Role-Based Teamwork Activity Recognition in Observations of Embodied Agent Actions

Linus J. Luotsinen
School of EECS
University of Central Florida
Orlando, Florida
lluotsin@mail.ucf.edu

Ladislau Bölöni
School of EECS
University of Central Florida
Orlando, Florida
lboloni@eecs.ucf.edu

ABSTRACT

Recognizing team actions in the behavior of embodied agents has many practical applications and had seen significant progress in recent years. One approach with proven results is based on HMM-based recognition of spatio-temporal patterns in the behavior of the agents. While it had been shown to work on real-world datasets, this approach was found to be brittle.

In this paper we present two contributions which together can significantly increase the robustness of teamwork activity recognition. First we introduce a technique to reduce high dimensional continuous input data to a set of discrete features, which capture the essential components of the team actions. Second, we prefix the actual team action recognition with a role recognition module, which allows us to present the recognizer with arbitrarily shuffled input, and still obtain high recognition rates.

We validate the improved accuracy and robustness of the team action recognizer on datasets derived from captured real world data.

Categories and Subject Descriptors

I.2.11 [**Distributed Artificial Intelligence**]: Multi-Agent Systems; I.5.4 [**Pattern Recognition**]: Applications

General Terms

Algorithms, Experimentation, Human factors

Keywords

teamwork, roles, recognition, embodied agents

1. INTRODUCTION

Recognizing teamwork in embodied agents has important applications in areas such as surveillance, training, automated annotation and commentary, as well as in improving the ability of robotic agents to participate in human teams. Most of the time, the teamwork of embodied agents have a natural expression in *coordinated movement* of agents playing specific roles in the team[1]. Examples of teams displaying such patterns are sport teams such as football, basketball or soccer, dance groups, pickpockets in action, groups of animals such as wolf-packs, bodyguards defending a celebrity, military units in the battlefield or a MOUT environment, tank units, air squads, terrorists in a busy marketplace and many others.

[1]Naturally, teamwork can have many other aspects, in addition to or replacing coordinated movement; examples are voice commands and wireless communication.

Cite as: Role-Based Teamwork Activity Recognition in Observations of Embodied Agent Actions, L.J. Luotsinen and L. Bölöni, *Proc. of 7th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2008)*, Padgham, Parkes, Müller and Parsons (eds.), May, 12-16., 2008, Estoril, Portugal, pp. 567-574.

Teamwork activity recognition was subject of significant research effort in recent years. The general consensus is that although some team actions can be recognized based on static configuration, for more complex team actions we need to consider the temporal evolution of the agents. Hidden Markov Models (HMMs) are one of the proven approaches to model temporal evolution of phenomena; indeed, a majority of team action recognition efforts use HMMs or closely related models. The HMM is a representation of the *idealized team action* (ITA), which can be either knowledge engineered (hand crafted) or acquired through learning. These efforts had shown good recognition accuracy for datasets ranging from simulations to real world data recorded in controlled settings (such as military exercises). However, the existing systems are too brittle for most real-world deployments. Perturbations due to the terrain or the natural variability in the behavior of the agents can make the HMM unable to recognize the team action. One additional challenge of the teamwork activity recognition is that for most applications, the recognized action needs to be matched with the human perception of the same action. Humans might label in identical ways actions which are very different in their physical description (U-turn to the left or U-turn to the right), or they might distinguish actions based on knowledge of the agents and the roles (for instance, "stalking", "following" and "chasing" map to almost the same physical description, they are differentiated by the role of the agents: victim, leader, prey).

Let us now see some of the challenges faced by a team action recognition system, which is assumed to have a representation of the ITA:

(a) Observation noise.
(b) Alignment problems: translation and rotation of the team actions compared to the ITA.
(c) Scaling problems: the movement of the team members happen at a different physical scale compared to the ITA.
(d) Temporal scaling: the movement happens slower or faster than in the ITA.
(e) Terrain distortion: the movement patterns are distorted as an adaptation to the particularities of the terrain, such as obstacles or roads.
(f) Movement variants: the team action has several alternative ways of execution, which map to the same human label.
(g) Role count variants: teams with different number of agents perform the action with the same label (for instance *n* bodyguards surround a person).
(h) Uncertainty regarding the role of the agents in the team ac-

tion. An important sub-case is the agents which appear in the scene but do not participate in the team action (agents in the bystander role).

(i) Agents changing their roles during the team action.

Different recognizer systems have already addressed at least a subset of these problems. The problem of observation noise (a) can be addressed by using Kalman or particle filters on the data stream. The alignment problems (b) can be handled by the normalization of the observations. This is usually done by transforming the agents into a coordinate system aligned with the centroid of the team. This, however, can be problematic if the team is not known in advance, as bystander agents might shift the position of the centroid. Furthermore, for many team actions, the physical centroid might not be the logical center of alignment of the team. For a team of bodyguards guarding a celebrity, the location and the movement of the celebrity is the logical center and direction of the team. Scaling problems (c) can be handled either through normalization or by training several HMMs for various sizes. To some level, the problem of temporal scaling (d) is already addressed by the HMM. However, certain movement patterns such as "alternative advance of the two sub-teams for arbitrary, but equal time" can not be encoded in the HMM, because it would require historical information, and the Markov property assumes that the evolution of the system does not depend on its history.

One example of movement variant (f) is the case when a group of agents perform a U-turn by turning to the left or to the right while maintaining formation. While this maneuver maps to the same label, it requires very different movement patterns and role assignments. A simple way to handle such variants is considering the different actions internally and only labeling them identically at the user interface level. In other cases, however, the differences are more subtle. One problem, naturally, is that these variants are labeled identically in training set. Learning techniques based on unimodal distributions might not be able to accommodate these variants.

