Activity Recognition for Physically-Embodied Agent Teams

Gita Reese Sukthankar

Oct 5, 2005
CMU-RI-05-44

Robotics Institute
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

Thesis Committee:
Katia Sycara (chair)
Illah Nourbakhsh
Paul Scerri
Milind Tambe (external)

Keywords: multi-agent plan recognition, activity inferencing, teamwork, spatial reasoning

Abstract

This thesis focuses on the problem of activity recognition for physically-embodied agent teams. We define team activity recognition as the process of identifying team behaviors from traces of agents' positions and orientations as they evolve over time; the goal is to completely annotate agent traces with: 1) the correct sequence of low-level actions performed by each agent; 2) an assignment of agents to teams and subteams; 3) the set of team plans consistent with the observed sequence. Activity traces are gathered from teams of humans or agents performing military tasks in urban environments. Team behavior annotations can be used for a wide variety of applications including virtual training environments, visual monitoring systems, and commentator agents.

For many physical domains, coordinated team behaviors create distinctive spatio-temporal patterns that can be used to identify low-level action sequences; we demonstrate that this can be done in a way that is robust to spatial variations in the environment and human deviations during behavior execution. This thesis addresses the novel problem of *agent-to-team* assignment for team tasks where team composition, the mapping of agents into teams, changes over time; this allows the analysis of more complicated tasks in which agents must periodically divide into subteams. To do this, we introduce a new algorithm, Simultaneous Team Assignment and Behavior Recognition (STABR), that generates low-level action annotations from spatio-temporal agent traces. Finally, we extend methods in symbolic plan recognition to exploit both temporal constraints between behaviors and agent role constraints in team plans to reduce the number of state history hypotheses that our system must consider.

Contents

Chapter 1

Introduction

Often when a task is too complicated to be performed by an individual agent, it can be achieved through the coordinated efforts of a team of agents over a period of time. To analyze the performance of these multi-agent tasks, we need to extend existing behavior recognition formalisms to accommodate team behaviors. In dynamic domains, the composition of the team can change over time as teams split into subteams, subteams merge with one another, and uncommitted agents join or leave teams. Moreover, in a scenario where there are a large number of uninvolved agents, it can be difficult to determine which agents are engaged in team behavior without prior knowledge of the agents' intentions (imagine a convoy or funeral procession driving on a crowded city street).

The proposed thesis will explore the problem of team behavior recognition for physically-embodied agents. In physical domains (military, athletic, or robotic), team behaviors often have an observable spatio-temporal structure, defined by the relative physical positions of team members and their relation to static landmarks; we suggest that this structure, along with role and temporal constraints defined by a team plan library, can be exploited to perform behavior recognition on traces of agent activity over time, even in the presence of uninvolved agents. The desired output of this system is a complete annotation of the agents' traces including: 1) the correct sequence of low-level actions performed by each agent; 2) an assignment of agents to teams and subteams; 3) the set of team plans consistent with the observed sequence.

Figure 1 shows some examples of applications of that could benefit from the inclusion of team behavior recognition: team training environments (Figure 1a), sports commentator agents (Figure 1b), and visual surveillance systems (Figure 1c). The proposed thesis focuses on team behavior recognition for military domains such as MOUT (Military Operations in Urban Terrain), but team behavior recognition has also been applied to human athletic domains, such as basketball [18] and American football [16], as well as robotic domains such as Robocup coaching [30] and commentating [26].

This document is structured as follows: Chapter 2 gives some background on the MOUT domain; Chapter 3 provides an overview of related work in team behavior

Figure 1.1: Potential applications for team behavior recognition. a) Team training environments such as Clemson University's MOUT training facility at 263rd Army National Guard, Air and Missile Defense Command site. Teams of trainees learn the room clearing task in a "shoothouse" consisting of 6 rooms and interconnecting hallways. Trainees are instrumented with customized weapons that wirelessly transmit position and orientation. Supplementing the system with team behavior recognition is a prerequisite for automatically generating critiques of the trainees' performance. b) Sports commentator agents (agents that process game logs and produce text summaries of game action). This screenshot is from the ISAAC system [26] for Robocup commentary. Adding detailed team behavior recognition to such a system could enable the addition of more detailed play-by-play descriptions to the text summaries. c) Visual monitoring systems such as VSAM [9]. Future surveillance systems could benefit from stronger spatial models of team behavior to analyze human interactions.

recognition; Chapter 4 summarizes our completed work; Chapter 5 presents an example scenario for our system, outlines the proposed plan of work, timeline, and concludes with thesis contributions.

Chapter 2

The MOUT Domain

Many of the experiments in this document use behaviors, maps, and scenarios derived from human soldiers performing MOUT (Military Operations in Urban Terrain) tasks as described in [27]. In MOUT scenarios, platoons of soldiers attempt to achieve strategic objectives, such as clearing buildings, escorting convoys, and attacking enemy positions, within a cluttered, hazardous urban environment. The commanding officers must react to new threats in the environment and changes in spatial layout (e.g., blocked roads, booby-trapped zones) without direct guidance from the chain of command. It is often unclear whether observed people in the combat zone are civilians or enemy snipers. Although deliberative planning is required to systematically clear large areas of urban terrain, soldiers must execute reactive behaviors in the face of unexpected threats such as snipers and booby-traps.

The MOUT domain poses several significant teamwork challenges. Adversarial team planning is required when soldiers coordinate to outmaneuver and flank an enemy in an covered position. Successful teams cooperate to assist team members in trouble, such as wounded soldiers or teammates pinned down by enemy fire. Some team tasks, such as moving in formation, demand tightly-coupled physical coordination whereas other team tasks (e.g., securing a defensive perimeter) only require loose coordination. Splitting into sub-teams is commonly required to achieve goals; even formation movement is usually accomplished by dividing into smaller groups.

Spatial environmental features (buildings, intersections, doorways) are important features that influence MOUT planning [5], yet unlike team sports which are played on a field that can be simply characterized by regions (e.g. goal, yard lines, end zone), MOUT spatial landmarks are a complex combination of features. According to the cognitive task analysis of the building clearing task [24], spatial cues dominate all of the task-focused decision requirements of building clearing MOUT operations. The six *task-focused* decision requirements include: 1) securing the perimeter; 2) approaching the building; 3) entering the building; 4) clearing the building; 5) maintaining and evaluating security; 6) evacuating the building. For these tasks, features such as proximity to other buildings, opportunities for cover, open spaces, windows in the building, street layout, fortifications, height of buildings, locations of stairways,

obstacles, potential booby-traps, and doors, are critical cues that human decision-makers use to determine how to achieve the building clearing task. Dynamic spatial features such as civilian activity and presumed enemy location also factor into several stages of the decision making process. Out of the five *task-independent* decision requirements, spatial cues are used for three of them: 1) maintain the enemy's perspective; 2) maintain big picture and situation awareness; 3) project into the future. For these three decision requirements, important spatial cues include: location of hallways, stairwells, general building layout, teammates' locations, and last known enemy positions.

Although some aspects of MOUT planning are highly structured, commanders are encouraged to be sufficiently unpredictable to frustrate potential enemies. The cognitive task analysis for building clearing [24] offers suggestions such as "Refrain from establishing patterns the enemy could learn" and "Use creative thinking to bypass obstacles". This suggests that expert MOUT teams demonstrate variability in how they approach buildings with similar layouts, perhaps approaching from different vantage points or clearing rooms in different orders. However, expert teams exhibit coherent movement patterns. In Jirsa's study on MOUT team coordination dynamics [17], he hypothesizes that expert team performance is limited to a low-dimensional subspace. He suggests that the degree of confinement to the experts' subspace is a measure of team performance and coherence.

In summary, the MOUT domain offers the following interesting challenges to team behavior recognition:

1. The behavior recognition algorithm must be able to handle deliberative, hierarchical behaviors, as well as reactive behaviors. Our previous work on human behavior recognition [36] focused only on reactive behaviors.

2. The spatial landmarks that are important cues for MOUT behavior recognition cannot be simply characterized by static regions in a playing field, as has been done in other work [16, 28] on team behavior recognition.

3. Team tasks frequently require the splitting and merging into subteams, making the agent-to-team assignment problem more challenging.

4. Complete MOUT scenarios, including a soldier platoon, enemy snipers, and civilians, can feature large numbers of agents (50–100) cooperating in multiple team groupings; this is significantly greater than the number of agents addressed in multi-agent plan recognition frameworks [31] that only address coordination between pairs of agents.

Chapter 3

Related Work

This section gives a detailed description of related work in the area of team behavior recognition, along with an overview of research specific to the MOUT domain.

3.1 Team Behavior Recognition

Previous work on team behavior recognition has focused primarily on athletic domains, including American football [16], basketball [6, 18], and Robocup soccer simulations [26, 28–30]. Recognition for military air-combat scenarios has been examined in the context of event tracking [38] and teammate monitoring [19]. Finally, a general framework for multi-agent plan recognition (Hierarchical Multiagent Markov Processes) [31] has been demonstrated for a single pair of humans.

The first part of Intille and Bobick's work [15] on football game annotation addresses the low-level problem of extracting accurate player trajectories from video sequences that include substantial camera movement (panning and zooming). Using knowledge of the football field geometry and markings, they warp the original distorted image sequence into a rectified one with the appearance of a stationary camera and background. Then, by exploiting closed-world knowledge of the objects in the scene, they create context-specific feature templates for tracking individual players. Play recognition [16] is performed on player trajectories using belief networks both to recognize agent actions from visual evidence (e.g., catching a pass) and to determine the temporal relations between actions (e.g. before, after, around). Jug *et al.* [18] used a similar framework for basketball analysis. The main limitation of their work is the requirement that the playing field regions, fiducial markings, and players be precisely known in advance.

Unlike Intille and Bobick, Riley does not attempt to produce annotated traces with sequences of Robocup team behaviors. Instead he extracts specific information, such as home areas [30], opponent positions during set-plays [29], and adversarial models [28], from logs of Robocup simulation league games to be used by a coach agent advising a team. For instance, information about opponent agent home areas

are used as triggers for coaching advice and for doing "formation-based marking", in which different team members are assigned to track members of the opposing team.[1] Formations are generated from past logs in two phases: 1) fitting a minimal rectangle that explains the majority of each agent's past positions; 2) modeling pair-wise distance correlations between agents by shifting the minimal rectangles. Commentating from post-game log analysis has been demonstrated in the Robocup domain by the ISAAC system [26] using a combination of data mining and inductive learning. However, the emphasis of that system was to produce a high-level summary of game action, suitable for natural language summaries or debugging general team problems, rather than detailed play recognition of each team's actions.

Tambe and Rosenbloom [39] published one of the earliest explorations of online plan recognition vs. a single opponent in a dynamic, adversarial domain (Tac-Air SOAR). Tambe later extended the plan recognition algorithm, RESC, for tracking team behaviors. RESC (Real-time Situated Commitments) is a model-tracing tracking system in which the tracker executes a model of the trackee and matches the model's predictions against the known observations; rather than maintaining a large hypothesis set, RESC commits to a single execution path and invokes a repair-mechanism if commitment causes tracker error. RESC$_{team}$ executes a team model and invokes a minimal cost repair mechanism if the operator hierarchy selection or the role assignment causes match failure and includes extensions to handle subteam formation and role assignments. RESC$_{team}$ assumes that the subteams are either known in advance or detected solely based on agents' proximity to each other; although this assumption is often valid, some teams are spatially separated, in spite of sharing a common purpose (e.g., perimeter guarding in the MOUT domain).

Unlike other groups, Kaminka and Tambe's work [19] on plan recognition for socially attentive monitoring does not treat teams as automatically sharing a single plan. Due to message failure and sensor error, team members' plans might diverge, in spite of sharing the same top level goal. The focus of their research on socially-attentive monitoring is the use of plan recognition by teammates to determine when other team members have inconsistent beliefs.

