Automatic Analysis of Embodied Team Actions

Brandyn White and Ladislau Bölöni
School of Electrical Engineering and Computer Science
University of Central Florida
Orlando, FL 32816–2450
Email: bwhite,lboloni@eecs.ucf.edu

Abstract

We describe a system which, building on previous team action recognition systems, performs a more in-depth analysis of an ongoing team action executed by a group of embodied agents. The system relies on team action states with human understandable semantics, estimates the current state and is able to make predictions or identify fringe cases such as incomplete or incorrectly executed team actions. The representation of the team action relies on a dynamic Bayesian network (DBN). We perform reasoning over the DBN using a sampling-importance-resampling particle filter. As a methodological illustration, we describe the process of model building for the bounding overwatch team action. We experimentally test our approach using data acquired from video recordings, and measure the system's ability to recognize a team action and to estimate the current state.

1 Introduction

Team action recognition is a field with important applications in training sport and military teams, automatic commentary systems, and surveillance. The pioneering work done by [Intille and Bobick, 2001], as well as [Sukthankar and Sycara, 2005; 2006] has made significant progress in the recognition of team actions. Our previous contributions have concentrated on the automatic learning of the team actions from representative examples [Luotsinen *et al.*, 2007] and on the extraction of specific features which approximate the way in which the team actions are perceived by human observers [Luotsinen and Bölöni, 2008].

For many applications, however, a simple recognition system might not be sufficient. We want our system to analyze the ongoing team action and extract information beyond its recognition. For instance, we want our system to be able to:

- identify the internal structure of the action and its internal phases (which may be repetitive)
- identify the current state of the execution of the action
- use the above to predict the likely evolution of the action
- continuously revise its belief about the likely nature of the action
- identify fringe cases such as incorrectly or partially executed actions (important in training)

In this paper we describe an approach which recognizes and analyzes team actions of embodied agents. The input to our system is a video recording (although the tracking of the agents is not a focus of our research). We start by identifying the teams and subteams in the scene, followed by the extraction of high level features. These have been chosen in such a way that they are semantically meaningful to a human operator and, as much as possible, match the features used by humans to identify the team actions. Examples of such high level features are "being under cover" or "being in sight of". We represent the team action using a dynamic Bayesian network (DBN) to represent a team action. We use a particle filter to perform probabilistic reasoning over the DBN.

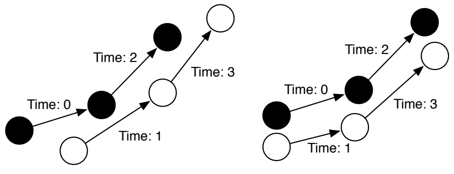
To illustrate our approach, throughout this paper we will use the bounding overwatch (BO) team action as a running example. BO is a well known maneuver, important in military operations and, despite being relatively simple, poses several challenges for the recognizer. It can not be recognized from static configurations; it requires dynamic analysis of movement patterns. In addition, bounding overwatch carries relatively heavy internal semantics: the goal of the maneuver is for the bounding subteams to provide cover for each other. This defines the relationship of the subteams to each other as well as to the environment (e.g., obstacles, enemy positions, line of sight).

We validate our approach through a series of experiments on recognizing the BO team action from recorded video data, and estimating the state of execution for the positive examples.

2 Related work

Related work in this area has primarily focused on the use of Hidden Markov Models (HMM) and other graphical models as a statistical framework on which inference can be performed.

[Intille and Bobick, 2001] used Bayesian networks to fuse multiple data sources and combine temporal information to produce an action likelihood for a given multi-agent event. Experiments were performed on American football play descriptions along with manually acquired play trajectories.



(a) (b)

Figure 1: Two examples of bounding overwatch with the time each team bounds listed on the transition. (a) Alternating bounds (b) Successive bounds

[Sukthankar and Sycara, 2005] used a random sample consensus (RANSAC) solution to fit observed spatial trajectories to those in a known model library by applying geometric transformations using Military Operations in Urban Terrain (MOUT) data. In [Sukthankar and Sycara, 2006] they used HMMs to model agent configurations over time.

[Liu and Chua, 2006] introduced an Observation Decomposed Hidden Markov Model (ODHMM) to allow for the use of an arbitrary number of agents in an HMM which requires a fixed length input.

[Luotsinen *et al.*, 2007] proposed automatically training the HMM probabilities using Baum-Welch and K-Means while experimenting on GPS data of military exercises. Extending this, role-based recognition and detailed feature discretization was used [Luotsinen and Bölöni, 2008].

[Hongeng *et al.*, 2004] proposed a multi-agent event recognition system using Bayesian networks to model actions such as standing and crouching.

One of the aspects of our work is the identification of the teams and subteams.