This paper describes our ongoing efforts to improve the performance of teamwork activity recognizer systems. We present two separate contributions, which are integrated in a full team action recognition workflow. Section 3 describes an approach for discrete feature extraction from the original data. The discrete features were carefully selected to match the ways in which a human observer would understand and analyze the scene; we favor features which refer to teams as a whole, rather than individual agents. Beyond reducing the dimensionality of the dataset, the features were chosen to address challenges (a), (c), (e) and, to some level,(g).

Section 4 describes a role recognition system, which allows the workflow to recognize the likely role of the agent which it would play in a certain team action before the actual team action was recognized. This component directly addresses challenge (h), and represents a necessary precondition to address challenges (g) and (i). The latter, however, are outside the scope of this paper.

The full team action recognition workflow is described in Figure 1. The contributions described in this paper are marked in shaded rectangles.

Section 5 describes the results of a series of experiments based on datasets derived from the recordings of a real world military exercise, and show that the contributions of this paper significantly improve the usability and robustness of team action recognition systems.

2. RELATED WORK

In this section we review work from both the teamwork activity recognition and the role recognition domains.

Teamwork Activity Recognition. Intille et. al. developed a system for recognizing multi-agent activities [5]. The system was evaluated on real-world dataset collected from American Football plays. Temporal logic and high-level descriptors are used together with Bayesian networks to classify 10 different Football plays.

Liu et. al. developed the Observation Decomposed Hidden Markov Model (ODHMM) for multi-agent activity recognition [9]. In the ODHMM approach, team observations are separated into sub-observations that capture the features associated with each individual agent in the team. Following the independence assumption, ODHMMs can recognize teamwork activity in sequences where the number of agents in the team may vary. Furthermore, ODHMM extends the traditional HMM with a role parameter to maintain the relationship between agents and their roles. Experimental results show that multi-agent activities modeled using ODHMMs outperforms discrete HMMs on an artificial dataset.

Sukthankar et. al. [15] developed the STABR-algorithm for simultaneous team assignment and behavior recognition. Initially, a set of possible team assignments are identified, using template matching, from the spatial relationships of independent agents. Next, teams that do not follow any of the pre-defined, parametric, behavior models over a period of time are filtered out. The STABR algorithm was evaluated, with good performance, for teamwork behavior recognition in a simulated military domain.

In a recent study by Vail et. al. [17] Conditional Random Fields (CRF), which are related to HMMs, were employed for activity recognition. Results show that CRFs outperform HMMs for activity recognition in a simple test scenario. One drawback of using CRFs is that significantly more time is required to estimate the models from observations.

Role Recognition. Multi-agent system frameworks such as STEAM [16] and Machinetta [14] execute teamwork tasks by carefully assigning roles to agents using predefined team organization knowledge. In teamwork activity recognition systems, however, the roles are often unknown and need to be inferred from activity observations.

In [11] Prasad et. al. developed the L-TEAM framework which use supervised machine learning to automatically learn organizational roles in multi-agent systems. By inferring organizational role knowledge, agents in the framework negotiate which role to play given the task. The learning approach eliminates the need to manually identify role organization for teams as is done in other frameworks such as STEAM [16] and Machinetta [14].

Overhearing is an approach for monitoring teamwork activity in distributed and open multi-agent systems [4] by listening to the conversation between agents in the team. In many monitoring applications the goal is to determine whether the team is progressing towards its joint goal or not. Overhearing systems have been developed to perform complex tasks such as teamwork plan recognition [7], building and maintaining group formations [8] and role recognition [13].

In [13], Rossi et. al. propose an approach for recognizing social roles of agents in a team using rule based systems. Social roles are assigned to agents by an overhearing agent that can sense the communication between all agents in the team. The system was successfully validated against artificially generated organizations that followed well known protocols.

In [3], Guessoum et. al. develop a fault tolerant multi-agent framework where the most critical agents are recognized and replicated. The *criticality* is determined by recognizing activities and roles of each agent in the multi-agent system. Similarly to overhearing approaches, roles are assigned to agents by employing a

