A Cost Minimization Approach to Human Behavior Recognition

Gita Sukthankar
Robotics Institute
Carnegie Mellon University
5000 Forbes Ave.
Pittsburgh, PA
gitars@cs.cmu.edu

Katia Sycara
Robotics Institute
Carnegie Mellon University
5000 Forbes Ave.
Pittsburgh, PA
katia+@cs.cmu.edu

ABSTRACT

This paper presents a cost minimization approach to the problem of human behavior recognition. Using full-body motion capture data acquired from human subjects, our system recognizes the behaviors that a human subject is performing from a set of military maneuvers, based on the subject's motion type and proximity to landmarks. Low-level motion classification is performed using support vector machines (SVMs) and a hidden Markov Model (HMM); 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. We demonstrate that our system is robust to action classification errors and deviations by the human subject from the expected set of behaviors. Our approach is well suited for incorporation into synthetic agents that cooperate or compete against human subjects in virtual reality training environments.

Categories and Subject Descriptors

I.5 [**Pattern Recognition**]: Miscellaneous; I.2.8 [**Problem Solving, Control Methods, and Search**]: Plan execution, formation, and generation

General Terms

Algorithms

Keywords

plan recognition, motion capture, support vector machines, dynamic programming

AAMAS'05, July 25-29, 2005, Utrecht, Netherlands.

1. INTRODUCTION

In cooperative and competitive domains, it is often important for agents to be able to reason about the future behavior of their fellow agents based on past observations; this knowledge can be incorporated into the agent's planning to guide its future actions. This process of *plan recognition* is especially critical in adversarial domains where it is advantageous for opponent agents not to reveal their intentions and in cooperative domains in which the cost of communication is too high for agents to synchronize their plans.

An additional complication is introduced when humans are added to the system as teammates, and agents must assist humans or participate in mixed-initiative teamwork tasks. In such cases, communication is even more expensive due to the difficulties of human-agent communication (lack of common vocabulary, need to include extra technologies such as natural language parsing or speech recognition/generation). By recognizing human behavior directly from observation traces, an agent can potentially function as a more effective partner, even in the absence of communication, and a more formidable adversary.

In this paper, we introduce a new approach to *human behavior recognition* based on the use of cost minimization to select the most parsimonious explanation for sequences of physical actions performed by the human. By combining information about the human's motion with proximity to geographic landmarks, we can disambiguate between different types of actions performed in similar geographic locations. We demonstrate the approach in the domain of MOUT planning (Military Operations in Urban Terrain). MOUT is a rich domain for behavior recognition since synthetic MOUT soldiers have opportunities for adversarial maneuvers (ambushing enemies), assistive plans (rescuing wounded teammates), and both loosely and tightly coupled teamwork tasks (performing surveillance, moving in formation). Our technique is well suited to handle behavior interruptions which occur in the dynamic MOUT domain and to account for incorrectly executed behaviors performed by novice human subjects.

2. BACKGROUND

2.1 Problem Description

The following domain properties make behavior recognition challenging:

large state spaces: Although traditional AI plan recognition has been demonstrated in small, closed world domains [1], real-world domains with large state spaces make exact inference methods slow or intractable.

noisy observations: Robotic domains are plagued by noisy observations which make the problem of recognizing the robot's state difficult. Since successful behavior recognition relies on being able to correctly classify sequences of states, the problem becomes very difficult; these domains are often modeled using dynamic Bayesian networks [4].

behavior interruption: In dynamic domains, the agent will interrupt behaviors before completion due to the need to immediately respond to unforeseen actions (e.g., an enemy's surprise attack). Also in continuous domains, such as Robocup [9], even segmenting fully completed behaviors can be difficult problem if there are no obvious transitions marking the transition of one behavior into the next.

hierarchical behaviors: Plans (or behaviors) can be structured in a hierarchical way such that the probability of executing an action is not only dependent on the current state of the agent but also on the current level of plan hierarchy. Models such as PSDG (Probabilistic State-Dependent Grammar) [17] or AHMEM (Abstract Hidden Markov Memory Model) [4] have been developed to handle these types of behaviors.

unmodeled actions: In open world environments, there is no guarantee that the agent is limited to executing domain specific actions. For instance, our human subjects can decide to spontaneously tie their shoes in the middle of executing a sequence of military maneuvers.

behavior deviations: The human's execution of behaviors might deviate slightly from the officially recognized military strategy, either due to individual differences or errors. Recent work [13] also addresses the problem of detecting anomalous user behavior.

In this paper, we focus on handling the effects of behavior interruptions, unmodeled actions, and human behavior deviations. Although there is uncertainty about the motion being executed by the human subject, we assume that the (x,y) position of the person is very accurately measured by the motion capture setup as described in Section 5.3.