[Avrahami-Zilberbrand and Kaminka, 2007] introduce a representation called Dynamic Hierarchical Group Model to identify groups of agents with suspicious behavior, in settings where the behavior can be captured only when tracking agents with respect to a group, not as individuals.

[Takács *et al.*, 2007] adapt the spectral clustering method from image segmentation to identifying teams of agents. Part of the challenge is to find an affinity matrix which captures not only spatial proximity, but also the temporal aspects of actions as well as the participation of the agents in events.

3 Team action analysis: Bounding Overwatch

While it was demonstrated that a *recognizer* can be trained from a small number of representative examples [Luotsinen *et al.*, 2007], an *analyzer* of the type we are developing here requires a careful analysis of the team action. We are interested not only in the most distinguishing characteristics of the team action, but also in its variants, preparation stage, fringe conditions, and ways in which it can fail or transition to other team actions.

Bounding overwatch is a tactical movement formation used by the military when enemy contact is expected. The formation consists of two teams, one of which travels (i.e., bounds) in a pre-specified direction while the other protects (i.e., overwatches) the traveling team. After the bounding team has reached its destination, the teams switch roles. This process continues until the previously designated destination is reached.

We can distinguish two variants of bounding overwatch (see Fig. 1). In *successive bounds BO*, one team consistently leads the other team. In *alternate bounds BO*, the teams alternatively take the lead position.

The bounding team's objective is to travel quickly in the pre-determined travel direction, stay within protective cover of the overwatch team, and end in a position that provides natural cover.

The overwatch team's objective is to maintain a position that provides natural cover, visually track the bounding team, and watch for enemies.

Although BO is a relatively simple action, it presents several challenges to a recognizer or analyzer. First, there is no static configuration which would allow us to recognize the team action. Even for a recognizer taking into account the dynamic behavior of the agents, we need to consider a time interval of at least two bounds to be able to conclusively identify the team action as a BO. Short time sequences in a BO are indistinguishable from other team actions such as a team split or team merge. In certain phases of the BO, the segmentation of the teams can be difficult due to close proximity of the agents. There are two principal states, bounding and overwatching, with one team assigned to each at a given moment; however, this idealization fails to properly model the transient behavior observed in reality where both teams may be performing the same role for a short period of time. To model the transient cases it is necessary to add two states after each bounding state: both teams bounding and both teams overwatching. While the addition of transient states makes modeling the true observed behavior more natural, it doesn't remove the state detection complexity due to inherent ambiguities. An example of this is given that both teams are overwatching it isn't possible to determine which is next to bound. As a matter of convention, the overall state defining the behavior of both teams is labeled based on the team that has been bounding last if neither is bounding or longest if at least one team is (e.g., T0B implies T_0 is bounding and T_1 is overwatching). The state machine shown in Fig. 2 specifies both the overall states and the individual roles of each team in those states. The roles represent what the individual teams are doing during each state. An example is a team whose role is OO_{PB} is now overwatching after previously bounding and they are awaiting the other team to bound. Below is a brief description of each of the team roles.

B The team is bounding and their motion is directed toward a cover position located within protective cover of the overwatching team. Their direction of travel advances the team closer to the final destination.

O The team is overwatching, they are stationary, located at a cover position, and are guarding the bounding team.

$\mathbf{OO_{PB}}$ The team has finished bounding and is preparing to guard the other team as they start to bound.

$\mathbf{OO_{PO}}$ The team is preparing to bound as the other team has just finished bounding.

$\mathbf{BB_{PB}}$ The team is finishing bounding while the other team has started bounding.

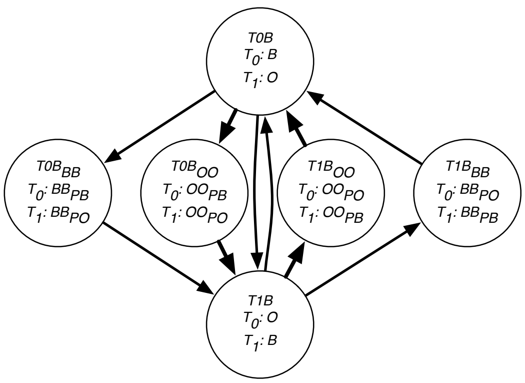


Figure 2: State diagram showing primary state transitions for both participating teams and individual roles within those states. All states have recurrent links which are omitted for clarity.

$\mathbf{BB_{PO}}$ The team has started bounding as the other team is finishing bounding.

4 Identification of the subteams

Bounding overwatch is defined in terms of the behavior of exactly two subteams; the representation of the action is based on teams, rather than the individual agents. Thus, the first step of data processing is the identification of the two subteams, by labeling the agents with the teams in which they participate.

A first insight is that while there are moments during the team action when the teams are clearly separated, this is not true for every moment in time. As shown in Fig. 1, it often occurs in successive bounding that the teams are in close proximity at the end of the trailing team's bound and are further apart at the end of the leading team's bound; in contrast, alternating bounds causes the teams to be furthest at the beginning and end of their bounds and closest at the middle.