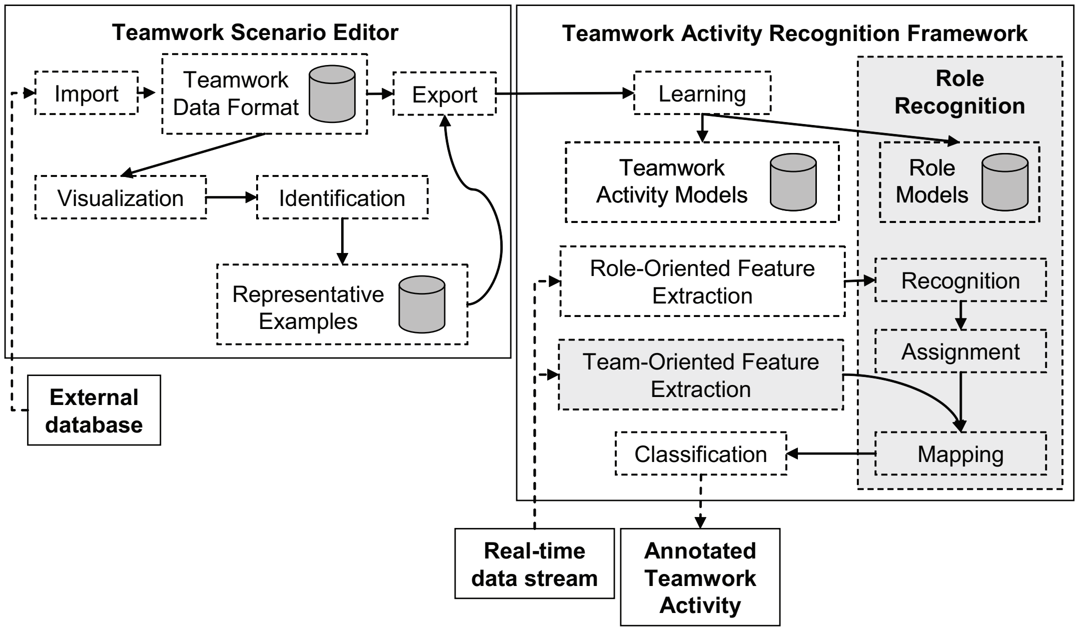

Figure 1: The workflow of teamwork activity recognition system. The contributions of this paper are marked with shaded rectangles.

recognition process to sequences of communication and social interaction between agents.

3. TEAM-ORIENTED FEATURE EXTRACTION

In the team action recognition workflow (Figure 1), the input of the recognition activity is the geometrical location of the agents in two dimensional coordinates, with, potentially, the addition of values such as orientation, or gaze. Basic preprocessing, such as alignment on the centroid does not change the nature of this data. As all agents have the same set of coordinates, the dimensionality of the input increases linearly with the number of agents in the scene. Although for small teams of 4-5 agents the resulting input space is still marginally acceptable for direct input to the HMM, the training process is facing difficulties in extracting relevant information. One problem is that important structural information is lost in the form of the data presented to the HMM. For instance, if we represent the input space with an 8 element vector $\{x_1, y_1, x_2, y_2, x_3, y_3, x_4, y_4\}$ the HMM will not know that x_1 and y_1 are referring to the same agent or that y_1, y_2, y_3 and y_4 are referring to the same direction.

In this section we describe a process through which the raw physical measurements with continuous values are replaced with semantically rich discrete features. The main goal of the discrete value extraction is to focus the HMM on the features used by humans when identifying a team action. Many of these features, such as velocity or curvature are extracted over a sliding window of time, thus containing historical information which is very difficult to encode in a HMM. The goal is not dimensionality reduction; in fact, as we will see, the number of discrete features exceeds the number of continuous values.

We classify the extracted features in agent-oriented, environment-oriented and team-oriented feature functions.

Agent-oriented feature functions extract information from the movement of a single agent over a sliding time-window. The features, together with the allowed discrete values, are summarized in

Table 1: Overview of agent-oriented feature functions.

Name	Discretization	Description
Velocity	{Idle, Low, High}	The rate of change of position.
Acceleration	{Accelerating, Decelerating, Constant}	The rate of change of velocity.
Orientation	{North, South, West, East, NorthEast, NorthWest, SouthEast, SouthWest}	Eight-way orientation with respect to a global reference system.
Turning	{Steady, Left, Right}	The change of orientation.
Curvature	{Straight, Curved}	The rate at which a curve changes direction.

Table 1. As they refer to individual agents, agent oriented features can not efficiently describe the correlations among the team members. They can, however, improve the recognition performance when used in combination with team oriented feature functions. Agent oriented features also play a significant role in recognizing team actions where specific agents play determining roles such as the leader or captain.

Environment oriented feature functions extract features from interactions of the team members with objects in the environment. The objects of the environment can be physical or virtual; both types can be static or dynamic. Terrain features such as buildings or roads are physical and static. The offside line in a soccer game is a virtual and dynamic environmental object that can be inferred from the position of the opponent player who is second nearest to the end of the playing field. The front-line in a war is a similar dynamic and virtual object. Table 2 provides a summary of the environment-oriented feature functions.

Table 2: Overview of environment-oriented feature functions.

Name	Discretization	Description
Collision	{?Object, None}	Collision with nearest environment object.
LineOfSight	{?Object}	Nearest environmental object in the agent's LOS.
LineOfSight Distance	{Near, Far, None}	Euclidean distance to object in LOS.

Table 3: Overview of the second type of team-oriented feature functions.

Name	Discretization	Description
Centroid-Relative Position Vector	$\{A_1, A_2, \ldots, A_n\}$	The relative eight-way position of each team member with respect to team centroid and direction.
Role-Relative Position Vector	$\{A_1, A_2, \ldots, A_{n-1}\}$	The relative eight-way position of agents with respect to a privileged agent's centroid and direction.
Cohesion	{Separated, Merged}	The bonding together of team members.
Cohesion Gradient	{Separating, Merging, None}	The rate at which the bonding of team members are changing.
Cohesion Direction	{Vertical, Horizontal, Upward, Downward}	The principal component direction of the positions of team members.

Team-oriented feature functions extract features expressed relative to the team. We consider two types of team-oriented features. The first type are obtained by replacing the (potential) team with a virtual agent following the team centroid. Thus we obtain the features *TeamVelocity*, *TeamAcceleration*, *TeamOrientation*, *TeamTurning* and *TeamCurvature*.

In addition to these features we designed a series of team-oriented features which are not directly related to agent-oriented feature functions. These features are summarized in Table 3.

Although we cannot exhaustively discuss all feature functions due to space constraints, we discuss the justification and implementation of some of the more complex features.

Centroid-Relative Position Vector (CRPV) is a vector of size n where for every agent we record the semi-quadrant it occupies in a Cartesian coordinate system centered in the centroid, with the *North* direction aligned with the movement of the team centroid over a sliding window in time. Figure 2 shows the encoding in the CRPV of a team of three agents.