Bui [7] has developed a framework for online probabilistic plan recognition (the Abstract Hidden Markov Memory Model) which he has applied to activity inferencing problems; unlike a standard Hidden Markov Model, the AHMEM includes a memory variable for tracking the level of the plan tree. Hierarchical Multiagent Markov Processes [31] is an extension of Bui's model to handle simple multi-agent coordination problems by adding joint agent policies and termination nodes. They investigate two different joint policy termination mechanisms: 1) terminate the joint policy when either agent terminates its local policy or 2) terminate the joint policy after both agents' terminate their local policies. It is unclear whether this type of simple coordination mechanism can scale to handle the complexity of larger teams with multiple

[1] Riley uses the term "formation" to specify a grouping of home areas, regions of the field where an agent can generally be found, rather than designating a temporal ordering of precise locations, which is this thesis' definition of MOUT formations.

subteams.

3.2 The MOUT Domain

Most of the work in the MOUT domain has tackled the problem of plan generation for computer-generated forces (CGFs) rather than the inverse problem of plan recognition. Pearson and Laird [23] created a diagrammatic tool for authoring exemplars of spatial plans which their system later generalized into SOAR productions to be incorporated into MOUT bots; Best and Lebiere [5] created a spatial language for authoring plans to be executed in ACT-R. The primary goal of these systems was to create easily-authored and reusable spatial representations that MOUT subject matter experts could use to program cognitive models. Previously, we developed methods [33, 34] for incorporating information about human physical capabilities, measured from motion capture data, into MOUT plan generation; the emphasis of that work was on generating plausible path plans given the particular abilities of a human soldier.

Several MOUT training systems have been developed to teach human soldiers the building clearing task. The Collaborative Warrior tutor, a prototype model-tracing tutoring system for MOUT described in [21], included both plan generation for CGFs and plan recognition of the trainee's actions. This system used very simple spatial models to evaluate the correctness of the trainee's plan (e.g. whether the trainee failed to clear rooms of enemy soldiers) and no recognition results are reported. Fowlkes *et al.* [12] developed a haptic-enhanced team training environment to train human subjects to move in a back-to-back formation with a simulated soldier; their study examined how haptics could improve acquisition of movement and communication skills. Recently, Hoover *et al.* [14] constructed a full-scale MOUT training environment at the 263rd Army National Guard, Air and Missile Defense Command site. Teams of trainees learn the room clearing task in a "shoothouse" consisting of 6 rooms and interconnecting hallways, instrumented with 36 cameras.

This chapter has reviewed the related research in both team behavior recognition and MOUT training. The next chapter describes our own research-to-date.

Chapter 4

Completed Work

This section summarizes our completed work in the area of behavior recognition:

- human behavior recognition from motion and environmental features [36];
- team formation recognition from overhead static snapshots [37];
- two-person team behavior recognition from Unreal Tournament traces [35].

This thesis proposes to extend some of the concepts developed in these systems into a unified model of team behavior recognition from spatio-temporal activity traces.

4.1 Cost Minimization for Behavior Recognition

Using full-body motion capture data acquired from human subjects, our system recognizes the behaviors performed by a human subject from a set of military maneuvers. Low-level motion classification is performed using support vector machines (SVMs); output from the classifier is used as an input feature for the behavior recognizer. Given the dynamic and highly reactive nature of the domain, our system must handle behavior sequences that are frequently interrupted and often interleaved. To recognize such behavior sequences, we employ dynamic programming in conjunction with a behavior transition cost function to efficiently select the most parsimonious explanation for the human's actions.

4.1.1 Representation

Our representation for single soldier behaviors in the MOUT domain includes physical actions, environmental features, states, and state transitions. Behaviors are described as directed acyclic graphs as shown in Figure 4.1.

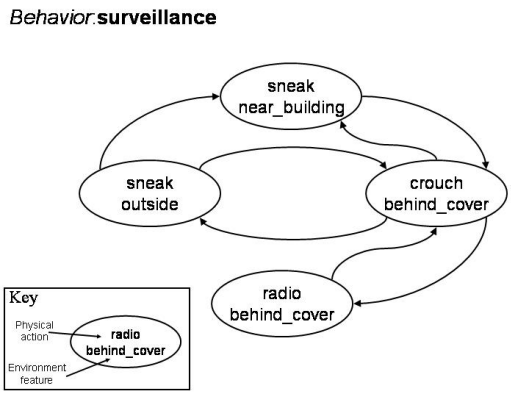

Figure 4.1: MOUT Behavior Representation: Surveillance. The *surveillance* behavior is used when a soldier wants to examine a building from the outside in preparation for entering the building, either as part of an attack or a building clearing operation. Any of the states listed in the diagram are valid starting points for the behavior; there is no single state transition that must always occur at the start of every surveillance behavior.

- *Physical actions* { walk, run, sneak, probe, wounded_movement[1], rise, crouch } are physical movements (Figure 4.4) classified from human motion capture traces using a support vector machine (SVM) action classifier as described in Section 4.1.2. Shoot and radio are special physical actions with auditory effects that are assumed to be reliably detected without relying on the action classifier.

- *Environmental features* { NEAR_HAZARD, BEHIND_COVER_INT, BEHIND_COVER_EXT, NEAR_CROSSING, NEAR_INTERSECTION, IN_CORRIDOR, IN_STREET, IN_ROOM, NEAR_INT_DOOR, NEAR_EXT_DOOR, IN_BUILDING, OUTSIDE } are derived directly from the simulator based on the human's (x, y) location as described in Section 4.1.2 and are assumed to be reliable.

- *States* are represented as a combination of the 9 recognized human actions with the 12 environment features (108 possible states).

- *Observation traces* are sequences of observed state transitions; since all of our states are self-connecting, only transitions between different states are recorded in this trace.

Human *behaviors* are represented as directed acyclic graphs of states, similar to the representation commonly used for robotic behaviors. For constant-time retrieval efficiency, this is implemented as a hashtable that maps state transitions to the set of behaviors consistent with the given observation. For the MOUT soldier domain, we created a library of 20 behaviors including ambush, bounding, scouting, and guarding.

[1]This system only models leg wounds, in which the human subject is limping with either their right or left leg. In the future, we could include crawling as a physical action; however it is difficult to acquire good crawling data with a ceiling ring of motion capture cameras since many of the markers on the subject's body are obscured.

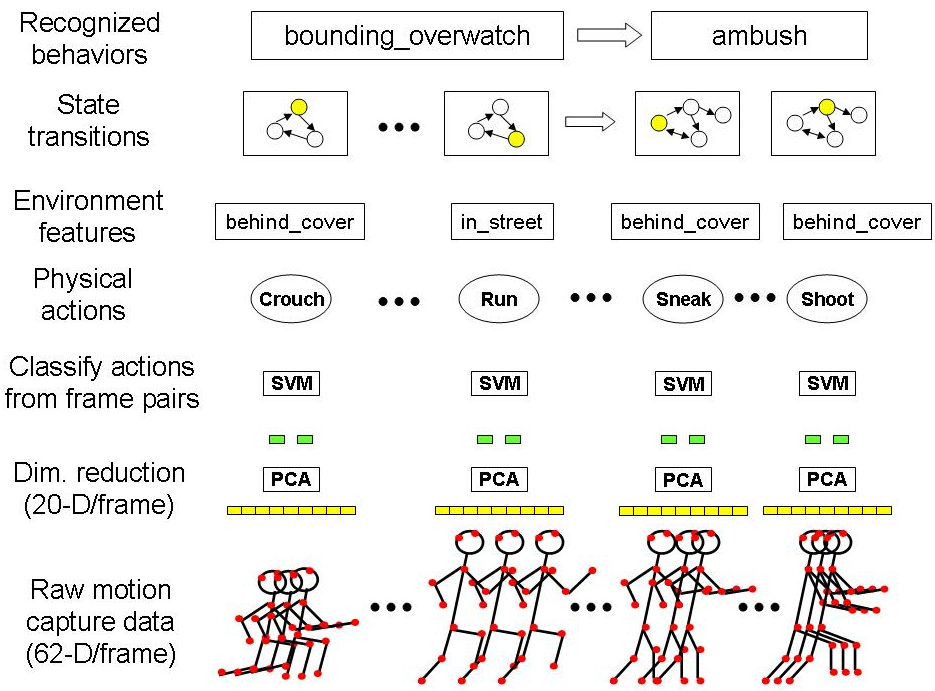

Figure 4.2: System diagram. The purpose of our system is to recognize and produce an accurate description of physical *behaviors* performed by a single human subject engaged in a MOUT scenario.

The behavior author need not explicitly describe every state transition in the graph; the system will automatically expand general feature descriptions into a complete set of legal state transitions for the specification. For instance, the surveillance behavior (Figure 4.1) includes the description *sneak* OUTSIDE which could refer to many possible states (such as *sneak* IN_STREET or *sneak* IN_BUILDING). All of the transitions between these expanded states are automatically generated when the behavior library is compiled.

4.1.2 Method

System Architecture

The purpose of our system is to recognize and produce an accurate description of physical *behaviors* performed by a single human subject engaged in a MOUT scenario. The human's *physical actions* are recorded using a motion capture system and classified by a set of SVM classifiers (Section 4.1.2). These actions are used by an environment simulator (Section 4.1.2) to generate *state* transitions; these sequences of state transitions, or *observation traces*, are used as the input to the behavior recognition system. To better understand the effectiveness of the individual components we decompose the process into two separate tasks, physical action classification and

behavior recognition, and evaluate them separately. The current implementation of our system operates in an off-line mode; data from the subject is recorded in the motion capture lab and processed off-line.

Procedure

The human subject is instructed to perform a sequence of physical actions in the motion capture lab while wearing a retro-reflective marker set (Figure 4.3). This produces a stream of high dimensional data describing the human's trajectory over time while performing specified physical actions. The data is collected and processed offline to produce training and test set for our action classifiers; to improve the classification performance we preprocess the data using principal components analysis (PCA) to reduce the dimensionality. Pairs of reduced-dimension motion capture frames are used as input for our action classifiers; a hidden Markov model is employed to post-process the raw classifications to reduce the occurrence of spurious state transitions.

To test our behavior recognition, we use a simulator that generates environmental features such as NEAR_BUILDING, NEAR_DOORWAY to supplement the motion capture data. Our assumption is that such features can be generated trivially from knowledge of the subject's (x, y) location (given by motion capture) and the simulated scenario environment. We synthesize observation traces in the simulator which serve as input to the behavior recognition.

Motion Capture Data

The human motion data was captured with a Vicon optical motion capture system. The system has twelve cameras, each of which is capable of recording at 120Hz with images of 1000×1000 resolution. We use a marker set with 43 14mm markers that is an adaptation of a standard biomechanical marker set with additional markers to facilitate distinguishing the left side of the body from the right side in an automatic fashion. The motions are captured in a working volume for the subject of approximately 8'×24'. A subject is shown in the motion capture laboratory in Figure 4.3. The data can be captured in an on-line fashion for immediate use or collected off-line for training purposes. Each motion sequence contains trajectories for the position and orientation of the root node (pelvis) as well as relative joint angles for each body part expressed as Euler angles, stored in the Acclaim AMC format, along with a skeleton that includes the subject's limb lengths and joint range of motion (computed automatically during a calibration phase), stored in an Acclaim ASF format. For our experiments, motion sequences were acquired at a rate of 30 frames per second. Out data has been included in the CMU Motion Capture database, available at `http://mocap.cs.cmu.edu/`.

Figure 4.3: A subject wearing a retro-reflective marker set in the CMU motion capture laboratory.

Dimensionality Reduction

To improve the robustness of action classification and prevent overfitting, we preprocess the motion capture data as follows. First, in order to make our action classifier invariant to global position, we transform the motion capture data to position the root node at the origin. Second, we eliminate trajectories of the minor appendages (fingers, thumbs, and toes) which are noisy and unimportant for the MOUT domain. Finally, we use Principal Components Analysis (PCA) [10] to reduce the pose vector to a manageable size, as described below. PCA has been employed in many applications, including the segmentation of motion capture data sequences [4].