To ensure that we capture sufficient samples where the teams are spatially separated, we perform the clustering over a window of time W that is at least the period between bounding cycles for a particular team. The team clustering problem can be formulated as trying to find the two team clusters T_0 and T_1 that minimize the spatial K-Means error (i.e., spatial variance) for both teams over all time-slices in the bounding period W

$$\sum V = \sum_{t=0}^{W} \sum_{i=0}^{1} \sum_{x_{j,t} \in T_i} (x_{j,t} - \mu_{i,t})^2 \qquad (1)$$

where $x_{j,t}$ is an agent's position at a given instant and $\mu_{i,t}$ is the a team's centroid at a given instant. For a small number of agents it is reasonable to try all possible clusterings, selecting the one that minimizes the $\sum V$; however, the number of clusters grow $O(2^n)$ in the number of agents. A solution to this problem is to solve the easier spatial K-Means clustering problem for each time-slice in W, and use the resulting set of clusterings as candidate solutions to the spatiotemporal problem. The motivation for this is that only one correct spatial clustering is necessary to solve the overall problem and the bounding period W ensures that if the team action is bounding overwatch, the straightforward case will be encountered (i.e., when the teams are separated). Lloyd's algorithm is used to solve the spatial K-Means problem [Lloyd, 1982] by initially selecting the most distant agents as cluster centers and updating the cluster centers based on the agents that are near them; this process is iterated until the clusters stabilize.

5 Feature Extraction

The inputs to the feature extraction process are the teams produced by the clustering process, each agent's 2-D world-plane position and heading, 2-D world-plane marker positions representing static natural cover (e.g., trees), and previously estimated states. The features should aid in the discrimination between the states specified in Fig. 2 and specification of bounding overwatch overall so that detection can be performed among other team events. Each feature must operate on the team level of abstraction to allow the underlying number of agents between sequences to be variable; furthermore, they should give similar high level semantic information about the team as used by humans in bounding overwatch identification. The prescribed goals for each team provided by the definition of bounding overwatch serve as a guide as to what properties are inherent to bounding overwatch, allowing us to avoiding features that are incidental to it.

TeamTraveling: Provides a very strong indicator of the bounding overwatch state as one team should be traveling while the other is stationary except during transient states. This is found by thresholding the team's velocity between frames.

TeamNearCover: Provided the team isn't traveling this is a strong indicator of an overwatching team, as their goal is to maintain a position near cover; however, if the team is traveling, this isn't a strong indicator as a bounding team may be incidentally passing a cover position. This is computed by finding the minimum distance from the team's centroid value and all cover positions.

TeamWatching: Provides a strong indicator that a team is overwatching if true and bounding if false. This is computed as the minimum angle difference between a team's agents and another team's centroid summed over a window of time. Intuitively, this value is low when an agent in a team is watching another team over a period of time. By taking this over a window of time we prevent the chance coincidence of the agents heading and its relative angle to another team.

TransientStateTimer: When this is true it is unlikely that the teams are in a transient state, as the threshold is set to be greater than the maximum expected time in a transient state. The duration of time that the current state has been active is computed online from the previous state information. While this does place a dependance on time, it only does so for the transient states; consequently, the major states of the algorithm are left independent of the execution time.

5.1 Feature Smoothing

As the features output discrete values obtained from continuous data, true values located near the threshold can cause oscillations due to the sensor noise causing the observed value to be on either side of the threshold over a short period of time. Moreover, if feature changes are to be used to generate observation events it is essential that the number of false transitions are reduced to an acceptable value. Since the feature levels (i.e., areas of constant feature value) are large in duration as compared to the feature edges, we will use median filtering over a window of time. For instance, the output from the **TeamTraveling** feature as a team starts moving can cause multiple false transitions as the velocity nears the threshold while only one true transition exists.

6 Modeling team actions with a Dynamic Bayesian Network

A Dynamic Bayesian Network (DBN) is a temporal probability model that consists of a semi-infinite collection of hidden state variables $\mathbf{X}_t$ and evidence variables $\mathbf{E}_t$ represented as a two-slice temporal Bayes' net (2TBN). Specification requires knowledge of the prior state probabilities $\mathbf{P}(\mathbf{X}_0)$, state transition probabilities $\mathbf{P}(\mathbf{X}_{t+1}|\mathbf{X}_t)$, and the sensor model $\mathbf{P}(\mathbf{E}_t|\mathbf{X}_t)$ [Murphy, 2002]. It is assumed that all modeled probability distributions are stationary (i.e., time invariant), time is represented as a discrete stochastic event, and that the process is first-order Markovian as enforced by the 2TBN structure.