There are several advantages of the CRPV compared to the absolute positions of the team members. First, the dimensionality of the input data is reduced. Second, the CRPV is invariant to translation, rotation and scale.

Determining the CRPV does not require full role recognition to be performed before its calculations, but it requires us to eliminate

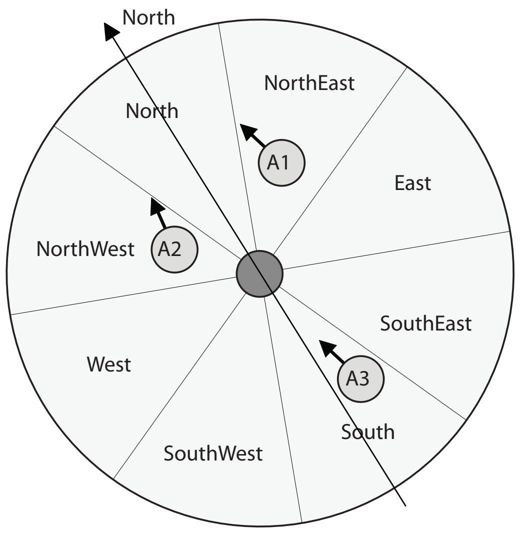

Figure 2: The centroid-relative position vector a team of three agents. The team centroid is marked with a dark shaded circle. The position of agents *A1*, *A2* and *A3* are described by the vector {*NorthEast, NorthWest, South*}.

the bystander agents (which would shift the position of the centroid). Thus, CRPV can serve as a feature used in role recognition.

Role-Relative Position Vector (RRPV) is an evolution of the CRPV feature. The RRPV is a vector of size $n - 1$, where each element encodes the semi-quadrant occupied by the agent in a coordinate system centered on one agent which has a privileged role. The coordinate system is aligned with the movement of the privileged agent. One of the advantages of such system is that we can, for instance, express the movement of the bodyguards relative to a VIP, or the movement of a group of pickpockets around a victim.

The disadvantage of this encoding is that at least the role of the privileged agent needs to be determined before feature extraction. Finally, the privileged role needs to be clearly marked in the ITA, which requires human intervention in the process.

Cohesion, Cohesion Direction and Cohesion Gradient. Intuitively, the *Cohesion* feature measures how tightly grouped together are the team members, the *CohesionDirection* shows the direction of the greatest cohesion, while *CohesionGradient* measures the variation of the cohesion. For instance, a team which is splitting in two groups is loosing cohesion. To extract a set of values which match well with this intuition, we use a method based on principal component analysis [2]. We first calculate a scatter matrix

$$\mathbf{S} = \sum_{\mathbf{k=1}}^{\mathbf{n}} (\mathbf{x_k} - \mathbf{m})(\mathbf{x_k} - \mathbf{m})^{\mathbf{t}}$$

where $\mathbf{x_k}$ is the position of agent k in a team consisting of n members and $\mathbf{m}$ is the location of the centroid:

$$\mathbf{x_k} = \begin{pmatrix} x_k \\ y_k \end{pmatrix} \text{ and } \mathbf{m} = \frac{1}{n} \sum_{k=1}^{n} \mathbf{x_k}$$

Next, we analyze the scatter matrix by finding its eigenvalues, $\boldsymbol{\lambda}$, and eigenvectors, $\mathbf{e}$. In our case, since the position of each agent is described in two dimensions, we will find two eigenvalues and two eigenvectors. Geometrically $\mathbf{e}$ and $\boldsymbol{\lambda}$ represent the principal directions of the team member positions and the magnitude of the directions respectively. The *Cohesion* feature is the value of the largest eigenvalue, λ_{max}.

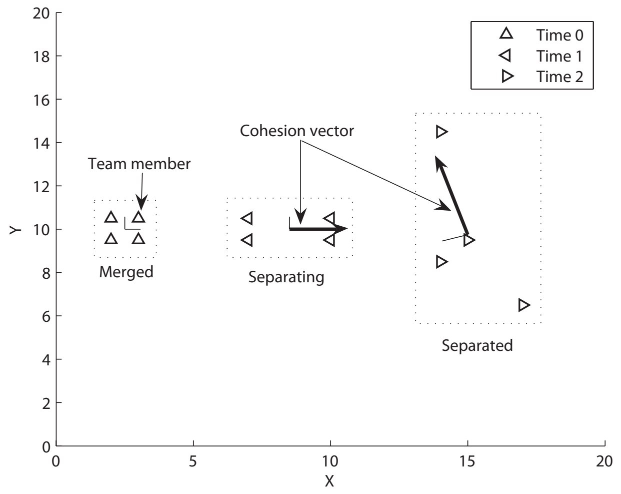

Figure 3: Plot of a scenario where a team of four agents change formation over time. The cohesion vector depicts the cohesion and principal orientation of the team.

The *CohesionDirection* feature function uses the eigenvector corresponding to λ_{max} to output the principal orientation of the team.

The *CohesionGradient* feature calculates the change of the cohesion value over a sliding time window of the historical values of cohesion.

Figure 3 illustrates a scenario where a team of four agents change formation over time. For each time step the figure shows the principal orientation vectors of the team. As discussed above, the larger of these vectors is chosen as the cohesion vector (and it is represented in bold in the figure). Initially, at $Time = 0$, the team members are in near vicinity of each other; λ_{max} is small and the output of the *Cohesion* feature is *Merged*. Next, at $Time = 1$, the team members separate in the x-direction; λ_{max} is increasing and the output of the *CohesionGradient* feature is *Separating*. Finally, at $Time = 2$, the separation of the team yields a large enough λ_{max} value, such that the output of the *Cohesion* feature is *Separated*.