Our behaviors are represented using directed acyclic graphs as shown in Figure 1; behaviors are often interrupted and resumed. For our MOUT domain, we have not made an attempt to create a hierarchy of plans and sub-plans; each of our behaviors is atomic and represents a complete sequence of recognizable actions. Any behavior can legally follow any behavior, assuming the state transition is valid. Behavior deviations occur when the detected transition is not in the library of valid behaviors, either because it's part of an unmodeled action or the subject has deviated in the execution of a behavior.

Our representation of the MOUT domain has a relatively small state space (about $11,000$ state state transitions) due to the proximal feature selection mechanism incorporated into our simulator; our system hashes state transitions for constant-time retrieval. To scale our system to larger state spaces, we would employ a tree-based technique, such as the one described by Kaminka and Avrahami [12].

2.2 Related Work

Related work on the problem of human behavior recognition has emerged from three communities: computer vision researchers who have examined the problem of activity inferencing with temporal constraints [21], graphics researchers who address the problems of clustering and segmenting unlabeled motion capture data [2, 11] and artificial intelligence plan recognition researchers [1, 4, 6, 8, 10, 17, 23] who have traditionally focused on behavior inference mechanisms. No single dominant approach has emerged that is both computationally efficient and deals well with all of the difficult aspects of the problem; typically researchers adapt their approaches to the characteristics of the domain.

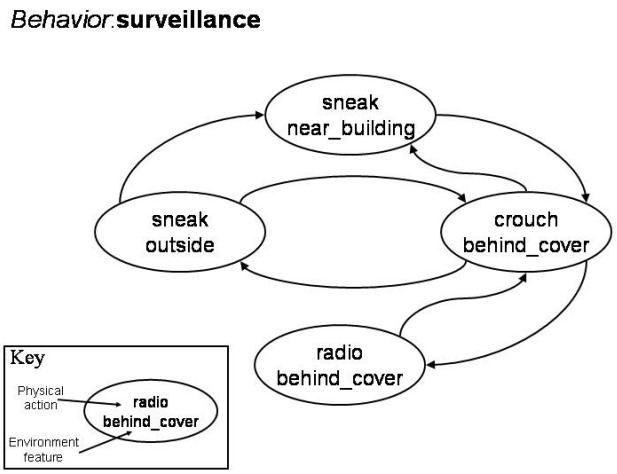

Figure 1: MOUT Behavior Representation: Surveillance. The *surveillance* behavior is used when a soldier wants to examine a building from the outside in preparation for entering the building, either as part of an attack or a building clearing operation. Any of the states listed in the diagram are valid starting points for the behavior; there is no single state transition that must always occur at the start of every surveillance behavior.

Several research groups have examined the problem of recognizing single person indoor activities (household or office tasks) from movement trajectories extracted from camera data [16] or wireless sensor signal strength [25]. These approaches define actions based on the closest specific geographic landmarks such as *Room1*, *Hallway2*, or *Refrigerator* and infer high-level goals from sequences of low-level actions. In our work, we introduce general categories of environmental features that allow us to generalize from actions performed in different places. Also by using motion capture apparatus to measure the human's movement, we can accurately classify the person's mode of movement (walking, crouching, probing) without relying on velocity difference measurements.

3. MOUT DOMAIN

In this paper we examine the problem of behavior recognition within the MOUT (Military Operations in Urban Terrain) infantry domain. In MOUT scenarios, platoons of soldiers have 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 (blocked roads, booby-trapped zones) without direct guidance from the chain of command. Individual soldiers must coordinate with their teammates to move through hazardous areas in defensive formations (e.g., bounding overwatch). Often it is unclear whether people moving around the combat zone are civilians or enemy snipers. Spatial environmental features (buildings, intersections, doorways) are important features which influence planning [3]; soldiers also must be able to quickly execute reactive behaviors in the face of immediate threat. Previous work on simulating MOUT agents has

addressed the creation of accurate physical capability models for simulated MOUT soldiers [22], but has not tackled the problem of behavior recognition.

Although there are many interesting plan recognition problems possible in the MOUT domain, here we focus on the problem of recognizing the behavior of individual MOUT soldiers in the context of physical actions and spatial environment features. MOUT soldier behaviors include scouting, ambushing, retreating, searching for hazards, and clearing rooms; we also model less structured behaviors appropriate for civilians in combat zones such as fleeing and hiding. In future work, we plan to extend our approach to the recognition of team behaviors, but that is not addressed in this paper.

4. 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 1.

- *Physical actions* { walk, run, sneak, probe, wounded_movement[1], rise, crouch } are physical movements (Figure 4) classified from human motion capture traces using a support vector machine (SVM) action classifier as described in Section 5.5. 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 5.8 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. 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 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.