A hidden Markov model (HMM) is a popular specialization of a DBN with a single discrete hidden node and one evidence node. Using an HMM as opposed to a more general DBN has the advantage of a simplified implementation; however, the reduction in expressivity forces the combination of otherwise independent hidden or evidence nodes into one 'meganode'. This process exponentially increases the number of probabilities that must be represented in the system resulting in slower inference, more training samples required to learn the probabilities, and larger storage requirements.

Our implementation uses a relatively simple DBN, which compared to the equivalent HMM decomposes the observations, but not the state. As we will see, however, even this relatively minor decomposition allows us to significantly reduce the number of probability values we need to acquire.

6.1 Event Generation

Given a DBN and set of sensor observations, two notable event generation methods arise: an event for every observation and an event for every change in observation (i.e., differing from the previous).

Generating an event for every observation causes the probability that the current state is maintained to depend on the sample frequency. For example, as the sampling frequency increases the probability of maintaining the current state also increases as there are only a constant number of transitions to different states in reality compared to the infinite number of effective transitions to the same state. This property is undesirable when multiple sampling rates are used or when the time a state is maintained doesn't follow a geometric distribution. As a practical matter, when the probability of maintaining a state approaches unity as the probability of changing state approaches zero the overall probabilities will incur increasing levels of computational error due to limited precision.

Generating an event for every change in observation causes the probability that a current state is maintained to become independent of the sample frequency provided that the features used change only when the state does. By making this assumption of the features, it allows the modeling of events irrespective of sample rate or the time of their execution. As the features diverge from this assumption, this method exhibits the same behavior as the previous; consequently, if the intended result is to have a model that has a reduced dependance on the sample rate, then it is necessary to use features that change values near true state transitions.

6.2 Proposed DBN

The states in Fig. 2 encode the individual roles of the teams, yet each role specifies only one sensor model which results in a symmetry between each of the team's sensor models. For example, when in state $T0B$ T_0 is bounding and T_1 is overwatching where in $T1B$ the roles switch; however, the observed behaviors between the bounding and overwatching teams are defined to be the same as the ordering of the teams is irrelevant. This property causes the probabilities between each of the team's sensor models to be the same when they are in the same role. As each team's behavior in bounding overwatch is specified by its current role, the team's sensor model only depends on that role; this property leads to a conditional independence between the team's sensor models. Moreover, as the transient state timer is defined in terms of how long the current state has been maintained it too is conditionally independent of each team's sensor model given the current state. These conditional independence assertions are expressed in Fig. 3

As seen in Fig. 3, the proposed DBN consists of three evidence nodes: one for each team's features and a transient state timer. The team features are, in general, dependent. An example of a dependance between the team features is whenever a team is stopped, they are more likely to be near a cover position as bounding overwatch is only performed when traveling through a region with a high likelihood of enemy contact. The number of probabilities that are needed to specify $\mathbf{P}(\mathbf{E}_t|\mathbf{X}_t)$ using this model are $2S$ for the timer and $S2^f$ for both of the teams where S is the number of unique states and f is the number of boolean team features used. As previously mentioned, both teams share the same sensor model due to symmetry between roles; consequently, we have to learn half of the team sensor model probabilities as would otherwise be required if only the conditional independence between them was taken into account. The number of boolean team features in our model is from one to three and the number of unique states is six (Fig. 2), resulting in the number of sensor model probabilities for the proposed DBN to be 24, 36, or 60. As a comparison, the equivalent HMM would have $S2^{1+2f}$ probabilities due to the inability to use the conditional independence between the sensor model nodes and the symmetry of

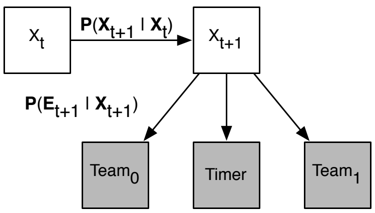


Figure 3: Proposed DBN for bounding overwatch state inference.

the sensor models for each team, resulting in 48, 192, or 768 sensor model probabilities.

6.3 Probability Computation

Similar to an HMM, our proposed DBN has three types of probabilities: state priors $\mathbf{P}(\mathbf{X}_0)$, state transition $\mathbf{P}(\mathbf{X}_{t+1}|\mathbf{X}_t)$, and sensor model $\mathbf{P}(\mathbf{E}_t|\mathbf{X}_t)$. While it may be possible to use expert knowledge to set every probability, an automated data driven approach allows for increased accuracy and overall flexibility. In total, there are 6 state prior probabilities, 36 state transition probabilities, and 24, 36, or 60 sensor model probabilities for 1, 2, or 3 boolean features respectively. To compute the probabilities automatically it is necessary to have a dataset of positive bounding overwatch sequences with annotated state information for every observation. State prior and transition probabilities can be computed immediately from the provided state level ground truth by creating histograms of all possible values and normalizing them to produce their probability. In computing the sensor model probabilities, if one histogram of all possible sensor observations is maintained for each team role (as opposed to each state) then two total samples are provided by each observation (i.e., one per team). This property doubles the number of samples available and it enforces the equivalence of the probabilities between the teams. This method of probability computation allows us to maintain our understanding of the underlying states as we have assigned them.