Curvature is a measure of the rate at which the trajectory of the agent changes direction. A simple measure of the curvature can be defined as:

$$\kappa(x) = \frac{|f''(x)|}{(1 + (f'(x))^2)^{3/2}}$$

A small value for $\kappa(x)$ indicates that the movement trace is straight and when $\kappa(x)$ is large the movement trace is curved.

The first and second derivative can be calculated directly from the movement trace of each agent (or team centroid) over a sliding window in time using a finite difference approximation (with central difference).

4. ROLE RECOGNITION

In systems which recognize teamwork from coordinated movement of agents, the input of the recognizer is a fixed arrangement of the recorded features of the agents. A four agent team might be represented by a feature vector $\{x_1, y_1, x_2, y_2, x_3, y_3, x_4, y_4\}$ which is either the direct input of the HMM, or the input to the discretization process, as shown in the previous section. However, what does "agent 3" mean? If we are observing a team sport, such as basketball, we can say that "agent 3" corresponds to the player wearing the number 3. However, in many applications such as surveillance, such a clear identification might not be possible. Even in the case of team sports, players might switch roles during the game. An HMM which was trained with an input order $\{1, 2, 3, 4\}$ will not recognize the same action if it appears encoded in the order $\{2, 3, 4, 1\}$. A related issue is concerned with the number of agents participating in the team action. Many essentially identical team actions can be executed with various team sizes. For instance, the defining feature of bounding overwatch is the alternating advancement of two groups of agents. This definition remains valid whether the action is performed in a configuration of 1+2, 2+1, 2+2 or 2+3 agents.

One brute force solution is to train a separate HMM for every possible team and role size combination. Then, at recognition time, we repeat the matching algorithm for every possible permutation of the input data, and pick the one which gives the best match. This is a significant computational penalty, especially in cases where the analyzed observations contain agents which are not part of any team.

In the following, we describe a role recognition module which allows us to recognize what are the (likely) roles played by the agents in a team action before the team action was recognized. Running such a role recognition module before the team recognition allows us to build the ITA based on roles, rather than agents.

4.1 Learning Role Models from Observations

A role model allows us to determine the likely roles the agents would play in a team action. To understand how this is possible without the need to recognize the team action as well, let us remember that certain roles are strongly identified with certain feature values.

For instance, in a "follow the leader" action, the leader is at the forefront of the team. Looking at the formation, we can identify which agent that "can be" the leader, even before we are certain that the team action is, indeed, "follow the leader". The best feature to decide on this is the centroid-relative position vector; the team leaders will need to be in the *North* semi-quadrant.

For identifying the two sub-teams of the bounding overwatch team action, however, the relative positions are useless. The best features for this determination are the velocity vectors: the members of one team are stationary, while the other ones are moving.

We can draw several conclusions from these examples. First, we need a different feature set for role recognition than for team action recognition (although overlaps, of course, exists). As role recognition is targeted towards individual agents, the agent-oriented feature functions have a significant role. Second, the discriminating feature changes from role to role. Thus, for our role recognition model we chose a decision tree representation, which has the advantage that besides representing the role model, it also allows us to extract and visualize the exact features which were used in the classification.

The decision tree learning algorithm used in our study was the ID3 algorithm followed by a tree pruning procedure to minimize the effects of over-fitting [12].

Figure 4 illustrate an example of a role model represented by a decision tree. The tree was trained using discretized agent observations for four different roles. The dashed rectangles represent leaves in the tree, which in turn contains class frequencies, $\mathbf{f} = \{f_{r_1}, f_{r_2}, f_{r_3}, f_{r_4}\}$, for the roles. The solid rectangles are internal nodes depicting the features required for classification.

4.2 Acquiring Role Assignment Probabilities

The decision trees, once trained, can be used to calculate the role assignment probabilities. The first step is to extract observation sequences from the movement traces of each agent, which are used

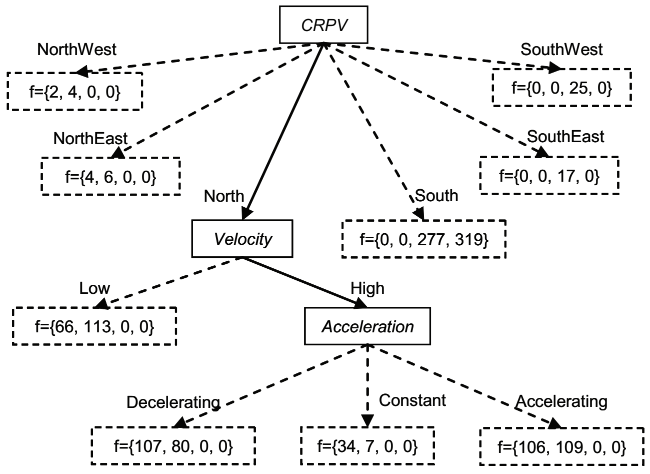

Figure 4: Role model represented by a decision tree.

as input to the decision tree classifier. The output of the classifier is a class frequency vector, $\mathbf{f}^t$, for every observation t in the sequence. The probability $Pr(a_i, r_j)$ that agent a_i is playing role r_j is calculated with the following formula:

$$Pr(a_i, r_j) = \frac{\sum_{t=1}^{T} \mathrm{f}_{r_j}^t}{\sum_{k=1}^{K} \sum_{t=1}^{T} \mathrm{f}_{r_k}^t}$$

where T is the length of the observation sequence and K is the number of available roles.