PCA, also known as the discrete Karhunen-Loève transform, is an optimal linear method for reducing data redundancy in the least mean-squared reconstruction error sense. Using PCA, points in $\Re^d$ are projected into $\Re^m$ (where $m < d$, typically $m \ll d$). The intuition is that many real-world high-dimensional data sets can be well-approximated by lower dimensional manifolds embedded in the original space. We believe that the intrinsic dimensionality of poses relevant to our domain is much lower than the raw pose vector generated by the motion capture system. Each raw motion capture frame can be expressed as a pose vector, $\mathbf{x} \in \Re^d$, where $d = 56$. This high-dimensional vector can be approximated by the low-dimensional feature vector, $\theta \in \Re^m$, using the linear projection:

$$\theta = \mathbf{W}^T(\mathbf{x} - \mu), \tag{4.1}$$

where $\mathbf{W}$ is the principal components basis and μ is the average pose vector, $\mu = \frac{1}{N}\sum_{i=1}^{N}\mathbf{x}_i$. The projection matrix, $\mathbf{W}$, is learned from a training set of $N = 16055$ frames of motion capture data, spanning the set of physical actions in our domain. $\mathbf{W}$ consists of the eigenvectors corresponding to the m largest eigenvalues of the training data covariance matrix, which are extracted using singular value decomposition (SVD). This reconstruction is theoretically perfect only when $m = d$; however, in our application, $m = 20$ produces reconstructions that are visually indistinguishable from the raw data (these components account for more than 95% of the energy in the data). The principal components are only computed once, in an off-line phase; di-

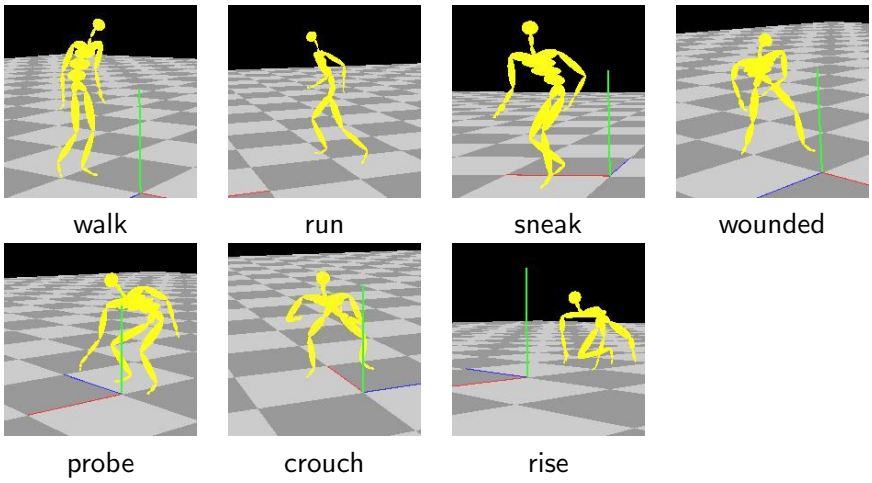

Figure 4.4: Representative poses for the subject's physical actions.

mensionality reduction of incoming motion capture frames is achieved by the efficient linear projection described by Equation 4.1.

Physical Action Classification

The goal of physical action classification is to label a short sequence of frames as a member of one of k categories (e.g., run, sneak, radio). We perform this classification using support vector machines [40]. Support vector machines (SVM) are a supervised binary classification algorithm that have been demonstrated to perform well on real-world visual pattern classification tasks [32]. Intuitively the support vector machine projects data points into a higher dimensional space, specified by a kernel function, and computes a maximum-margin hyperplane decision surface that separates the two classes (e.g., run from walk). Support vectors are those data points that lie closest to this decision surface; if these data points were removed from the training data, the decision surface would change. More formally, given a labeled training set

$$\{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_l, y_l)\},$$

where $\mathbf{x}_i \in \Re^N$ is a feature vector and $y_i \in \{-1, +1\}$ is its binary class label, an SVM requires solving the following optimization problem:

$$\min_{\mathbf{w},b,\xi} \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{l}\xi_i$$

constrained by:

$$y_i(\mathbf{w}^T\phi(\mathbf{x}_i) + \mathbf{b}) \geq 1 - \xi_i,$$
$$\xi_i \geq 0.$$

The function $\phi(.)$ that maps data points into the higher dimensional space is not explicitly represented; rather, a *kernel* function, $K(\mathbf{x}_i, \mathbf{x}_j) \equiv \phi(\mathbf{x}_i)\phi(\mathbf{x}_j)$, is used to implicitly specify this mapping. In our application, we use the popular radial basis function (RBF) kernel:

$$K(x_i, x_j) = \exp(-\gamma||x_i - x_j||^2), \gamma > 0.$$

Many efficient implementations of SVMs are publicly available; we use LIBSVM [8] because it includes good routines for automatic data scaling and model selection (appropriate choice of C and γ using cross-validation). To use SVMs for k-class classification, we train ${}^kC_2$ pair-wise binary classifiers and assign the most popular label.

After applying PCA to the raw motion capture data, the human's pose in each frame is represented by a 20-dimensional vector. To train our action classifier, we form 40-dimensional vectors by concatenating 2 pose vectors separated by 1/3 second (10 frames). Our accuracy using a single frame without concatenation is about 20% worse; intuitively certain action classifications (e.g., walk vs. sneak) are much more difficult without information about how the pose evolves over time.

To train and evaluate our action classifier, we collected motion capture data of the subject performing the domain physical actions (Figure 4.4). We acquired 17 minutes of motion capture data stored in 25 AMC files. From the 32,111 total data frames, we divided the first half of the frames in each file to use as the training set and used the second half for testing. Using LIBSVM, we ran cross-validation on the training set to determine the best parameters C and γ for the RBF kernel. The confusion matrix for our action classification on the test data is shown in Table 4.1. The left column shows the label of the correct behavior, and the top row shows the assigned label. The average classification accuracy over the testing dataset was 76.9%. For the locomotion actions (walk, run, etc.) we achieved higher accuracies than on the non-locomotion actions (crouch, rise, probe). To improve performance on these physical actions, we believe that we need to increase our training set size since our initial motion clips of those actions were short compared to the other actions (only about 2000 frames per action class rather than 6000).

State Transition Filtering

To reduce spurious state transitions caused by false detections we filter our raw SVM classifications using a hand-coded Hidden Markov Model. To do this, we treat the classification labels generated by the SVM as observations and the true motion label as a hidden state. The most likely path of state transitions for a given sequence of observations is computed using the Viterbi algorithm [25] as implemented in the Hidden Markov Model toolbox [22]. Our model is parameterized by the following:

- $N = 7$, the number of hidden motion states (walk, run, sneak, probe, wounded_movement, probe, crouch, rise).

Table 4.1: Confusion matrix for SVM action classification using pairs of concatenated frames with a $1/3$ second (10 frame) separation (40-dimensional vectors). The left column shows the label of the correct behavior, and the top row shows the assigned label. High results along the diagonal indicate good performance. The average classification accuracy over the testing dataset was 76.9%.

	walk	run	sneak	wound	probe	crouch	rise
walk	85.4%	3.9%	7.8%	0.6%	1.8%	0.5%	0.0%
run	12.5%	75.8%	7.9%	0.5%	2.8%	0.5%	0.0%
sneak	6.4%	0.8%	81.9%	0.0%	1.7%	9.2%	0.0%
wound	1.3%	3.4%	0.8%	93.9%	0.4%	0.3%	0.0%
probe	8.8%	3.6%	8.7%	19.3%	56.5%	0.6%	2.5%
crouch	2.9%	18.0%	10.3%	20.0%	11.2%	35.7%	1.9%
rise	12.1%	3.7%	15.1%	32.2%	6.4%	16.3%	14.2%

- $\mathbf{A} = \{a_{ij}\}$, the matrix of state transition probabilities, where $a_{ij} = Pr(q_{t+1} = j | q_t = i), \forall i, j$ and q_t denotes the state at frame t. For all states we assume that the probability of remaining in a given motion state ($Pr(q_{t+1} = i | q_t = i)$) is high and transitioning to any of the other six motions is equally likely.

- $\mathbf{B} = \{b_i(o_t)\}$, the vector of observation probabilities derived from the confusion matrix (Table 4.1). Intuitively the confusion matrix captures how the classifier label (observation) is correlated with the ground truth (hidden motion state).

- $\pi = \{\pi_i\}$, the initial state distribution. All initial states are assumed to be equally likely.

Behavior Recognition

During the final behavior recognition phase, we assign behavior labels to observation traces (sequences of state transitions). Observation traces are generated using the simulator as described below. These state transitions are of the form *(physical_action1* LOCATION_1 *physical_action2* LOCATION_2*)* where some change has occurred such that *physical_action1* $\neq$ *physical_action2* and LOCATION1 $\neq$ LOCATION2.

First we initialize the behavior recognition with a table hashing state transitions (about 11000 entries) to sets of legal behaviors (hypothesis sets). The hashtable is compiled directly from the behavior specifications to enable efficient searching; behaviors that include states with general features (e.g., outside) are expanded to include legal transitions for more specific cases (e.g., NEAR_BUILDING, NEAR_INTERSECTION).

The domain author designates a cost function to be applied to each potential *behavior transition*. A simple parsimonious cost function is to penalize any behavior transition by a fixed amount; self-transitions (explaining the subsequent behavior with the same label) are not penalized. This type of cost function delays as long as possible before assigning a new behavior label to state transitions.

Using the behavior library and the cost function, we search for the minimum cost

explanation for the sequence of state-transitions; this is a shortest path problem which we solve efficiently using dynamic programming:

$$B_{t+1}^{q} = \min_{\forall p}\{B_{t}^{p} + T_{p,q}\},$$

$$T_{p,q} = \begin{cases} 0 & \text{if } p = q \\ 1 & \text{if } p \neq q. \end{cases}$$

B_{t+1}^{q} is the cost of explaining a state-transition with behavior label q at time $t+1$; this cost can be calculated by finding the minimum of over all previous behavior labels p of explaining a set with B_{t}^{p} combined with the cost transition function, $T_{p,q}$.

Environmental Simulator

To test new plan libraries and cost functions without the prohibitive expense of acquiring human data in the motion capture lab, we implemented an environmental simulator system capable of generating valid observation traces that correspond to a sequence of known behaviors performed in a specific MOUT environment, composed of typical urban terrain features.

The simulator is equipped with behavior descriptions, either known ones taken from the library used by the behavior recognizer or new ones written to represent cases in which the human deviates from the correct military procedure. The input to the simulator is a list of behaviors from which the simulator stochastically generates one run of valid observation traces that could have occurred if the human performed those behaviors in that MOUT environment layout. Currently our stochastic model is very simple; all possible state transitions exiting a state are equally likely.

Environmental feature descriptors (e.g., NEAR_DOORWAY, NEAR_INTERSECTION) are generated by selecting the location nearest to the soldier's (x, y) location; there is a preference order (e.g., NEAR_HAZARD dominates features such as NEAR_INTERSECTION) that dictates which environmental feature is reported if the human is equally close to multiple annotated map regions.

In the simple case, we assume that all state transitions are accurately detected; to model the effects of imperfect state transition detection we use the confusion matrix (Table 4.1) to stochastically model the likely output of our action classifier. For instance, the true state transition might be (*walk* NEAR_DOORWAY *walk* NEAR_DOORWAY); consulting the top row of the confusion matrix we see that the walk physical action has an 85.4% chance of being detected correctly as walk, 3.9% chance of being classified as a run, 7.8% of being classified as a sneak, and a negligible chance of being detected as anything else. Using the confusion matrix as our simulator noise model we can systematically generate faulty data that realistically models the effects of imperfectly classified the motion capture data. Also, as we improve our classification procedure, we can quickly assess the impact on the behavior recognition.