7 Particle Filtering

Particle filtering is a method of performing approximate inference on a DBN by maintaining a number of samples (i.e., particles) that each maintain a current state x_t which is initially sampled from the prior state probability distribution $\mathbf{P}(\mathbf{X}_t)$ and a weight ω. They are filtered from one state to the next by sampling the state transition probability $\mathbf{P}(\mathbf{X}_{t+1}|x_t)$ and weighting the particle based on the sensor model $P(e_{t+1}|x_{t+1})$. In an effort to avoid degenerate cases due to a majority of the particles residing in improbable states Sampling Importance Resampling (SIR) is used (Algorithm 1). The effective number of particles in SIR is computed on the normalized particle weight vector ω_t.

$$\hat{N}_t = \frac{1}{\sum_{i=1}^{N} \omega_{i,t}^2} \tag{2}$$

where t is the current frame and N is the number of particles.

Algorithm 1 Sampling Importance Resampling (SIR) particle filter with maximum overall probability and state probability output.

```
Inputs: $e_{t+1}$: current evidence, N: number of samples, $S_t$: previous sample N-vector, $\omega_t$: previous weight N-vector, $W_t$: previous overall weight N-vector, $\mathbf{P}(\mathbf{X}_0)$: prior, $\mathbf{P}(\mathbf{X}_{t+1}|\mathbf{X}_t)$: transition model, $\mathbf{P}(\mathbf{E}_{t+1}|\mathbf{X}_{t+1})$: sensor model, $T_e$: minimum effective particle threshold
Outputs: $S_{t+1}$: next sample N-vector, $\omega_{t+1}$: next weight N-vector, $W_{t+1}$: next overall weight N-vector, $\mathbf{P}(\mathbf{X}_{t+1}|e_{1:t+1})$: state probabilities given evidence
if $S_t == \emptyset$ then
    $S_t \leftarrow$ sample from $\mathbf{P}(\mathbf{X}_0)$
    $\omega_t \leftarrow$ N-vector of ones
    $W_t \leftarrow$ for each $s_t$ in $S_t$: $P(s_t)$
end if
for $i = 1$ to N do
    $S_{t+1}[i] \leftarrow$ sample from $\mathbf{P}(\mathbf{X}_{t+1}|S_t[i])$
    $\omega_{t+1}[i] \leftarrow \omega_t[i] \times P(e_{t+1}|S_{t+1}[i])$
    $W_{t+1}[i] \leftarrow W_t[i] \times P(e_{t+1}|S_{t+1}[i]) \times P(S_{t+1}[i]|S_t[i])$
end for
$\omega_{t+1} \leftarrow$ normalize-weights($\omega_{t+1}$)
$\mathbf{P}(\mathbf{X}_{t+1}|e_{1:t+1}) \leftarrow$ state-probabilities($S_{t+1}, \omega_{t+1}$)
if effective-num-particles($\omega_{t+1}$) $< T_e$ then
    Local: $rstates$: holds the resampled states
    $rstates \leftarrow$ weighted-sample($S_{t+1}, \omega_{t+1}, N$)
    $W_{t+1} \leftarrow$ max-state-prob($rstates, S_{t+1}, W_{t+1}$, N)
    $S_{t+1} \leftarrow rstates$
    $\omega_{t+1} \leftarrow$ N-vector of ones
end if
```

7.1 State Estimation

The particles represent a potential path through the DBN from initialization up some time period t provided the evidence $e_{1:t}$. The state probability can be found by finding the proportion of the particles in each state at time t by defining $N(x_t|e_{1:t})$ as the number of particles in state x_t

$$P(x_t|e_{1:t}) \approx \frac{N(x_t|e_{1:t})}{N} \tag{3}$$

where N is the number of particles in total and as $N \rightarrow \infty$ then this converges to $P(x_t|e_{1:t})$.

7.2 Overall BO Probability

In order to perform identification of bounding overwatch among other common team actions, it is necessary to produce a measure of confidence from the system that expresses the likelihood that the sequence is bounding overwatch. In Algorithm 1, we compute the overall probability vector W_{t+1} for each particle. For each particle j this value represents its probability through the filtering process.

$$W_{t,j} = P(x_1) \prod_{i=1}^{t} P(x_{i+1}|x_i) P(e_{i+1}|x_{i+1}) \tag{4}$$

The overall BO probability is found as

$$\max(W_t)^{-(t+1)} \tag{5}$$

where t is the number of observations. By choosing the maximum value for this, we are finding the likelihood of the most probable explanation for the observations given our model. Since resampling the particles erases their state history, we can set it to that of the most probable particle currently in that state (Algorithm 7.3). As the probability of a particle transitioning from one state to another is only dependent on the current state (i.e., the markov property), it ensures that selecting the maximum overall probability for each state will lead to the true maximum overall probability. Taking the maximum overall probability to the $-(t+1)$ power is equivalent to taking the geometric mean of the most likely path's probabilities of each time period including the prior, resulting in a value that is invariant to the number of observations.

