. In *Proceedings of IJCAI-95 Workshop on Entertainment and AI/Alife.*

Kuniyoshi, Y.; Rougeaux, S.; Ishii, M.; Kita, N.; Sakane, S.; and Kakikura, M. 1994. Cooperation by observation: the framework and the basic task pattern. In *Proceedings of the IEEE International Conference on Robotics and Automation.*

Laird, J. E.; Newell, A.; and Rosenbloom, P. S. 1987. Soar: An architecture for general intelligence. *Artificial Intelligence* 33(1):1–64.

Maes, P.; Darrell, T.; Blumberg, B.; and Pentland, S. 1994. Interacting with animated autonomous agents. In Bates, J., ed., *Proceedings of the AAAI Spring Symposium on Believable Agents.*

Newell, A. 1990. *Unified Theories of Cognition.* Cambridge, Mass.: Harvard Univ. Press.

Rao, A. S. 1994. Means-end plan recognition: Towards a theory of reactive recognition. In *Proceedings of the International Conference on Knowledge Representation and Reasoning (KR-94).*

Searle, J. R. 1990. *Collective intention and action.* Cambridge, MA: MIT Press. 401–415.

Tambe, M., and Rosenbloom, P. S. 1995. RESC: An approach for real-time, dynamic agent tracking. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI).*

Tambe, M., and Rosenbloom, P. S. 1996. Architectures for agents that track other agents in multi-agent worlds. In *Intelligent Agents, Volume II: Lecture Notes in Artificial Intelligence 1037.* Springer-Verlag, Heidelberg, Germany.

Tambe, M.; Johnson, W. L.; Jones, R.; Koss, F.; Laird, J. E.; Rosenbloom, P. S.; and Schwamb, K. 1995. Intelligent agents for interactive simulation environments. *AI Magazine* 16(1).

Tambe, M.; Schwamb, K.; and Rosenbloom, P. S. 1995. Building intelligent pilots for simulated rotary wing aircraft. In *Proceedings of the Fifth Conference on Computer Generated Forces and Behavioral Representation.*

Tambe, M. 1996a. Executing team plans in dynamic, multi-agent domains. In Pryor, L., ed., *AAAI FALL Symposium on Plan Execution: Problems and Issues.*

Tambe, M. 1996b. Tracking dynamic team activity. In *Proceedings of the National Conference on Artificial Intelligence (AAAI).*