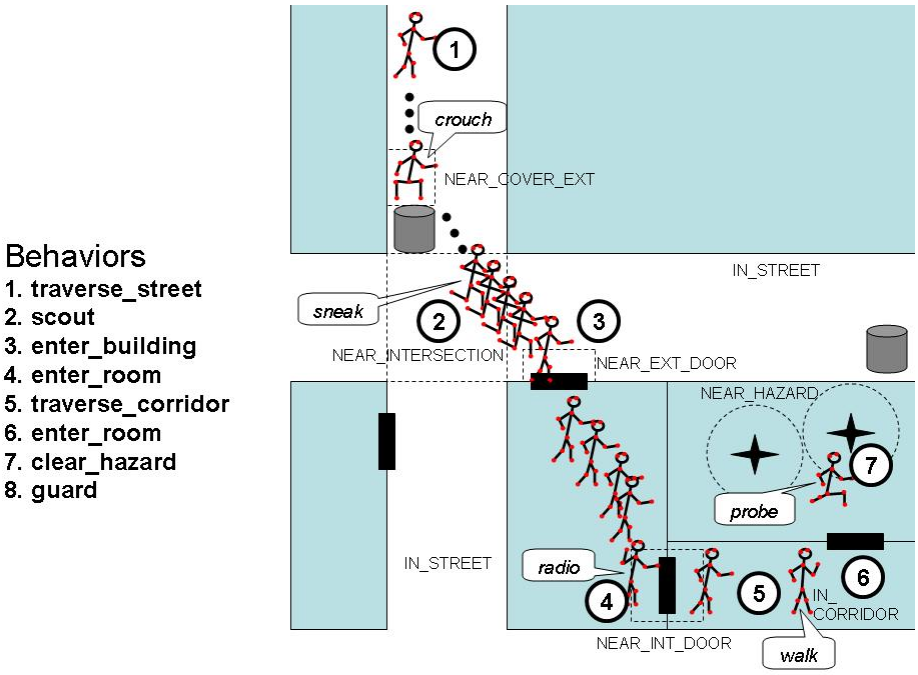

Figure 4.5: MOUT Scenario: Building Clearing. An overhead view of the schematic used by our simulator to generate observation traces for an example building clearing scenario. The standard military procedure would be to use the following sequence of behaviors: **traverse_street**, **scout**, **enter_building**, **enter_room**, **traverse_corridor**, **enter_room**. Since the room marked by the star appears cluttered and might potentially contain booby-traps, the soldier should choose to check the area for hazards (**clear_hazard**).

4.1.3 Results

We examine our cost minimization approach to behavior recognition in the context of a common MOUT scenario to assess the impact of the following factors:

behavior transition cost function: how does changing the transition cost function affect the behavior explanation generated? Is there a good method for using domain knowledge to author a function $T_{p,q}$ that produces good recognition results?

human behavior deviations: how should we model human deviations from the textbook military behaviors? Can we effectively recognize these deviations?

MOUT Scenario: Building Clearing

Building clearing, the process of investigating a building and eliminating hostile occupants and hazards within, is a common goal in MOUT operations. The standard followed military procedure used to clear the building shown in Figure 4.5 would be to use

the following sequence of behaviors: **traverse_street**, **scout**, **enter_building**, **enter_room**, **traverse_corridor**, **enter_room**. Since the room marked by the star appears cluttered and might potentially contain booby-traps, the soldier should choose to check the area for hazards (**clear_hazard**). In the final state, the soldier remains to guard the building to ensure that no one enters the building. If there are enemy forces in the area that fire on the soldier during the **traverse_street** behavior, the soldier should make a strategic withdrawal and counterattack; this can be accomplished by executing the behavior sequence, **retaliate retreat fast_bound** and **ambush**, before returning to the original building clearing operation.

Impact of Cost on Behavior Recognition

The behavior transition cost function, $T_{p,q}$, directly affects the explanation generated by the dynamic programming search process. Using the parsimonious cost function described in Section 4.1.2, we analyze stochastically-generated state transition sequences for the building clearing operation (**traverse_street**, **scout**, **enter_building**, **enter_room**, **traverse_corridor**, **enter_room**, **clear_hazard**, **guard**). Typically, this behavior sequence generates about 50 state transitions; in the absence of classification noise, the recognizer with parsimonious cost function correctly labels between 90–100% of the state transitions. Often it mislabels the final guard behavior as being part of a **enter_room** behavior, because many of the same state transitions appear in both. However using our domain knowledge we know that the soldier should guard an area after clearing it; to represent that domain knowledge we decrease the cost of **clear_hazard** followed by **guard**. Injecting noisy state-transitions only causes a slight degradation in the classification of the remaining non-noisy state transitions.

We believe that the cost function is closely related to the application as well as the domain. For instance if the agent were attempting to model an opponent, using a paranoid cost function that is sensitive to the ambush behavior (allowing cheaper transitions from common behaviors to ambush) might be more useful, even at the expense of a slight decrease in overall behavior recognition accuracy. The interaction between behavior recognition accuracy, application objectives (teamwork, opponent modeling, human training) and behavior transition cost functions is complicated and worthy of further study.

Human Behavior Deviations

Potentially more problematic than the action labelling noise is the concern that the behavior executed by human subjects could deviate from the state transitions specified by the domain expert's behavior library. There are two types of potential deviations: (1) rarely-executed action sequences that are correct but inadequately represented by our behavior library; and (2) errors made by human trainees that should be flagged for correction. For instance, a human clearing an area of hazards with the probe physical action might occasionally crouch to visually inspect the area from a different vantage point; this less commonly used physical action is not currently represented

as a valid state transition in the probe behavior. However, it is a tactical error for a soldier to enter an unknown building without executing a defensive physical action (e.g., crouch, sneak, or shoot); yet this is a mistake that novice trainees might make if they relaxed their guard in the absence of an obvious threat.

To deal with novice errors we explicitly developed alternate behaviors encoding common novice mistakes as well as behaviors suitable for panicked soldier or civilian (e.g., **flee** or **hide**). Since the MOUT domain is relatively structured compared to other activity-inferencing tasks such as food preparation, most of the deviation in the subject's behavior can be attributed to error rather than individual variation. The behavior transition cost function can be explicitly tailored to be hypersensitive to potential errors (preferentially selecting the error in favor of other explanations) or to only flag transitions as errors if no valid behavior explanation exists.

4.1.4 Summary

By incorporating a behavior transition cost function into the behavior recognition process, we can use the same behavior library for all three tasks by adjusting the cost function. For instance, during behavior recognition for teamwork, we want our agent to be sensitive to potential assistive actions (e.g., aiding an injured teammate or coordinating to ambush an enemy); by modifying the cost function to make certain behavior transitions cheaper we ensure that the appropriate behavior explanations are preferred when they are possible. The behavior transition cost function implicitly codes the agent's signal detection preferences— whether false-hits for certain behaviors are preferable to missed detections.

Our default parsimonious behavior transition cost function, as described in Section 4.1.2, is similar to Kautz's minimum cardinality assumption (MCA) [1]; the assumption is that it is desirable to choose the minimal set of behaviors to explain the observation trace. Our transition cost function minimizes the number of transitions between behaviors by penalizing every additional behavior transition; this usually produces the same results as minimizing the number of behaviors. However, in uncommon cases where multiple behavior sequences have identical transition costs, the dynamic programming solution does not necessarily prefer the sequence with the fewest different behaviors.

4.2 Identifying Team Behaviors from Spatial Cues

An important part of classifying MOUT (Military Operations in Urban Terrain) team behaviors is recognizing subtle spatial relationships between physical entities: opponents waiting in ambush, teammates organizing around a rendez-vous point, and potentially dangerous cul-de-sacs. In this section, we present a RANSAC (Random Sampling and Consensus) based algorithm for identifying spatial relationships of MOUT entities based on a model library; possible configurations are scored based on a

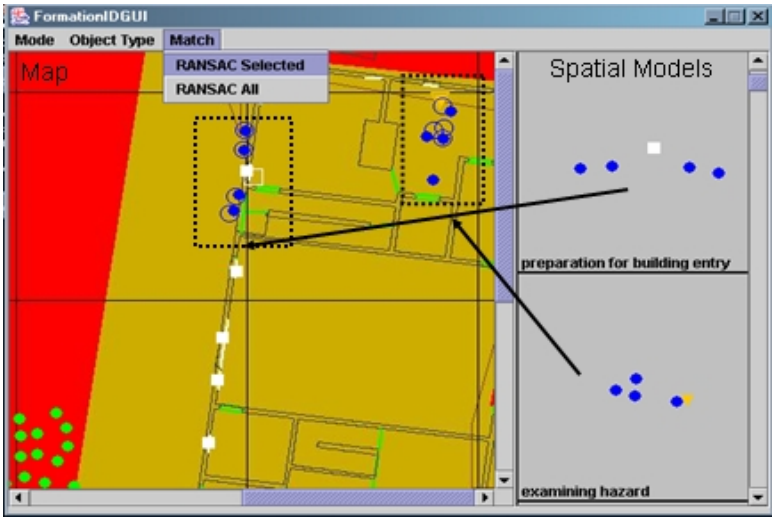

Figure 4.6: Spatial model authoring and matching system. The library of previously created spatial models are shown to the right of the screen; the left side of the GUI displays an annotated map to be analyzed. A fire team of soldiers (blue circles) examining a hazard (orange inverted triangle) are displayed on the map; a second fire team prepares to enter the building through the door (marked by the white rectangle). Our matching technique successfully associates both groupings of soldiers with the correct spatial models; hollow circles and rectangles show the locations of the entities as predicted by model projection. Note that the positions of the entities as predicted by the projection of the hazard examination model are not precisely localized, even though the model is correctly situated. A method for enhancing the position estimation is discussed in Section 4.2.2.

similarity function that incorporates information on entity type matching, transform validity, spatial proximity, and preservation of visibility constraints. Configurations can include both static entities (doors, buildings, hazards) and dynamic ones (opponents, teammates, and civilians). We demonstrate that our algorithm is robust to spatial variations and generalizes across scenarios.

4.2.1 Method

Our method requires an initial phase of constructing one or more spatial models to correspond to each physical behavior; we designed an authoring graphical user interface (GUI) to facilitate the model creation process (see Figure 4.6). Once a library of spatial models has been constructed, they can be used to classify formations of MOUT entities on a 2D annotated map as being characteristic of a particular team behavior. The same technique can also be used to classify spatial groupings of static entities (e.g., doors and walls) to be used as decision cues for the building clearing task.

Spatial Representation of MOUT Entities

To model MOUT team behaviors, we developed a tool that enables the author to describe behaviors by designating a set of characteristic spatial relationships that commonly occur during the execution of the behavior. The model contains the following attributes:

behavior name: Behaviors are represented by collections of spatial models; however no particular temporal structure, or execution order, is attached to the collection. Each spatial model can only belong to a single behavior; we do not include models which appear in a large set of behaviors since they are unlikely to help in discriminating between multiple behaviors.

spatial position of relevant entities: Entities are represented by a single (x,y) coordinate of their centroid; larger entities are represented as groups of points connected by a visibility constraint (see below).

entity type: For our library of MOUT behaviors, we designated eleven types of entities along with a relationship hierarchy which we use in scoring the compatibility between entity matches (See Section 4.2.1). Some of the types (e.g., objectives and hazards) do not refer to a specific physical type of object or area but are used to designate the role that the object plays in the world. Entity types include: person (unknown), civilian, teammate, opponent, hard cover, soft cover, empty area, windows, intersections, doorways, hazards, and objectives.

pairwise constraints between entities: For certain behaviors, lines of visibility (or lack of visibility) between entities are an important part of the spatial relationships. We model these visibility relationships as line segments between map entities that are either visibility preserving (cannot cross occlusions) or enforce lack of visibility between entities (must cross an occlusion).

scaling limitations: Certain models are only valid at a limited range of scales. For instance, a model representing a formation of foot soldiers would remain valid if the separation between soldiers was rescaled from 3m to 10m; however if the separation between soldiers increased to 1 km, the same model should no longer be valid (even if visibility constraints were satisfied).

Using Spatial Transforms to Generalize Models

One consideration in developing models is generalization—how well do models developed for one scenario match behaviors executed in a different spatial layout? Without generalization it becomes impractical to exhaustively enumerate all possible spatial relationships that can occur across different maps. To solve this problem, we define a set of legal transforms to project models to new spatial layouts and score the quality of the match.

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

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

where θ is the angle of rotation, s is a scale factor, x is the x-translation, and y is the y-translation. This formulation can easily be extended to model three dimensional transforms by increasing the matrix to 4×4. The next section describes a robust and efficient technique for searching the space of possible transforms.

Robustly Matching Models to Maps

Given a set of spatial models and valid transforms, the problem of determining which spatial models are applicable to the current map can be solved by searching the space of potential transforms and models to find all the combinations of model plus transform that result in a match of sufficient quality. A commonly-used approach is exhaustive template matching [3]. Each templates is applied to all possible locations in the map using a sliding window; the distance function is calculated over the window area and matches that score under the threshold are retained. This process is exhaustively repeated for a range of scales and rotations, over all models in the library. Unfortunately this process is time consuming, scales poorly to higher dimensional transforms, and is sensitive to noise, occlusion and misalignment.