7.3 Temporal Segmentation

As the events are not assumed to be perfectly segmented, a method is needed to ensure that transient observations on either side of the true event have as little of an impact as possible. Without temporal segmentation, transient data can cause the probabilities to go toward zero which will negatively impact the results from then on. The proposed solution is to reinitialize the particle filter whenever the overall BO probability has gone below a predetermined limit near zero. This allows for arbitrary placement of the event in the observation sequence and the particle filter will continually reinitialize until observations are encountered that fit the proposed model. With this addition we are now able to generate a list of subsequences present in the original sequence and produce a confidence measure that each is bounding overwatch.

Now that there may be significantly less observations considered than present in the sequence, there needs to be a way to ensure that the overall BO probability is taken from a significant sample size. One solution would be to only consider the overall probability to be valid if it has been computed with a minimum number of observations; however, this method is tightly coupled with event duration which our system up to this point has been largely invariant to. As an example, a human observer of bounding overwatch needs less observations from teams performing the event quickly (i.e., they are spread across the action) than teams performing it slowly due to the need to observe the entire process to ensure that the common ambiguities (e.g., Team Split) are eliminated. The proposed solution is to define the appropriate minimum number of samples in terms of BO state transitions, thus ensuring that the data is from more than one observed state, that the system will remain insensitive to the execution time of the event, and that common ambiguities can be confidently eliminated from consideration.

8 Data collection

The dataset used consists of various team actions captured on video taken from two scenes and several camera angles. The video has been manually annotated to capture both the agent's position and heading. The position is specified by one point on the ground plane between the agent's feet while the heading is specified by placing an additional point on the ground plane in the direction the agent is facing. For each camera position, a 3x3 homography matrix H is computed between the image plane and the metric ground plane by placing targets on the ground plane and measuring their relative locations. The 4-point homography algorithm and RANSAC [Hartley and Zisserman, 2003] were used to compute the homography and reject measurement outliers respectively. Redundant measurements were made to allow for validation of this process with a maximum observed error of ±6 inches. To find the 3x1 homogeneous world position x' use

$$x' = Hx \qquad (6)$$

where x is a 3x1 homogeneous image position and H is 3x3 image to world coordinate projective homography. To compute the agent's heading (i.e., relative to the world plane x-axis), both the position and heading points are warped onto the ground plane coordinate system using (6), the vector difference is taken between the 2x1 Euclidean (i.e., inhomogeneous) heading and position points, and finally the heading angle is computed as $\arctan(y/x)$. As the distance between the position and heading points is unused, they are spread out as far as possible to increase the accuracy of each agent's heading.

Algorithm 2 max-state-prob(R, S, W, N)

```
Inputs: N: number of samples, R: resampled state N-vector,
  S: current state N-vector, W: overall weight N-vector
  Local: $out$: maximum overall weight N-vector that corresponds to R
  for $i = 1$ to $N$ do
    Local: $x_0$: temp max weight index whose state is $S[i]$
    $x_0 \leftarrow \{x — \forall y : W[y] \leq W[x] \wedge S[i] = S[x]\}$
    $out[i] \leftarrow W[x_0]$
  end for
  return $out$
```

Act#	Activity	Sequences	Sample Time (sec.)
1	Random (w/o cover)	6	343
2	Random (w/ cover)	5	297
3	Walking Line	4	234
4	Meeting	3	192
5	Following	2	135
6	VIP Guarding	2	133
7	Both Travel/Watch	7	129
B.O. Positive			
8	Alternating Bounds	11	286
9	Successive Bounds	10	328
10	No Progress	2	101
11	Off Cover	1	34

Figure 4: Number of sequences and total sample time for each team action in the dataset.

8.1 Recorded team actions

As shown in Fig. 4, the dataset features positive variants of the bounding overwatch team action along with several other common team actions to serve as negative classification examples. The specified team action for each video is located in the middle of the sequence with minor setup time before

Figure 5: Sample pictures from the dataset used. The markers on the ground are used as bounding overwatch cover positions as well as for homography computation.

and after. By leaving a few seconds of neutral movement on either side of the action it ensures that the algorithm being evaluated is able to perform temporal segmentation of the event and that it is taking into account the entire sequence as opposed to some initial period. Below is a brief description of each team action.

Random: The agents move around without purpose.

Walking Line: The agents form a tight vertical line and move around the scene, finely mimicking the leading agent's path.