4.3 Role Assignment and Mapping

The goal of the role assignment process is to identify the best match of role to agent assignments by searching the $Pr(a_i, r_j)$ values so that the team-oriented feature vectors, which are used by the teamwork recognizer, can be re-mapped accordingly. In this section we describe two distinct strategies for this purpose.

The first strategy, described in Algorithm 1, allows agents to play multiple roles. For each role in the Pr table the agent with the maximum probability is selected to represent the role.

The second strategy, described in Algorithm 2, limits agents to play only one role. A set of candidate assignments is first identified for all available agents. The candidate set is generated by traversing the roles of each agent and for each agent selecting the role with maximum probability. Next, conflicts are resolved among the candidate assignments, by selecting the agent with maximum probability to represent the role. The agent is then marked as already assigned a role and not available for future allocation. The procedure is repeated until all roles have been assigned to an agent.

Algorithm 1 Multiple role assignment algorithm.

Require: Role model, *RoleModel*, and agent observations, $\mathbf{O}_{a_i}$.
Ensure: Mapping, *Map*, of agents, a_i, to roles, r_j.

```
1: $Pr(a_i, r_j)$ = Apply($RoleModel$ to $\mathbf{O}_{a_i}$)
2: for all Roles $r_j$ in $Pr(a_i, r_j)$ do
3:     $i_{\max} = \arg\max_i Pr(a_i, r_j)$
4:     Assign($r_j \rightarrow a_{i_{\max}}$ in $Map$)
5: end for
6: return $Map$
```

5. EXPERIMENTAL RESULTS

Algorithm 2 Unique role assignment algorithm.

Require: Role model, *RoleModel*, and agent observations, $\mathbf{O}_{a_i}$.
Ensure: Unique mapping, *Map*, of agents, a_i, to roles, r_j.

```
1: $Pr(a_i, r_j)$ = Apply($RoleModel$ to $\mathbf{O}_{a_i}$)
2: while Unmapped roles remaining do
3:     for all Agents $a_i$ in $Pr(a_i, r_j)$ do
4:         $j_{\max} = \arg\max_j Pr(a_i, r_j)$, where $j \neq k$, $\forall r_k$ in $Map$
5:         Assign($r_{j_{\max}} \rightarrow a_i$ in $HypothesisMap$)
6:     end for
7:     for all Roles $r_{j_{\max}}$ in $HypothesisMap$ do
8:         $i_{\max} = \arg\max_i Pr(a_i, r_{j_{\max}})$, where $i \in k$, $\forall a_k$ of $r_{j_{\max}}$
           in $HypothesisMap$
9:         Assign($r_{j_{\max}} \rightarrow a_{i_{\max}}$ in $Map$)
10:    end for
11: end while
12: return $Map$
```

Table 4: Dataset summary of teamwork activities, number of sequences and observations that were used during training and validation.

Activitiy	Sequences	Observations
Traveling column	29	319
Traveling line	30	330
Traveling box	33	363
Bounding overwatch	9	189
Wedge	17	187
Team split	15	192
Team merge	15	177

5.1 Experimental Setup

We performed a series of experiments using a dataset which was acquired from a real world military exercise. The dataset consists of observations of four tanks which performed seven distinct teamwork activities. To achieve a larger corpus, the approximately 40 initial team action observations were multiplied synthetically by applying various distortion operations. The resulting semi-synthetic dataset is summarized in Table 4. The original observations were made using high resolution GPS devices embedded in the tanks, thus for practical purposes, they are free of observation noise. For practical applications, which require observations of opponent agents, such accuracy is not achievable. To simulate observation noise in our dataset, we have added Gaussian noise to the position of each agent by offsetting the coordinates with a randomly generated number following the Gaussian distribution, $N(\mu = 0, \sigma^2 = 1)$, multiplied with a noise magnitude ϵ.

5.2 Creating the Idealized Team Actions

In our workflow the idealized team actions were encoded in the form of an HMM, which was trained using representative examples of the team action. These examples were extracted from the observation flow using the teamwork scenario editor and labeled by hand. The positional observations were fed into the discretization module which extracted the team-oriented features.

The HMM was trained using the Baum-Welch expectation maximization algorithm for a predetermined number of states. We assumed that the emission probabilities of the HMM are described by a "mixture of Gaussians" multi-modal probability density distribution. The number of hidden states and the number of Gaussian components are parameters of the learning process.

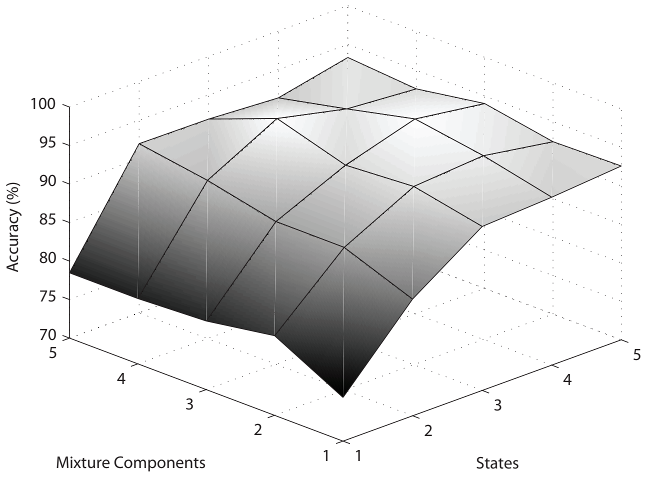

Figure 5: Mean accuracy of teamwork activity recognition accuracy. Hidden states and number of Gaussian mixture components range from 1 to 5.