Instead, we employ a statistically robust technique, RANSAC (Random Sampling and Consensus) [11], to efficiently sample the space of transforms using hypotheses generated from minimal sample sets of point correspondences. RANSAC can be summarized as follows:

hypothesis generation: entities are drawn uniformly and at random from the annotated map and associated with randomly selected entities of the same type in the model. Two pairs of corresponding entities are sufficient to uniquely specify a transform hypothesis. This data-driven method of generating hypotheses is much more efficient than uniformly sampling the space of possible transforms or exhaustively searching a discretization of the transform space.

hypothesis testing: Given a transform hypothesis, we project all of the entities in the model to the coordinate frame of the map and assess the quality of the match based on both spatial similarity and type matching. This gives us the likelihood that the given hypothesis generated the observed data in the map. Our likelihood estimate is robust to missing data (entities in the model that are missing on the map) and to outliers.

For each spatial model, we use RANSAC to randomly generate and test a large number of plausible transforms and select those hypotheses (a combination of a model

and a valid transform) with match quality better than a specified threshold. Below, we describe the two phases of our algorithm in greater detail.

Hypothesis Generation

Since our spatial transforms have four degrees of freedom, they can be fully specified by two pairs of point correspondences. First, we randomly select two entities from the model under consideration; then based on the types of the entities (e.g., civilians, hard cover, hazard) we randomly select candidate entities on the map of compatible object types. The positions of these entities is used as the minimal set to generate a transform hypothesis as follows.

Given the minimal set $\{(x_1, y_1), (x_2, y_2)\}$ from the model and the corresponding set of points $\{(X_1, Y_1), (X_2, Y_2)\}$ from the map, we generate a third virtual pair of correspondences $(x_3, y_3) \mapsto (X_3, Y_3)$ where

$$\begin{aligned} x_3 &= x_1 + y_2 - y_1 \\ y_3 &= y_1 + x_1 - x_2 \\ X_3 &= X_1 + Y_2 - Y_1 \\ Y_3 &= Y_1 + X_1 - X_2 \end{aligned}$$

From these three correspondences, we can directly recover $\mathbf{T}$ using matrix inversion.

$$\begin{bmatrix} t_{11} & t_{12} & t_{13} \\ t_{21} & t_{22} & t_{23} \\ t_{31} & t_{32} & t_{33} \end{bmatrix} = \begin{bmatrix} X_1 & X_2 & X_3 \\ Y_1 & Y_2 & Y_3 \\ 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} x_1 & x_2 & x_3 \\ y_1 & y_2 & y_3 \\ 1 & 1 & 1 \end{bmatrix}^{-1}$$

This is a solution to a general affine transform [13] given three pairs of point correspondences, however $\mathbf{T}$ is guaranteed to be a valid, orientation-preserving similarity transform due to our construction of the third point.

For this domain, we assume that models are valid for all rotations and translations but are limited to a valid range of scales. If the candidate point correspondences yield a transform outside the valid scale range, it is discarded before the hypothesis testing phase.

Hypothesis Testing

We score each sampled hypothesis as follows (illustrated in Figure 4.7). First we transform the location of every entity in the model to the map using the transform $\mathbf{T}$. Each model entity contributes a positive vote for the given hypothesis if the distance from its predicted location to the closest map entity of compatible type falls below a specified threshold. The quality of a hypothesis is defined as the normalized sum of these individual votes. Additionally, we enforce pairwise constraints between entities, such as visibility and occlusion, as specified by the model. If the constraints are violated, we penalize the quality of the hypothesis.

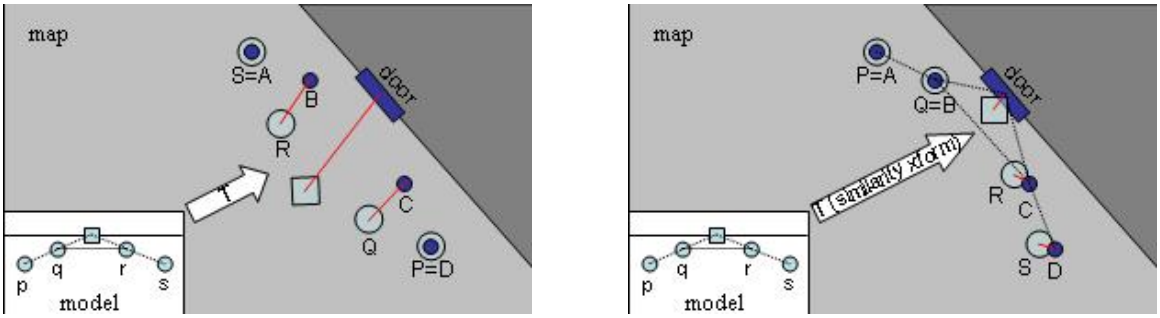

Figure 4.7: Examples of hypothesis generation and testing using RANSAC. For the building entry maneuver model shown here, we show two different sampled minimal correspondence sets. In (left), $\{P \mapsto D, S \mapsto A\}$, while in (right) $\{P \mapsto A, Q \mapsto B\}$. Given these correspondences, we derive the the transform, $\mathbf{T}$. Applying $\mathbf{T}$ to each of the model entities, gives us their predicted locations on the map. Each predicted location casts a vote in favor of $\mathbf{T}$ if there exists a map entity of compatible type within range. The first hypothesis (left) receives no additional votes since the distances to compatible entities is too great. The second hypothesis (right) is consistent with the map and receives many votes. Pairwise constraints, shown as dotted lines, are then verified. In this case, they denote desired visibility constraints between MOUT soldiers, and we confirm that the second hypothesis does not violate them. Although not shown in these simple examples, RANSAC is robust to large numbers of outliers and to missing data.

For our library of building clearing MOUT behaviors, we designated eleven different types of entities and defined three compatibility functions. Entity types include: person (unknown), civilian, teammate, opponent, hard cover, soft cover, empty area, windows, intersections, doorways, hazards, and objectives. Compatibility functions are used to determine how matches between two entities of different types affect the quality score. We used three types of compatibility functions: 1) exact match—the type of the model entity and map entity must match exactly to contribute positively toward the quality score; 2) categorical match—the map entity and model entity have to belong to the same general category, e.g., person, cover, empty space; 3) functional match—the map entity has to serve the same functional purpose as the model entity. For our experiments, we specified that the random point correspondences used to generate candidate hypotheses must be drawn from the set of exact matches; the more relaxed categorical match criteria was used during to score the quality of the hypothesis.

Our current implementation supports two types of pairwise constraints, visibility and occlusion. A visibility constraint specifies that two entities on the map must have unobstructed line of sight (or fire), while the occlusion constraint specifies that no unobstructed line of sight should exist between those entities. Once a hypothesis has passed the initial screening stage, we can verify that the constraints are obeyed using standard line intersection algorithms against obstacles and entities on the map. This is shown on Figure 4.7 (right).

We apply RANSAC to all the models in the library and generate a set of hypotheses (transform plus model) whose normalized match quality exceeds a specified

threshold. These hypotheses are all consistent with the observed data. If desired, the transforms for each of these hypotheses can be refined by applying standard least squares estimation techniques to the set of inliers for each hypothesis [13]. This is generally unnecessary for our application, except as discussed in Section 4.2.2.

4.2.2 Results

We implemented our spatial matching technique as a Java application that supports authoring of model libraries and automated matching of spatial models to annotated 2D maps (see Figure 4.6). The models that we created for the MOUT building clearing scenario contained 3–15 entities and were matched to maps with approximately 100 annotations.

Physical Team Behaviors for Building Clearing

The maps employed in our evaluation are motivated by the building clearing scenarios described in [24] and the physical behaviors given in [27]. The physical team behaviors are: traverse street (stacked formation), bypass window, enter building, flank enemy position, examine hazards, clear room, cross intersection (L-shaped and T-shaped). We also model less structured physical behaviors such as enemy sniper placements and civilian crowds.

We examine the effectiveness of our spatial matching technique on the following three dimensions:

generality: how well do spatial models developed for one scenario generalize to scenarios with different layouts?

robustness: is our matching technique robust to noisy data, mislabeled entities, and occluded points?

threat prediction: how can we use our model to predict the location of occluded elements such as enemy snipers and missing team members?

Each of these points is discussed below in greater detail.

Generality

To examine how models developed for one spatial layout generalize to different layouts, we developed several maps that include instances of the flanking behavior. When executing the flanking behavior two members of the team fire at an enemy soldier to pin him down while the other two team members move to a location that offers a better line of fire. In the left panel of Figure 4.8, the original spatial model for the terminal position of the flanking behavior is shown. Our technique successfully identifies the correct instance of flanking in the center panel despite significant spatial differences

and the existence of outlier entities. The model easily generalizes to instances with different scale and rotation because RANSAC can correctly identify (even in the presence of outliers) the similarity transform that accounts for these changes.

The third panel shows an example of flanking where the angle between the two friendly fire teams is significantly different from the model; in this scenario they have achieved a crossfire position on the enemy opponent by moving around a building. This instance of flanking fails to match since no single similarity transform applied to the entire spatial model can explain the observations on the map. This could be addressed in several ways. The simplest solution is to provide additional spatial models for this behavior to account for the diversity. Another approach is to employ a broader class of transforms, either global (affine or projective) or non-uniform (elastic graphs). The trade-off of employing a broader class of transforms is discussed in Section 4.2.3.

Robustness

Since each map contains multiple models in addition to extraneous entities that don't match any of the models, our matching technique must be able to ignore these outlier entities. In the maps we analyzed, about 95% of the entities were effectively "outliers", unrelated to the model under comparison. Fortunately RANSAC handles outliers very well by generating hypotheses only using minimal subsets; as the number of outliers increases, we can compensate by iterating more times as described by the formula given in Section 4.2.3.

Maps generated from the perceptual viewpoint of a synthetic character are incomplete, due to occlusion and limited sensor range. Our technique should be able to match models to partially populated maps; to do that we must examine the role of the model's normalized quality score and how it relates to the match acceptance threshold. Currently, we set the normalized quality threshold at 0.75; effectively this means that approximately three-quarters of the points in the model must be successfully matched. In Section 4.2.2 we discuss strategies for setting this threshold on a individual model basis according to signal detection theory.

Threat Prediction

We can exploit our technique's robustness to missing data to predict the potential locations of unseen threats. A complete spatial model can match a partial set of entities on the map if the normalized sum of the existing votes is sufficiently high. For instance, given a spatial model describing commonly occurring enemy sniper vantage points, our technique can predict likely sniper positions even if the enemy forces themselves are not visible. This is useful in the case where the map is incomplete and reflects only the accumulated observations from one team's agents.

To predict the location of hidden entities, we simply project all the unmatched entities in the matching model to the map using the similarity transform; no addi-

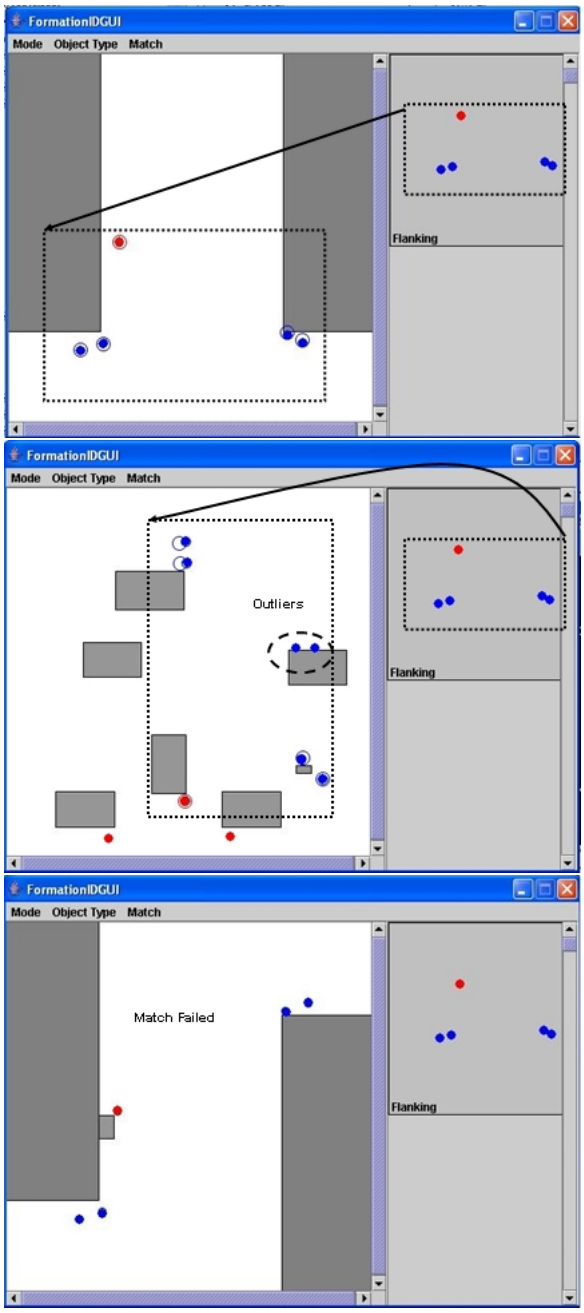

Figure 4.8: Three examples of team flanking behavior in different scenario layouts. The spatial flanking model (shown in the right hand side of the GUI) was originally designed for the layout displayed in the top panel. The middle panel shows the flanking behavior occurring in different spatial layout that includes more opponents and an additional fire team. Our technique successfully matches the flanking model to this new situation in spite of the differing spatial layout and the addition of extraneous entities (marked as outliers). The bottom panel shows an instance of the flanking behavior that fails to match the model because no single similarity transform applied to the entire spatial model can explain the observations on the map.