Meeting: The agents come together at a central point and stop for a period of time.

Following: One agent is followed from a distance by the other agents, coarsely mimicking the leading agent's path.

VIP Guarding: One agent is followed closely by guards from slightly behind and to the side.

Both Travel/Watch: Similar to bounding overwatch but both teams travel and watch at the same time, thus eliminating the protection bounding overwatch provides.

B.O. Alternating Bounds: Bounding overwatch performed using alternating bounds (i.e., the bounding team passes the overwatching team). Bounding is terminated at cover positions as denoted by markers on the ground plane.

B.O. Successive Bounds: Bounding overwatch performed using successive bounds (i.e., one team is consistently in front, regardless if they are bounding or overwatching). Bounding is terminated at cover positions as denoted by markers on the ground plane.

B.O. No Progress: Bounding overwatch as far as state transitions are concerned; however, this is counter-productive in that motion is not directed in an overall direction (e.g., returning to previous locations). Combinations of alternating and successive are not defined as no overall travel direction is defined.

B.O. Off Cover: Bounding overwatch performed while avoiding cover positions.

9 Results

9.1 Team action recognition

The first set of results concern the use of the system as a recognizer. We consider the system as a binary classifier separating the positive examples of BO in the data set from the negative ones. Naturally, this binary classifier can be used as a building block for multi-class classifiers

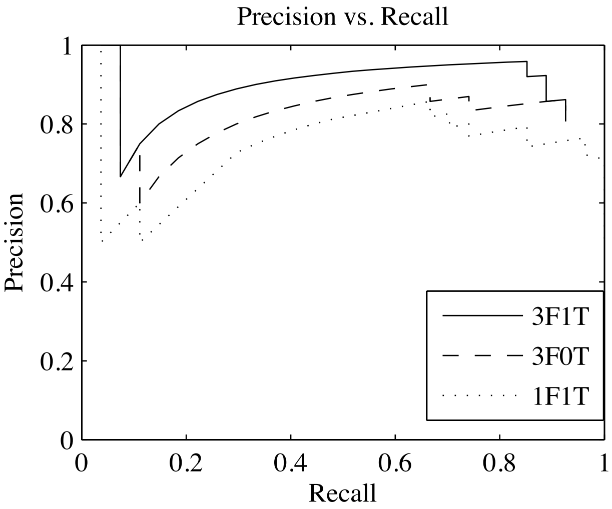


Figure 6: Bounding overwatch classification results using all three team features and the timer ($3F1T$), all three team features without the timer ($3F0T$), and one team feature (**TeamTraveling**) and the timer ($1F1T$).

Fig. 6 shows a precision-recall graph of the bounding overwatch classification of the dataset. Using **TeamTraveling**, **TeamNearCover**, and **TeamWatching** as the team features along with the **TransientStateTimer** achieves the best performance over all operating values with a maximum F1 measure (i.e., harmonic mean of recall and precision) of .91 with an associated precision of 92% while achieving an 89% recall. Using the same three team features and removing the **TransientStateTimer** performs the next best over all operating values with a maximum F1 measure of .89 with an associated precision of 86% and a recall of 92%. By only using **TeamTraveling** and the **TransientStateTimer** the results are fair considering that it uses significantly less information while receiving a maximum F1 measure of .83 with an associated precision of 75% and a recall of 92%. As seen in Fig. 7, a few of the sequence #7 (**Both Travel/Watch**) videos provided the greatest classification challenge because as both teams moved together, the system would predict that they are bounding as a transient state but not for a long enough duration to be penalized by the transient event timer.

9.2 State estimation

Our second set of results concern the system as an analyzer of the bounding overwatch action. Once the action has been identified as bounding overwatch, the challenge is to estimate the state of the system in the state diagram shown in Fig. 2. As a note, once the state is estimated, the probabilities of this graph can be used for state prediction.

Fig. 8 shows the state estimation results performed on the positive bounding overwatch samples present in the dataset. This was performed using **TeamTraveling**, **TeamNearCover**, and **TeamWatching** as the team features along with the **TransientStateTimer**. The highest number of errors occurred when the true state was $T0B_{OO}$ as it was confused with $T1B_{OO}$ due to the ambiguous sensor values between them. When both teams are overwatching it is likely that they are both stopped, near targets, and facing away from each