Although one can try to analytically infer appropriate values for these parameters, this would require an in-depth knowledge of the team actions, as well as the number of variants which are labeled the same by human observers. The automatic tuning of these parameters can therefore save a lot of modeling effort.

We have optimized these parameters by repeating the learning process for every combination of the number of hidden states and Gaussian mixture components in the 1 to 5 range with $\epsilon = 2$. For each case we have tested the recognition accuracy using a 10-fold cross-validation procedure. The resulting accuracy values are shown in Figure 5. The optimal parameter configuration was found to be 4 hidden states and 3 Gaussian mixture components.

5.3 Performance Evaluation with Unknown Team-Organization

The experiments in this section measure the performance of teamwork activity recognition in applications where the role assignments are unknown. To simulate unknown role assignments in the dataset, we have randomly shuffled the agent observations.

Figure 6 shows the teamwork activity recognition performance, using 10-fold cross-validation. The observation sequences are distorted with Gaussian noise with the multiplier ϵ ranging from 0 to 10. We repeat the experiments with role assignment algorithms 1 and 2. For comparison, we performed a series of experiments under the assumption that the agents are always correctly assigned to the roles, as well as a series of experiments where the randomly shuffled agent observations were fed into the recognizer without role recognition.

From the experimental results in Figure 6 we can conclude that our system is able to recognize and assign roles to agents while still maintaining good teamwork recognition accuracy using both algorithms. *Algorithm 2* slightly outperforms *Algorithm 1* for this problem because the dataset used in this experiment consisted of teamwork activities performed by four agents where each agent always play a unique role in the team.

Table 5 illustrates the confusion matrix for teamwork recognition with *Algorithm 2* when $\epsilon = 2$. In this case the mean accuracy is 92.62% with standard deviation 5.53%.

As a point of reference, our previous work [10] achieved recognition rates of approximatively 82% using noise-free data and the

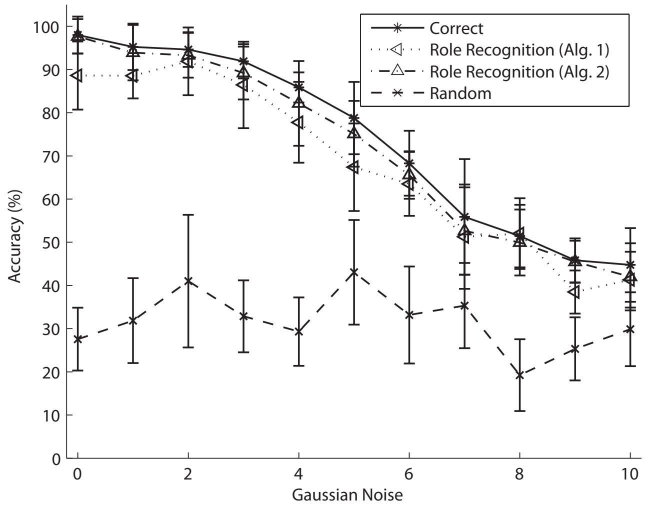

Figure 6: Mean recognition accuracy with error bars depicting standard deviation.

agent observations in the correct order. We conclude that the two improvements presented in this paper, the team oriented discrete feature extraction and the role recognition module we expect a system which achieves a better performance under less favorable input assumptions: noisy data and input in which the order of the agents was randomly shuffled.

6. CONCLUSIONS AND FUTURE WORK

In this paper we concentrated on improving the accuracy of systems for team activity recognition in embodied agents. First, we described a set of team oriented discrete features, selected such as to mirror the features used by humans in the recognition of team activity. Second, we described a role recognition module, which allows us to recognize the possible roles played by the agents before the team action is recognized. This allows us to represent the idealized team action in terms of roles (rather than agents), and allows the recognizer workflow to recognize the action even is the input does not contain information about the roles played by the agents (a situation frequently encountered in applications such as surveillance).

Acknowledgments

This work was partially funded by NSF Information and Intelligent Systems division under award 0712869.

The research was sponsored in part by the Army Research Laboratory and was accomplished under Cooperative Agreement Number W911NF-06-2-0041. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Laboratory or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation heron.

Appendix: A Workflow of Teamwork Activity Recognition with Hidden Markov Models

In the following we quickly summarize a standard workflow of teamwork activity recognition. The complexity and size of a HMM is determined by the number n of hidden states. Let us denote the hidden states of an HMM by $\omega = \{\omega_1, \ldots, \omega_n\}$ and a sequence of

Table 5: Confusion matrix depicting teamwork activity recognition performance with unknown team-organization.

			True							Precision
			a	b	c	d	e	f	g	(%)
Predicted	Traveling column	a	29	0	0	0	0	0	0	100.0
	Traveling line	b	0	26	0	0	0	0	0	100.0
	Traveling box	c	0	3	32	2	0	0	0	86.49
	Bounding overwatch	d	0	0	1	7	0	0	0	87.50
	Wedge	e	0	0	0	0	17	0	0	100.0
	Team split	f	0	0	0	0	0	13	2	86.67
	Team merge	g	0	1	0	0	0	2	13	81.25
	Recall (%)		100.00	86.67	96.97	77.78	100.00	86.67	86.67	

observation vectors with $\mathbf{V}^T = \{\mathbf{v_1}, \mathbf{v_2}, \dots, \mathbf{v_T}\}$. The parameters determining the HMM can now be defined as follows:

(a) Transition probabilities,
$\mathbf{A} = \{\alpha_{ij}\}$, where $\alpha_{ij} = P(\omega_j(t+1) \mid \omega_i(t))$

(b) Emission probabilities,
$\mathbf{B} = \{\beta_j(\mathbf{v})\}$, where $\beta_j(\mathbf{v}) = P(\mathbf{v} \mid \omega_j(t))$

(c) Initial probabilities,
$\boldsymbol{\pi} = \{\pi_i\}$, where $\pi_i = P(\omega_i(t))$

Thus, we can model a teamwork activity as $\boldsymbol{\theta} = \{\mathbf{A}, \mathbf{B}, \boldsymbol{\pi}\}$.