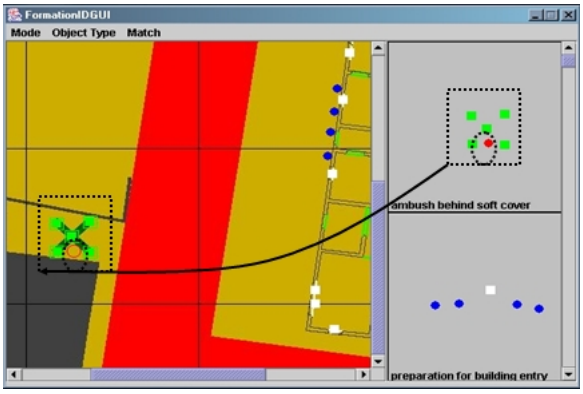

Figure 4.9: Using the best hypothesis match to predict unseen threat locations. A fire team of soldiers (denoted as blue circles) is shown moving in a stacked formation toward the building entry. To the left hidden behind a clump of trees (denoted by the green rectangles) is an enemy sniper covering the building. Although the sniper itself is not visible, its likely position is detected (marked by the empty red circle) by a successful match of the model shown in the top right hand corner. The transform found by RANSAC during the matching process is used to project all the entities of the spatial model to their hypothesized locations.

tional computation is necessary since the transform was already computed during the hypothesis generation phase. However this initial transform was computed from a minimal set of two point correspondences. To produce a more refined estimate of the similarity transform, we can incorporate information from all the points matching the model to create an overconstrained system of equations which can be solved using standard least squares techniques. This additional effort is justified only if we need extremely accurate estimates of the positions of entities within the model.

Figure 4.9 shows a fire team of soldiers (denoted as blue circles) moving in a stacked formation toward the building entry. To the left, hidden behind a clump of trees (denoted by the green rectangles), is an enemy sniper covering the building. Although the sniper itself is not visible, its likely position is detected (marked by the empty red circle) by a successful match of the model shown in the top right hand corner.

Applying spatial reasoning for threat prediction typically entails a trade-off; as the number of unmatched entities in the model increases we must lower the matching threshold to enable these partial matches to be detected. Increasing the sensitivity in this manner also increases the number of false alarms (i.e., hallucinating threats where none exist). It may be useful to set matching thresholds on a model by model basis to correctly reflect the cost of false alarms vs. missed matches. For detecting enemy behaviors (snipers, enemies concealed in crowds) the matching threshold should be set lower to reflect the high cost of missing enemy threats.

4.2.3 Discussion

Since RANSAC stochastically searches the space of possible transforms it is not guaranteed to find the best match. However the following formula can be used to determine how many iterations are necessary to achieve the best match with a specified probability of success [41]:

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

P is the target probability (e.g., $P = 0.99$ means the best match is found 99% of the time). s is the number of elements required to define the minimal set ($s = 2$ since a similarity transform requires 2 pairs of point correspondences). ϵ is the expected fraction of outliers in the data set. In traditional RANSAC applications ϵ is typically only about 0.1 (10% of the points are expected to be invalid). For our application, the fraction of outliers refers to the number of map annotations that do not match a single model; since each map actually contains multiple models in addition to entities that do not match any model the fraction of expected outliers is approximately 0.95. From the formula above, this indicates that the number of RANSAC iterations required to reliably find the best match is 1840 which is computationally inexpensive, especially compared to exhaustive searching on a large map.

The number of iterations is relatively small due to the low value of s, the number of elements required to define the minimal set which completely specifies the spatial transform. If we expanded the class of allowable transforms, either to include 3D similarity transforms or a broader class of 2D transforms (e.g., affine), the number of elements required to specify the spatial transform would increase to $s = 3$ and the iterations required would increase to 36841.

4.2.4 Summary

By developing composite spatial models for team behaviors, we avoid an exponential expansion of state space; instead of representing each team member's state separately we collapse the state of the team into a single spatial model representing the total state of the team (or subteam). This spatial representation, as given, does not include a representation of how each behavior evolves over time; without this it is difficult to identify repetitive behaviors such as bounding overwatch that are defined by a combination of temporal and spatial relationships. However, this simple spatial model is robust to outliers, invariant to many types of spatial transforms, and can express a wide variety of static relationships and constraints between physical entities. For a given model and quality function, we can guarantee that 99% of the time RANSAC finds the best potential transform within 2000 iterations which can be executed in a fraction of a second.

4.3 Team Behavior Recognition in UT (2 Person)

This section describes a methodology for recording, representing, and recognizing team behaviors performed by human players in an Unreal Tournament MOUT scenario. To directly monitor the performance of human players, we developed a customized version of Unreal Tournament (UT) that records position and orientation of all the team members through time as they participate in a simulated MOUT scenario of a firing team moving through an urban area. Behavior recognition is performed offline using a set of Hidden Markov Models on short movement sequences that are translated into a canonical reference frame; the behavior model with the highest log likelihood for a given sequence is identified as correct. By transforming the data into a canonical frame, we create spatially-invariant Hidden Markov Models that correctly identify the same behaviors performed in different map regions.

4.3.1 Team Behaviors

We focus on three behaviors, stacked movement, bounding overwatch, and buttonhook entry used by MOUT teams to approach and enter a building. These behaviors are difficult to identify solely on the basis of static snapshots due to their spatial similarity. During stacked movement the purpose is to move the team in such a way that their gun angles completely span all possible areas of approach; the team moves slowly and in synchronization. For moving through open areas or intersections, this approach is less feasible since it's hard to span all possible threatened areas. In this case, the bounding overwatch behavior is used; one soldier moves forward while the other remains stationary. The buttonhook entry is similar to bounding overwatch; one soldier moves through the doorway hugging the wall while the other soldier waits and guards. After the entry is clear, the second soldier moves through the doorway hugging the opposite wall.

4.3.2 Method

The results described in this section were obtained using the following procedure:

1. Pairs of human players using a modified Unreal Tournament game interface manipulated "bots" through a small urban layout.

2. After some initial practice familiarizing themselves with the map and working together as a team, the subjects were instructed to perform a particular sequence of team behaviors.

3. Using the modified version of Unreal Tournament, traces of the players' behaviors were recorded in a text file for offline analysis.

Figure 4.10: MOUT scenario in customized Unreal Tournament environment from spectator viewpoint. A pair of human players control soldiers A and B as they move through a small urban layout. The bot models and animations were modified to conform to the appearance of real human soldiers rather than the larger-than-life UT fantasy fighter models.

4. Behavior traces were preprocessed to generate short, overlapping segments and converted into a canonical reference frame based on the motion of the team's centroid.

5. Offline traces were automatically classified using a set of hidden Markov models.

6. The results of our automatic recognition were compared with a manually annotated version of the trace.

Section 4.3.3 gives preliminary results obtained with a single pair of players who performed the task multiple times in different layouts.

Data Collection

To directly monitor the performance of human players, we customized the first-person shooter game, Unreal Tournament (UT), using the game development language Unrealscript. Many of the original UT game classes were written in Unrealscript and thus can be directly subclassed to produce modified versions of the game (known as mods); for example, Gamebots [20] is an example of a mod that allows external programs to control game characters using network sockets.

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

To enhance the immersive experience of the players, we created a custom MOUT soldier model (see Figure 4.10) that conforms more closely to the appearance of a real soldier than the standard UT fantasy fighter models. We also developed a set of hand-signal animations to allow the players to use a small set of military hand signals triggered on key-presses. For our experiments, players executed team behaviors in an uncluttered map environment of corridors and rooms; in the future we plan to extend the scenario to include less structured regions with more clutter.

Representation

Due to the continuous nature of the domain, automatically determining the exact transition points between team behaviors is a difficult problem. While approaching and entering buildings, the players continue moving their bots, changing team behaviors as appropriate for the physical layout. We address this issue by dividing the traces into short, overlapping time windows during which we assume that a single behavior is dominant; these windows are classified independently as described in Section 4.3.2. To recognize team behaviors performed in different physical layouts, it is important for our classifier to be rotationally and translationally invariant; we achieve this by transforming the data in each window into a canonical coordinate frame as described below. More formally, we define:

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

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

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

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

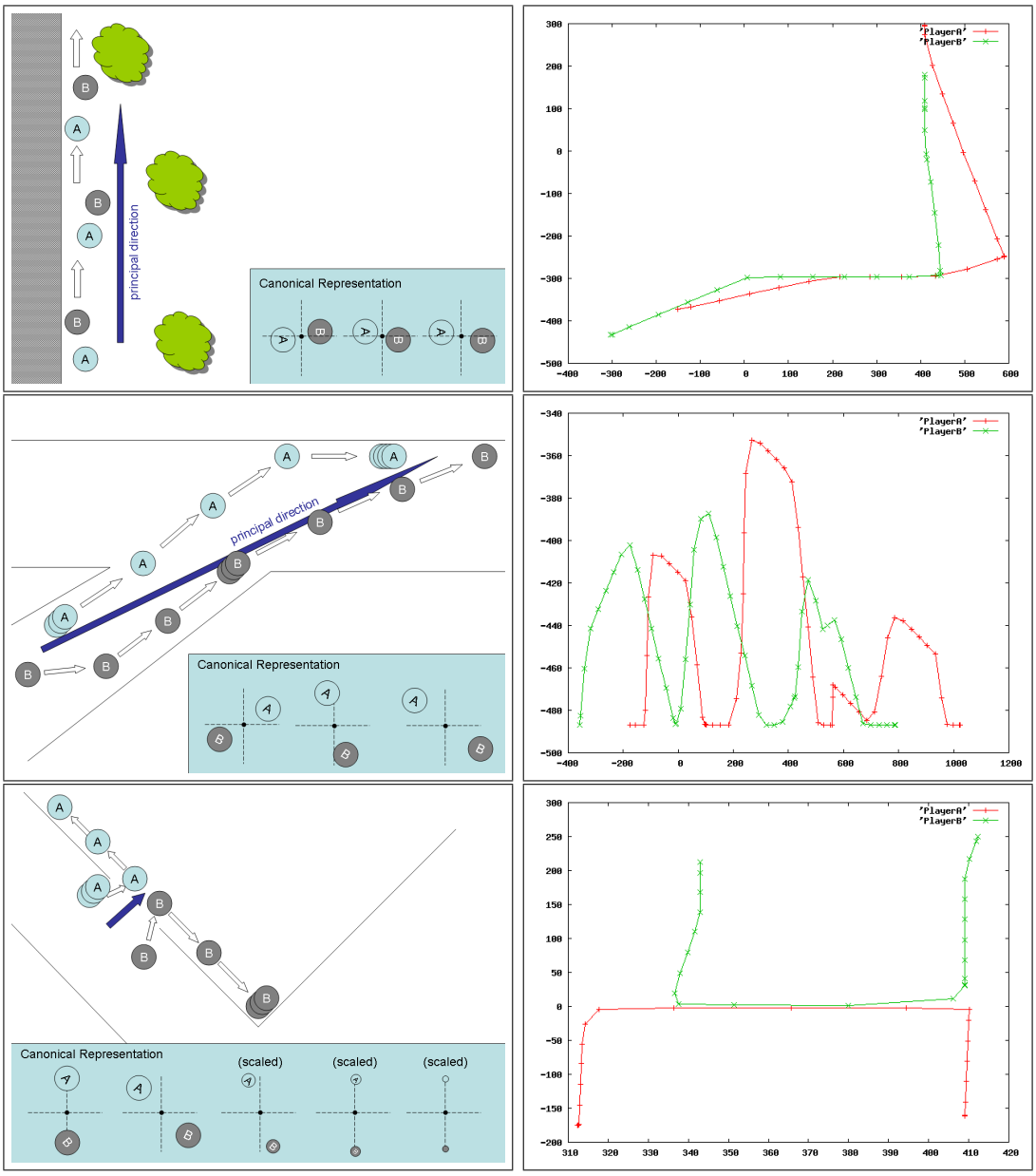

Figure 4.11: Team Behaviors: Stacked Formation (top), Bounding Overwatch (middle), Buttonhook Entry (bottom). Schematics for each behavior, along with the canonical representation for several frames, are depicted in the left column. A sample raw trace for each behavior is shown in the right column; the coordinates of the axes are in Unreal Tournament length units.