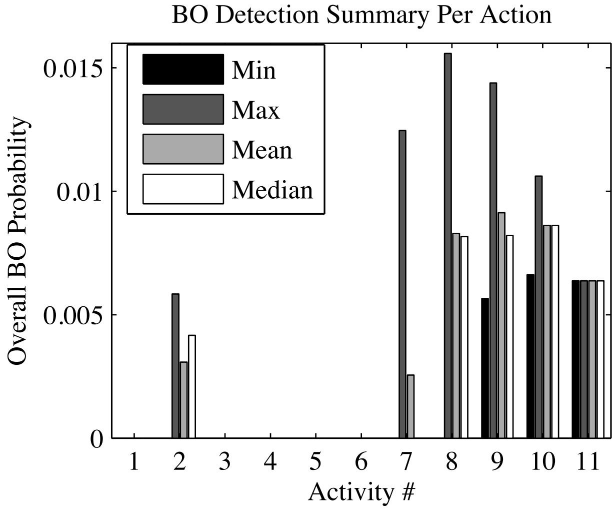


Figure 7: Bounding overwatch classification results using all three team features and the timer ($3F1T$) shown for each activity as specified in Fig. 4. This includes both negatives [1-7] and positives [8-11]. The majority of the negative sequences generated a zero probability due to behavior not witnessed in bounding overwatch. These are the maximum probabilities from each sequence that also met the minimum number of BO state transitions as specified in Section 7.3.

GT/Pred.	$T0B$	$T0B_{BB}$	$T0B_{OO}$	$T1B$	$T1B_{BB}$	$T1B_{OO}$
$T0B$	278	1	0	0	0	0
$T0B_{BB}$	1	42	0	0	3	0
$T0B_{OO}$	0	0	84	1	0	4
$T1B$	1	1	1	321	0	3
$T1B_{BB}$	1	0	0	0	38	0
$T1B_{OO}$	3	0	1	0	0	76

Figure 8: Confusion matrix between bounding overwatch states as specified in Fig. 2. While the team numbering is irrelevant, it is important to have both sets of states so that errors occurring where the predicted state is the complement of the true state can be expressed.

other; consequently, the only way to correctly detect this state is to use previous state information which occurred the majority of the time. The least ambiguous combination of states is $T0B$ and $T1B$ which received nearly zero confusions due to the distinct differences between the teams in this situation.

10 Conclusion

In this paper we described a system which analyzes an ongoing team action, identifies its current state and has the ability to make predictions and identify fringe cases (incomplete or incorrectly executed actions). We have used the bounding overwatch team action, with its successive and alternative bounds variants, to describe the proposed method. Our approach relies on representing the team action with a DBN, which was found to simplify the model-building, by reducing the number of probabilities which need to be considered compared with an equivalent HMM. We performed reasoning over the DBN using a SIR particle filter.

Acknowledgments

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

References

[Avrahami-Zilberbrand and Kaminka, 2007] D. Avrahami-Zilberbrand and G. Kaminka. Towards dynamic tracking of multi-agents teams: An initial report. In *Proceedings of Workshop on Plan, Activity, and Intent Recognition (PAIR 2007)*, 2007.

[Hartley and Zisserman, 2003] R Hartley and A Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press: Cambridge, UK, 2003.

[Hongeng *et al.*, 2004] S Hongeng, R Nevatia, and F Bremond. Video-based event recognition: activity representation and probabilistic recognition methods. *Computer Vision and Image Understanding*, 96(2):129–162, 2004.

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

[Liu and Chua, 2006] Xiaohui Liu and Chin-Seng Chua. Multi-agent activity recognition using observation decomposedhidden markov models. *Image and Vision Computing*, 24(2):166 – 175, 2006.

[Lloyd, 1982] S Lloyd. Least squares quantization in PCM. *Information Theory*, 28(2):129–137, 1982.

[Luotsinen and Bölöni, 2008] L.J. Luotsinen and L. Bölöni. Role-based teamwork activity recognition in observations of embodied agent actions. In *The Seventh Intl. Joint Conf. on Autonomous Agents and Multi-Agent Systems (AAMAS 08)*, pages 567–574, 2008.

[Luotsinen *et al.*, 2007] L. J. Luotsinen, H. Fernlund, and L. Bölöni. Automatic annotation of team actions in observations of embodied agents. In *The Sixth Intl. Joint Conf. on Autonomous Agents and Multi-Agent Systems (AAMAS 07)*, pages 32–34, 2007.

[Murphy, 2002] K Murphy. *Dynamic Bayesian Networks: Representation, Inference and Learning*. PhD thesis, UC Berkeley, Computer Science Division, Jan 2002.

[Sukthankar and Sycara, 2005] Gita Sukthankar and Katia Sycara. Identifying physical team behaviors from spatial relationships. In *Proceedings of 2005 Conference on Behavior Representation in Modeling and Simulation (BRIMS)*, pages 638–645, 2005.

[Sukthankar and Sycara, 2006] Gita Sukthankar and Katia Sycara. Robust recognition of physical team behaviors using spatio-temporal models. In *Proceedings of Fifth International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, pages 638–645, 2006.

[Takács *et al.*, 2007] B. Takács, S. Butler, and Y. Demiris. Multi-agent behaviour segmentation via spectral clustering. In *Proceedings of Workshop on Plan, Activity, and Intent Recognition (PAIR 2007)*, 2007.