[1]In this paper, we only consider leg wounds, in which the human subject is limping with either their right or left leg. In the future, we plan to 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.

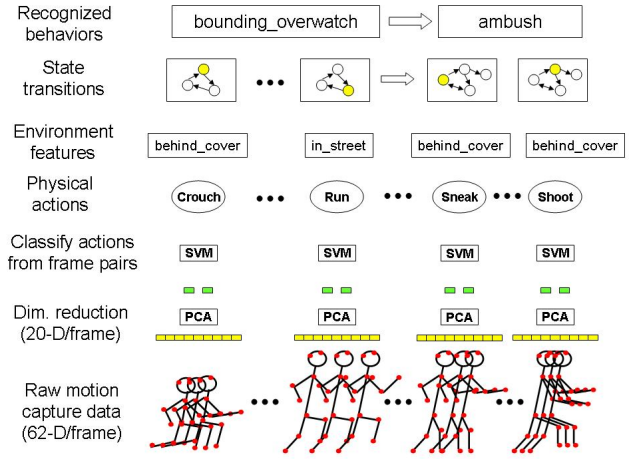

Figure 2: System diagram. The purpose of our system is to recognize and produce an accurate description of physical *behaviors* performed by a single human subject engaged in a MOUT scenario. The human's *physical actions* are recorded using a motion capture system as described in Section 5.3 and classified by our classifier (Section 5.5). These actions are used by an environment simulator (Section 5.8) to generate *state* transitions; these sequences of state transitions, or *observation traces*, are used as the input to the behavior recognition system.

5. METHOD

5.1 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 as described in Section 5.3 and classified by our classifier (Section 5.5). These actions are used by an environment simulator (Section 5.8) 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.

Although the current implementation of our system operates in an off-line mode (data from the subject is recorded in the motion capture lab and then processed off-line), we plan to integrate the system into a virtual environment that trains humans to perform physical MOUT tasks. Other training environments have been developed for military tasks [14, 19], but those systems have focused on the cognitive and language aspects of the task rather than the physical maneuvers.

5.2 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 3); the process of recording data is described in Section 5.3. This produces a stream of high dimensional data describing the human's trajectory over time while performing specified physical actions. This 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

Figure 3: A subject wearing a retro-reflective marker set in the CMU motion capture laboratory.

(Section 5.4). Pairs of reduced-dimension motion capture frames are used as input for our action classifiers (Section 5.5); a hidden Markov model is used to post-process the raw classifications to reduce the occurrence of spurious state transitions as described in Section 5.6.

To test our behavior recognition, we use a simulator (Section 5.8) 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 (Section 5.7).

5.3 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 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/`.

5.4 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) [7] 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 [2].

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:

$$\theta = \mathbf{W}^T(\mathbf{x} - \mu), \tag{1}$$

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; dimensionality reduction of incoming motion capture frames is achieved by the efficient linear projection described by Equation 1.

5.5 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 [24]. Support vector machines (SVM) are a supervised binary classification algorithm that have been demonstrated to perform well on real-world visual pattern classification tasks [20]. 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,

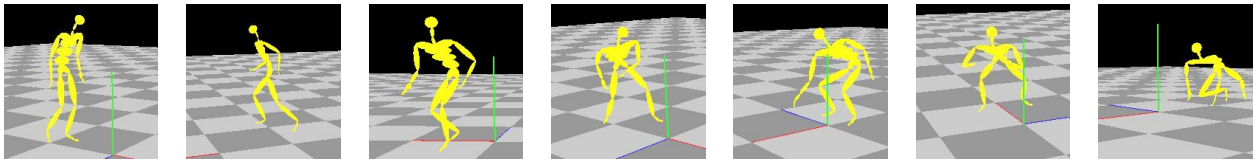

Figure 4: Representative poses for the subject's physical actions.

$K(\mathbf{x}_i, \mathbf{x}_j) \equiv \phi(\mathbf{x}_i)\phi(\mathbf{x}_j)$, 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 [5] 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). 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 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).

5.6 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 [18] as implemented in the Hidden Markov Model toolbox [15].

Our model is parameterized by the following:

- $N = 7$, the number of hidden motion states (walk, run, sneak, probe, wounded_movement, probe, crouch, rise).
- $\mathbf{A} = \{a_{ij}\}$, 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 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.

Table 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 down 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%

5.7 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 in Section 5.8. 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.

$$B^q_{t+1} = \min_{\forall p}\{B^p_t + T_{p,q}\}$$

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

B^q_{t+1} 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^p_t combined with the cost transition function, $T_{p,q}$.

5.8 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 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.