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

Classification

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

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

For our problem, given A agents in a team, the observations at time t and window w are the tuple: $o_t = (\mathbf{x}'_{1,w,t}, v_{1,w,t}, \ldots, \mathbf{x}'_{A,w,t}, v_{A,w,t})$. We determine the structure for each behavior HMM based on our domain knowledge. For instance, the stacked behavior can be described using only two states ($N = 2$), whereas we represent the more complicated bounding overwatch behavior using six states connected in a ring. Each hidden state captures an idealized snapshot of the team formation at some point in time, where the observation tuple (in canonical coordinates) is well modeled by a single Gaussian. Rather than initializing the HMMs with random parameters, we use reasonable starting values. These can be polished using expectation-maximization (EM) [10] on labeled training data.

To determine the probability, $Pr(o_{1...T}|\lambda_b)$, of generating the observed data with the model λ_b, we employ the forward algorithm [25] as implemented in the Hidden Markov Model toolbox [22]. We classify each window segment with the label of the model that generated the highest log-likelihood.

4.3.3 Results

To evaluate our automatic recognition method, we developed behavior models for the standard team behaviors used by a 2-person firing team during the approach phase

Table 4.2: Confusion matrix for HMM behavior classification. The ground truth is given in the left column; the classification result is given in the top row. Our hidden Markov model approach achieves good accuracy. Buttonhook entry is often confused with bounding overwatch, as may be expected from similarities in the canonical representation as shown in Figure 4.11.

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

of the MOUT building clearing task: stacked movement, bounding overwatch, and buttonhook entry (see Figure 4.11). A pair of human players performed sequences of team behaviors in our Unreal Tournament urban simulation; position data was recorded from both of the players at 1.5 second intervals using our `TrainingBot` mod (as described in Section 4.3.2). Players executed team behaviors in predesignated sequences, transitioning seamlessly from one behavior into the next, adapting each behavior as needed to the local physical layout (turning corridors, entering rooms). Figure 4.11 (right) shows a raw trace for each behavior; note that even consecutive executions of the same behavior exhibit significant variation as shown by the bounding overwatch behavior. Player traces were divided into overlapping 20 frame (3 second) windows, which were converted into a canonical coordinate frame as described in Section 4.3.2. This process is illustrated in the inset of Figure 4.11.

Table 4.2 presents the classification results (confusion matrix) for the three modeled behaviors; the accuracy of the HMM approach is good, particularly for the stacked formation. Buttonhook entry is sometimes confused with bounding overwatch, as may be expected from similarities in the canonical representation as shown in Figure 4.11.

4.3.4 Summary

The experiments described in the previous section were deliberately designed to omit many spatial and temporal cues that would exist in a real MOUT scenario. This enables us to examine the raw accuracy of our classifier. For instance, we chose not to incorporate the existence or position of static spatial features (e.g., doors and walls) into our observation model, even though this impacts the team's choice of behavior in a more realistic MOUT task. For example, the buttonhook entry behavior is usually performed at doorways or windows whereas stacked movement typically occurs when the team is paralleling a wall.

In our experiments, the sequence of behaviors was arbitrarily chosen to create a variety of behavior transition opportunities. Our HMMs perform recognition at a very low level and do not currently exploit inter-window dependencies. Realistically, there are temporal dependencies between behaviors; for example, in a typical building

clearing operation, there are often long periods of bounding overwatch followed by a single buttonhook entry through a doorway, followed by another period of bounding overwatch. By exploiting higher-level domain knowledge about the building clearing task, we believe that we can improve recognition performance over a full-length MOUT scenario.

4.4 Synopsis of Completed Work

In the next chapter, we demonstrate how the concepts that we explored in completed work on behavior recognition can be unified to create a team behavior recognition algorithm for analyzing scenarios that contain:

- hierarchical, as well as reactive, behaviors;
- dynamic teams that split and merge into different subteams;
- larger teams of agents embedded within a population of uninvolved agents.

Using this algorithm, we can analyze and annotate more complicated MOUT scenarios than the ones discussed above.

Chapter 5

Proposed Work

5.1 Scenario

The goal of the proposed work is to be able to analyze and annotate scenarios such as the one shown in Figure 5.1. In this scenario, there are three groups of agents: a platoon of soldiers moving through the city clearing buildings (Agents 0-7), uninvolved civilians wandering the area (Agents 10-13), and a pair of enemy snipers (Agents 8-9) that emerge from the trees to engage some of the soldiers before retreating back into the park. This scenario is difficult to analyze because the team composition changes over time; to complete the building clearing task, the soldiers must divide into subteams (noted in the figure as Team 4 and 5); four soldiers guard the building while four enter the building in a stacked formation. The enemy snipers coordinate (Team 6) to attack the soldiers but approach and flee from the area separately to avoid attracting attention. While under attack, the soldier subteam (Team 5) further subdivides into Teams 7 and 8; Team 7 counter-attacks and pursues the fleeing attackers while Team 8 guards and rejoins Team 4 when the building clearing is aborted.

Although spatial relationships are an important cue for analyzing team composition, clustering techniques do not accurately predict the team composition; for instance the agents in Team 5 are physically separated to form a guarding perimeter around the building; the sniper agents 8 and 9 deliberately move in isolation to avoid alerting the soldiers to their intentions. The plan recognition problem is complicated by the fact that team tasks are interrupted and resumed; for instance, Team 5 interrupts their guarding to retaliate against the snipers before subdividing into two teams, one of whom resumes guard duty while the other pursues the fleeing attackers.

The desired output of our system are the annotations shown in Figure 5.2:

- the team-to-behavior assignment;
- an agent-to-team assignment over time;
- the set of valid team plan trees (not shown).

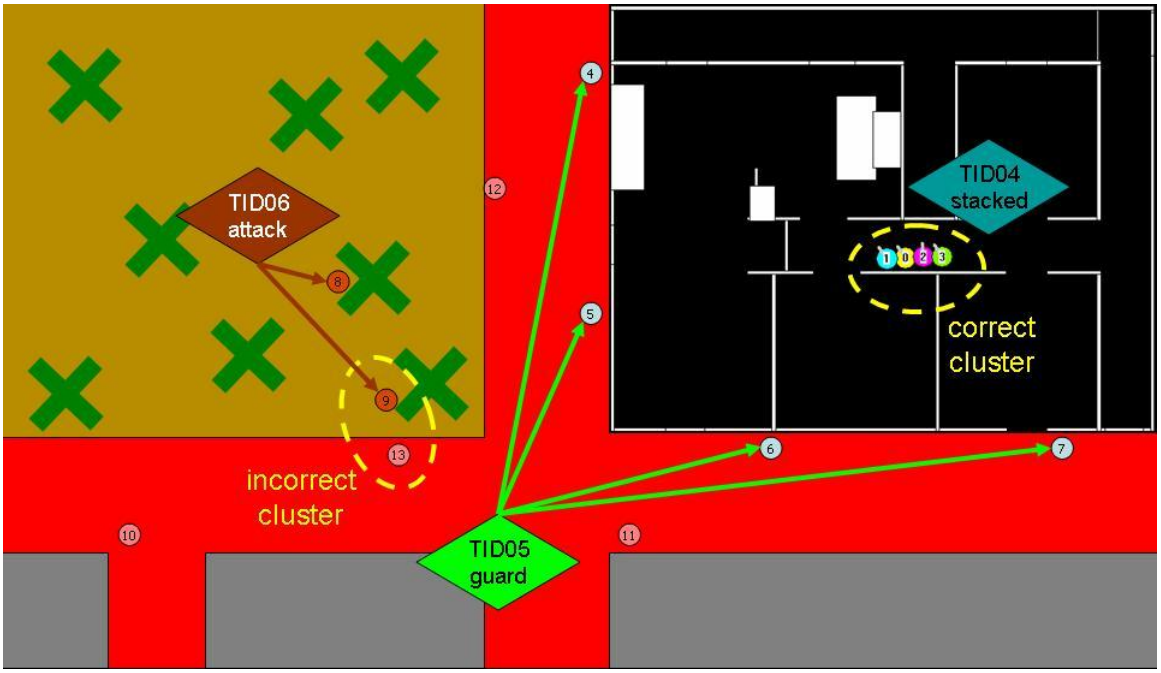

Figure 5.1: MOUT Scenario with Changing Team Composition (inner building layout courtesy of Clemson University MOUT Project). This is an overview snapshot at time $t = 15$ of the agents described by the assignments given in Figure 5.2. Correct team and behavior annotations are marked by diamonds. The output of a naive distance-based spatially clustering algorithm is shown by the yellow dashed lines; note that the naive spatial clustering does not identify Team 5 performing the guarding behavior, although it finds Team 4 who are in close proximity to one another doing the stacked maneuver. It incorrectly groups Agents 9 and 13 into a single team; Agent 9 is actually cooperating with Agent 8 who is hiding behind a tree preparing to snipe. By matching agent groupings against team templates, our proposed algorithm, STABR (Section 5.2) can identify team behaviors missed by the simple spatial heuristic.

	t=10	t=15	t=20	t=25	...
agent00					
agent01		tid04		tid09	
agent02					
agent03	tid01				
agent04			tid07		
agent05		tid05			
agent06			tid08	tid09	
agent07					
agent08	tid02	tid06	tid02		
agent09	tid03		tid03		
...					

	t=10	t=15	t=20	t=25	...
tid01	stacked				
tid02	wander		hide	flee	
tid03	wander		flee	flee	
tid04		stacked	guard		
tid05		guard			
tid06		attack			
tid07			bound	bound	
tid08			attack		
tid09				guard	
...					
...					

Figure 5.2: Agent-to-Team Assignment and Team-To-Behavior Assignment for MOUT Scenario with Changing Team Composition (Figure 5.1)

To analyze the scenario our system requires three components:

1. techniques for annotating agent traces with low-level team actions (discussed in Completed Work);

2. an algorithm for efficiently searching the space of valid agent-to-team assignments;

3. a symbolic plan recognition algorithm adapted for the recognition of team plans.

However, the first part, annotating agent traces with low-level actions, cannot be performed without addressing the second part, determining valid agent-to-team assignments; previous work in this area only handles teams that have constant agent-to-team assignments over time. We introduce a new algorithm, Simultaneous Team Assignment and Behavior Recognition (STABR), that generates low-level action annotations from spatio-temporal agent traces.

5.2 Simultaneous Team Assignment and Behavior Recognition (STABR)

Given a set of spatio-temporal traces for each agent, STABR focuses on small temporal segments over which we assume that a subteam's agent membership and behavior remain constant. This assumption may be violated across behavior transitions, therefore we apply STABR to overlapping time segments.