Estimating HMM Parameters. Estimating $\boldsymbol{\theta}$ from sequence observations is performed using the Baum-Welch expectation maximization (EM) algorithm [1]. It is important that $\boldsymbol{\theta}$ is carefully initialized. For this purpose we have employed the segmental k-means algorithm [6].

Teamwork Activity Classification. We represent a idealized team action with multiple, internal HMMs estimated with different number of hidden states. At the cost of increased processing time for parameter estimation and classification, this approach is more robust towards variations in activity complexity. The teamwork activity classification algorithm can be separated into three phases:

(a) The probability $P(\mathbf{V}^T \mid \boldsymbol{\theta})$ of the input sequence is calculated for all internal HMMs in each activity model using the forward evaluation algorithm [2].

(b) The internal HMM with maximum $P(\mathbf{V}^T \mid \boldsymbol{\theta})$ is selected to represent each activity model.

(c) Finally, the activity model with maximum $P(\mathbf{V}^T \mid \boldsymbol{\theta})$ is chosen as the final classification.

7. REFERENCES

[1] L. E. Baum, T. Petrie, G. Soules, and N. Weiss. A maximization technique occurring in the statistical analysis of probabilistic functions of markov chains. *Annals Mathematical Statistics*, 41(1):164–171, 1970.

[2] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification (2nd Edition)*. Wiley-Interscience, 2000.

[3] Z. Guessoum, J. P. Briot, S. Charpentier, O. Marin, and P. Sens. A fault-tolerant multi-agent framework. In *First International Conference on Autonomous Agents and Multiagent Systems (AAMAS-2002)*, pages 672–673, New York, NY, USA, 2002. ACM Press.

[4] G. Gutnik and G. Kaminka. Towards a formal approach to overhearing: Algorithms for conversation identification. In *Third International Conference on Autonomous Agents and Multiagent Systems (AAMAS-2004)*, pages 78–85, Washington, DC, USA, 2004. IEEE Computer Society.

[5] S. S. Intille and A. Bobick. Recognizing planned, multi-person action. *Computer Vision and Image Understanding*, 81(3):414–445, 2001.

[6] B. H. Juang and L. R. Rabiner. The segmental k-means algorithm for estimating parameters of hidden markov models. *IEEE Trans. Acoustics. Speech. and Signal Processing*, 38(9):1639–1641, 1990.

[7] G. A. Kaminka, D. V. Pynadath, and M. Tambe. Monitoring teams by overhearing: A multi-agent plan recognition approach. *Journal of Artificial Intelligence Research*, 17, 2002.

[8] F. Legras and C. Tessier. Lotto: group formation by overhearing in large teams. In *Second International Conference on Autonomous Agents and Multiagent Systems (AAMAS-2003)*, pages 425–432, 2003.

[9] X. Liu and C.-S. Chua. Multi-agent activity recognition using observation decomposed hidden markov model. In *Third International Conference on Computer Vision Systems ICVS*, volume 2626 of *Lecture Notes in Computer Science*, pages 247–256, Graz, Austria, 2003. Springer.

[10] L. J. Luotsinen, H. Fernlund, and L. Bölöni. Automatic annotation of team actions in observations of embodied agents. In *Sixth International Conference on Autonomous Agents and Multiagent Systems (AAMAS-2007)*, Honolulu, HI, USA, May 2007.

[11] M. V. N. Prasad, V. Lesser, and S. Lander. Learning organizational roles in a heterogeneous multi-agent system. *Proceedings of the Second International Conference on Multi-Agent Systems*, pages 291–298, January 1996.

[12] J. R. Quinlan. Simplifying decision trees. In *Knowledge Acquisition for Knowledge-Based Systems*, pages 239–252, London, 1988. Academic Press.

[13] S. Rossi and P. Busetta. Towards monitoring of group interactions and social roles via overhearing. In *Cooperative Information Agents VIII*, volume 3191, pages 47–61. Springer Berlin / Heidelberg, 2004.

[14] P. Scerri, D. Pynadath, N. Schurr, A. Farinelli, S. Gandhe, and M. Tambe. Team oriented programming and proxy agents: The next generation. In *Proceedings of 1st international workshop on Programming Multiagent Systems*, 2004.

[15] G. Sukthankar and K. Sycara. Simultaneous team assignment and behavior recognition from spatio-temporal agent traces. In *Proceedings of Twenty-First National Conference on Artificial Intelligence (AAAI-06)*, 2006.

[16] M. Tambe. Towards flexible teamwork. *Journal of Artificial Intelligence Research*, 7:83–124, 1997.

[17] D. L. Vail, M. M. Veloso, and J. D. Lafferty. Conditional random fields for activity recognition. In *Sixth International Conference on Autonomous Agents and Multiagent Systems (AAMAS-2007)*, Honolulu, HI, USA, May 2007.