6. 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?

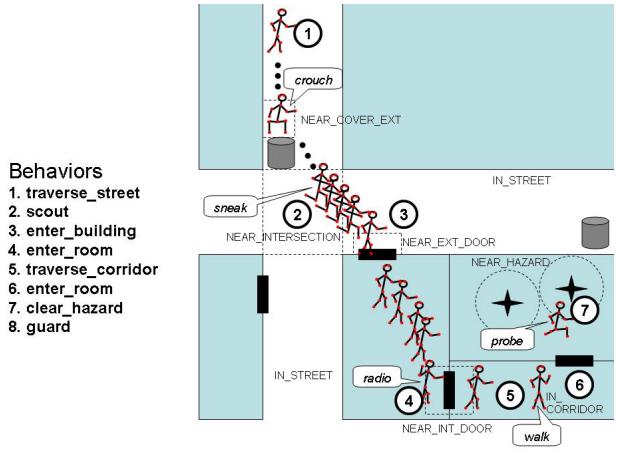

Figure 5: MOUT Scenario: Building Clearing. An overhead view of the schematic used by our simulator to generate observation traces for an example building clearing scenario. The standard followed military procedure would be to use the following sequence of behaviors: traverse_street, scout, enter_building, 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).

6.1 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 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.

6.2 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 5.7, 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 remain-

ing 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 is 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.

6.3 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 our 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.

7. DISCUSSION

We believe that the efficacy of behavior recognition algorithms should be evaluated within an application-specific context. Our interest in behavior recognition is focused in the context of three problems often found in virtual training environment applications: (1) mixed initiative agent-human teamwork; (2) effective modeling of human opponents to make the training applications challenging and instructive; (3) the detection and correction of novice human errors.

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 5.7, 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.

8. CONCLUSION

This paper introduces a cost minimization approach to the problem of behavior recognition and demonstrates how it can be used to recognize physical behaviors even with imperfect action classification. Once the classifier has been trained, the action classification plus behavior recognition executes within seconds making it suitable for online applications such as team coordination and opponent modeling.

9. ACKNOWLEDGMENTS

The authors would like to thank Michael Mandel for assisting with the motion capture data collection, Rahul Sukthankar for his thoughtful comments, and the CMU Motion Capture lab personnel for assisting with data post-processing. This work has been supported by ONR grant N00014-02-1-0438.

10. REFERENCES

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

[2] 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 (GI'04)*, 2004.

[3] 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.

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

[5] 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`.

[6] E. Charniak and R. Goldman. A Bayesian model of plan recognition. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, 1993.

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

[8] R. Goldman, C. Geib, and C. Miller. A new model of plan recognition. In *Proceedings of Uncertainty in Artificial Intelligence (UAI)*, 1999.

[9] K. Han and M. Veloso. Automated robot behavior recognition applied to robotic soccer. In *Proceedings of the IJCAI-99 Workshop on Team Behavior and Plan Recognition*, 1999.

[10] M. Huber, E. Durfee, and M. Wellman. The automated mapping of plans for plan recognition. In *Proceedings of Uncertainty in Artifcial Intelligence (UAI)*, 1994.

[11] O. Jenkins and M. Matarić. Deriving action and behavior primitives from human motion data. In *Proceedings of 2002*

IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2002.

[12] G. Kaminka and D. Avrahami. Symbolic behavior-recognition. In *Proceedings of Workshop on Modeling Other Agents from Observations (MOO)*, 2004.

[13] L. Liao, D. Fox, and H. Kautz. Learning and inferring transportation routines. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, 2004.

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

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

[16] N. Nguyen, D. Phung, S. Venkatesh, and H. Bui. Learning and detecting activities from movement trajectories using the hierarchical hidden Markov model. In *Proceedings of IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2005.

[17] D. Pynadath. *Probabilistic Grammars for Plan Recognition*. PhD thesis, University of Michigan, 1999.

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

[19] J. Rickel and W. L. Johnson. Virtual humans for team training in virtual reality. In *Proceedings of the Ninth International Conference on AI in Education*, pages 578–585. IOS Press, 1999.

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

[21] Y. Shi, Y. Huang, D. Minen, A. Bobick, and I. Essa. Propagation networks for recognizing partially ordered sequential action. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2004.

[22] 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)*, 2004.

[23] M. Tambe and P. Rosenbloom. RESC: An approach to agent tracking in a real-time dynamic environment. In *Proceedings of the International Conference on Artificial Intelligence (IJCAI)*, 1995.

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

[25] J. Yin, X. Chai, and Q. Yang. High-level goal recognition in a wireless LAN. In *Proceedings of Twentieth National Conference on Artificial Intelligence Conference (AAAI)*, 2004.