We process each temporal segment using a sampling-based generate-and-test procedure as follows:

1. Find support for a subteam agent composition and behavior at the start of the temporal window. We use RANSAC to generate hypothesis correspondences between current agent positions and behavior templates. For example, a four-person subteam performing a buttonhook building entry can be characterized by the minimal template of two agents and a doorway. Our hypothesis maps two agents from the scenario to these roles and evaluates the support for this hypothesis by examining the positions of other agents and doorways in the environment. We search our library of templates and identify all of those that have sufficient support based on agent positions at the start of the temporal window. These behaviors and team assignments form a candidate pool that we examine over the entire temporal window. To obtain the most complete candidate pool we employ lenient matching thresholds at this initial stage.

2. The evaluation of each hypothesis in the candidate pool over the entire temporal window can be done in several ways. These can include: 1) direct matching against a spatio-temporal template; this is well suited for actions with less variability; 2) checking whether invariants for the behavior are maintained over the

temporal window; for instance, determining whether agents covering an area maintain lines of fire; 3) motion models, such as spatially-invariant HMMs, which are well suited for recognizing periodic motion, such as bounding overwatch; 4) model-tracing, simulating an agent's behavior over the time window given a particular behavior and local spatial environment. The surviving hypotheses are locally consistent with observed agent activity and contain both an agent-to-subteam assignment as well as a behavior for the subteam.

3. However, there may be conflicting hypotheses within the surviving candidates; for example, a given agent might be assigned to multiple subteams or a given subteam's actions might be described by multiple behaviors. Resolving these assignment conflicts is an interesting research question. If an unambiguous annotation of agent traces is required, we can eliminate hypotheses with weaker support until a consistent interpretation is achieved. Alternatively we can delay conflict resolution and propagate all surviving candidates to the symbolic team plan recognition algorithm where domain knowledge and temporal ordering constraints can be used. In practice, we may choose to prune unlikely candidates at this level to reduce the computational burden at the symbolic level.

5.3 Symbolic Plan Recognition for Teams

At the conclusion of the STABR phase, the agents' activity traces have been mapped to a complete agent-to-team assignment and a team-to-behavior assignment. To annotate the traces with the set of valid team plan trees, we will employ symbolic plan recognition techniques to enforce temporal constraints between behaviors, which are ignored by STABR, and role constraints in team plans. We expect that incorporating information about roles into our planning library should greatly reduce the number of state history hypotheses that our symbolic plan recognition algorithm must consider. This is important because STABR does not necessarily return a unique team-to-behavior assignment for every time step; if there is evidence for multiple behaviors, conflict resolution can be delayed and all valid hypotheses propagated to the symbolic plan recognition layer. By exploiting temporal and role constraints to eliminate state hypotheses, we can keep the number of hypotheses examined tractable, despite ambiguities in behavior assignment. We propose extending the algorithm for fast and complete symbolic plan recognition described in [2] to handle team plans with agent role constraints, as follows:

- We perform temporal consistency checks on observed behaviors by propagating timestamp tags associated with behaviors identified by STABR up through the plan tree. This enables the elimination of behavior sequences that are temporally inconsistent with a given plan.

- Agent role assignments are checked in a similar manner using the agent-to-team assignment tables generated by STABR. For instance, certain team plans might

require that the commander agent be a member of the subteam located at the command post; if an agent-to-team assignment were to violate this constraint its behavior tag would be removed.

- As a post-processing step, we can use heuristics such as cost minimization to select between the remaining valid plans.

Role assignments can be defined in the following ways: 1) required: a designated agent must be assigned to a certain team to make the behavior valid; 2) forbidden: the designated agent cannot be a member of the listed subteam; 3) AND: multiple designated agents must be assigned to the same subteam; 4) OR: at least one of a set of agents must be a member of the subteam. Using symbolic plan recognition, this system should be able to distinguish between very similar scenarios, such as a planned attack on a building (using the *Attack Building* plan) and *Clear Building*, interrupted by *Respond to Attack*.

5.4 Contributions

The main contributions of this thesis are:

- Methods for detecting and identifying complex spatial configurations of agents and environmental landmarks.
- A new algorithm (STABR) for generating low-level action annotations from spatio-temporal traces of physically-embodied agent activity, capable of handling team tasks where team composition changes over time.
- A symbolic plan recognition algorithm to derive valid team plans, based on temporal and role constraints, from sets of team-to-behavior and agent-to-team assignments.

Although we have primarily discussed team behavior recognition in the context of the MOUT domain, we believe that our algorithms should generalize to several other physical domains, such as UAV coordination or multi-person activity recognition for social settings.

5.5 Research Plan

The proposed research activities are structured as follows:

Fall 2005	Implement STABR for MOUT team scenario
Spring 2006	Evaluate STABR on simulated data Develop symbolic team plan recognition Evaluate team plan recognition on UT traces
Summer 2006	System integration Evaluate complete system on simulated and real data
Fall 2006	Write thesis and defend

5.6 Acknowledgments

The work described in Chapter 4.1 was completed with the assistance of the CMU Motion Capture lab, most particularly Michael Mandel, Moshe Mahler, and Dr. Jessica Hodgins. Reid van Lehn and Jason Yates assisted with software development in Unreal Tournament. Susan Eitelman of CHI Systems made many helpful suggestions on potential sources for MOUT data. Dr. Adam Hoover provided videos and data acquired at the Clemson University MOUT training environment. Patrick Worcester from the Potomac Institute for Policy Studies made available general video footage of MOUT soldier training. The members of my thesis committee, Drs. Katia Sycara, Illah Nourbakhsh, Paul Scerri, and Milind Tambe, provided significant feedback on this research.

Bibliography

[1] J. Allen, H. Kautz, R. Pelavin, and J. Tennenberg. *Reasoning About Plans*, chapter 2, pages pp.121–148. Morgan Kaufmann Publishers, 1991.

[2] D. Avrahami-Zilberbrand and G. Kaminka. Fast and complete symbolic plan recognition. In *Proceedings of International Joint Conference on Artificial Intelligence (IJCAI 2005)*, 2005.

[3] D. Ballard and C. Brown. *Computer Vision*. Prentice-Hall, 1982.

[4] J. Barbic, A. Safonova, J-Y. Pan, C. Faloutsos, J. Hodgins, and N. Pollard. Segmenting motion capture data into distinct behaviors. In *Proceedings of Graphics Interface 2004 (GI'04)*, 2004.

[5] B. Best and C. Lebiere. Spatial plans, communication, and teamwork in synthetic MOUT agents. In *Proceedings of Behavior Representation in Modeling and Simulation Conference (BRIMS)*, 2003.

[6] I. Bhandari, E. Colet, J. Parker, Z. Pines, R. Pratap, and K. Ramanujam. Advanced Scout: Data mining and knowledge discovery in NBA data. *Data Mining and Knowledge Discovery*, 1(1):121–125, 1997.

[7] H. Bui. A general model for online probabilistic plan recognition. In *Proceedings of the International Conference on Artificial Intelligence (IJCAI)*, 2003.

[8] C.-C. Chang and C.-J. Lin. *LIBSVM: a library for support vector machines*, 2001. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[9] R. Collins, A. Lipton, T. Kanade, H. Fujiyoshi, D. Duggins, Y. Tsin, D. Tolliver, N. Enomoto, and O. Hasegawa. A system for video surveillance and monitoring. Technical Report CMU-RI-TR-00-12, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, May 2000.

[10] R. Duda, P. Hart, and D. Stork. *Pattern Classification*. Wiley & Sons, Inc, 2001.

[11] M. Fischler and R. Bolles. Random sample consensus: A paradigm for model fitting with application to image analysis and automated cartography. *Communications of the ACM*, 24(6), 1981.

[12] J. Fowlkes, D. Washburn, S. Eitelman, J. Daly, and J. Cohn. Use of haptic displays to enhance training in a virtual environment. In *Proceedings of HCI International 2005*, 2005.

[13] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2000.

[14] A. Hoover, E. Muth, and F. Switzer. Clemson University MOUT Project, 2005.

[15] S. Intille and A. Bobick. Visual tracking using closed-worlds. Technical Report 294, MIT Media Lab, 1994.

[16] S. Intille and A. Bobick. A framework for recognizing multi-agent action from visual evidence. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, 1999.

[17] V. Jirsa. Context-independent team analysis. Presentation to VIRTE meeting, 2005.

[18] M. Jug, J. Pers, B. Dezman, and S. Kovacic. Trajectory based assessment of coordinated human activity. In *Proceedings of the International Conference on Computer Vision Systems (ICVS)*, 2003.

[19] G. Kaminka and M. Tambe. Robust agent teams via socially attentive monitoring. *Journal of Artificial Intelligence Research*, 12:pp.105–147, 2000.

[20] G. Kaminka, M. Veloso, S. Schaffer, C. Sollitto, R. Adobbati, A. Marshall, A. Scholer, and S. Tejada. Gamebots: A flexible test bed for multiagent team research. *Communications of the ACM*, 45(1), 2002.

[21] T. Livak. Collaborative warrior tutoring. Master's thesis, Worcester Polytechnic Institute, 2004.

[22] K. Murphy. The Bayes Net Toolbox for Matlab. *Computing Science and Statistics*, 33, 2001.

[23] D. Pearson and J. Laird. Redux: Example-drive diagrammatic tools for rapid knowledge acquisition. In *Proceedings of Behavior Representation in Modeling and Simulation Conference (BRIMS)*, 2004.

[24] J. Phillips, M. McCloskey, P. McDermott, S. Wiggins, and D. Battaglia. Decision-centered MOUT training for small unit leaders. Technical Report 1776, U.S. Army Research Institute for Behavioral and Social Sciences, 2001.

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

[26] T. Raines, M. Tambe, and S. Marsella. Automated assistants to aid humans in understanding team behaviors. In *Proceedings of the 4th International Conference on Autonmous Agents*, 2000.

[27] J. Rhodes. *Military Operations on Urbanized Terrain (MOUT)*, 1980.

[28] P. Riley and M. Veloso. On behavior classification in adversarial environments. In L. Parker, G. Bekey, and J. Barhen, editors, *Distributed Autonomous Robotic Systems 4*. Springer-Verlag, 2000.

[29] P. Riley and M. Veloso. Recognizing probabilistic opponent movement models. In A. Birk, S. Coradeschi, and S. Tadorokoro, editors, *RoboCup-2001: Robot Soccer World Cup V*. Springer Verlag, 2002.

[30] P. Riley, M. Veloso, and G. Kaminka. An empirical study of coaching. In H. Asama, T. Arai, T. Fukuda, and T. Hasegawa, editors, *Distributed Autonomous Robotic Systems 5*. Springer-Verlag, 2002.

[31] S. Saria and S. Mahadevan. Probabilistic plan recognition in multiagent systems. In *Proceedings of the International Conference on AI and Planning Systems (ICAPS)*, 2004.

[32] C. Schuldt, I. Laptev, and B. Caputo. Recognizing human actions: A local SVM approach. In *Proceedings of International Conference on Pattern Recognition (ICPR 2004)*, 2004.

[33] G. Sukthankar, M. Mandel, K. Sycara, and J. K. Hodgins. Modeling physical capabilities of humanoid agents using motion capture data. In *Proceedings of Autonomous Agents and Multi-Agent Systems (AAMAS)*, July 2004.

[34] G. Sukthankar, M. Mandel, K. Sycara, and J. K. Hodgins. Modeling physical variability for synthetic MOUT agents. In *Proceedings of Behavior Representation in Modeling and Simulation Conference (BRIMS)*, 2004.

[35] G. Sukthankar and K. Sycara. Automatic human team behavior recognition. In *Proceedings of Workshop on Modeling Others from Observations (MOO 2005)*, 2005.

[36] G. Sukthankar and K. Sycara. A cost minimization approach to human behavior recognition. In *Proceedings of Autonomous Agents and Multi-Agent Systems (AAMAS)*, 2005.

[37] G. Sukthankar and K. Sycara. Identifying physical team behaviors from spatial relationships. In *Proceedings of Behavior Representation in Modeling and Simulation Conference (BRIMS)*, 2005.

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

[39] M. Tambe and P. Rosenbloom. Event tracking in a dynamic multi-agent environment. Technical Report ISI/RR-94-393, USC/Information Sciences Institute, 1994.

[40] V. Vapnik. *Statistical Learning Theory*. Wiley & Sons, Inc, 1998.

[41] G. Zu and Z. Zhang. *Epipolar Geometry in Stereo, Motion, and Object Recognition*. Kluwer, 1996.