Modeling Physical Capabilities of Humanoid Agents Using Motion Capture Data

Gita Sukthankar
Robotics Institute
Carnegie Mellon
gitars@cs.cmu.edu

Michael Mandel
Computer Science
Carnegie Mellon
mmandel@cs.cmu.edu

Katia Sycara
Robotics Institute
Carnegie Mellon
katia+@cs.cmu.edu

Jessica Hodgins
Computer Science
Carnegie Mellon
jkh@cs.cmu.edu

Abstract

In this paper we demonstrate a method for fine-grained modeling of a synthetic agent's physical capabilities—-running, jumping, sneaking, and other modes of movement. Using motion capture data acquired from human subjects, we extract a motion graph and construct a cost map for the space of agent actions. We show how a planner can incorporate this cost model into the planning process to select between equivalent goal-achieving plans. We explore the utility of our model in three different capacities: 1) modeling other agents in the environment; 2) representing heterogeneous agents with different physical capabilities; 3) modeling agent physical states (e.g., wounded or tired agents). This technique can be incorporated into applications where human-like, high-fidelity physical models are important to the agents' reasoning process, such as virtual training environments.

1. Introduction

Much research attention [3, 6, 11, 19] has been devoted to the problem of cognitive modeling to create agents with reasoning skills similar to humans. For applications such as training environments, computer games, and military simulations with computer-generated forces, it is not enough to create agents that are merely successful at completing tasks; we seek to develop agents that can perform in a human-like fashion. Ultimately, the user must step outside the simulation environment into the real-world and transfer the skills learned from interactions with agents to human teammates and opponents. The better our agents can function as "virtual humans", the more successful a training experience the user will have.

In many domains, our reasoning is constrained and influenced by an internal model of our physical capabilities. For instance, when guarding an opponent in a basketball game, our choice of action depends not only on tactical and strategic concerns, but also on what we believe our abilities to be relative to our opponents' physical speed, agility, and endurance. This paper addresses the problem of modeling human physical skills and incorporating this model into the planning process of a humanoid agent, enabling it to predict the effect of its actions and respond realistically to those of its teammates and opponents. We believe that fine-grained modeling of an agent's physical capabilities is an important aspect of planning in sports domains (e.g., basketball, football, hockey) as well as in certain combat scenarios, such as small-unit urban operations. We focus on the following issues:

- modeling humanoid physical capabilities;
- incorporating knowledge of the agent's physical aptitude into planning;
- modeling variations in physical aptitude among individual agents and heterogeneous agent categories;
- representing different physical states (e.g., wounded and tired agents).

We believe that these questions are as vital to creating human-like agents in active physical domains as cognitive modeling is for creating agents in more "cerebral" domains.

2. Framework

When selecting actions for agents, the following important question often arises: how much time does it actually take the agent to accomplish each action? For instance, is it faster for an agent to move to a point behind its current position by turning around and running forward, or by moving backwards without changing its direction? Either sequence of actions will ultimately get the agent to the same (x,y) location so there is no reason for a naïve planner lacking a model of human behavior to prefer one plan over the other. *A priori*, the planner might assume that the plan requiring fewer discrete actions (moving directly backward) should be preferred over the slightly "longer" plan (turning, moving forward, turning again), even though for a real human that "shorter plan" would take longer to execute. Human behavior is often asymmetric in a way that computer generated plans are not. Humans have a dominant side (eye, hand, foot) that leads them to perform actions such as manipulating objects, jumping, and kicking in different ways. Similarly, in path planning, humans often exhibit the trait of taking one path from *A* to *B* and a different (possibly longer) path from *B* to *A*, violating the predictions of typical robot path planning algorithms.

We propose the following general framework for building a cost model of human actions:

1. gather exemplars of domain-specific behavior (e.g., running, dribbling, sneaking) using a human motion capture system;
2. construct a motion graph that enables rapid generation of animation sequences for each behavior;
3. identify an appropriate cost function for scoring candidate animation sequences based on elapsed time and distance criteria;
4. precompute a cost map that expresses the variation in cost for executing a particular behavior;
5. given a set of equivalent goal-achieving behaviors, select the one that minimizes the total cost.

The basic assumption underlying our approach is that motion sequences of behaviors that are implausible or difficult for humans to execute cannot be constructed without incurring a substantial penalty in the cost function. Our method requires a fairly complete basis set of data for every behavior that the agent is allowed to execute, otherwise the cost function will falsely penalize behaviors possible for a human to perform but not well represented in the data set. The remainder of this section presents details about each aspect of the model construction.

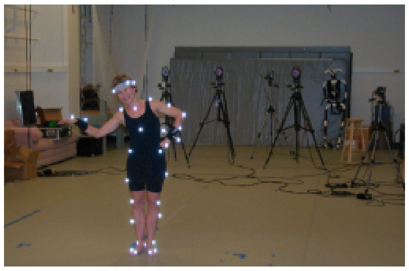

Figure 1. A subject wearing a retro-reflective marker set in the CMU Motion Capture Laboratory.

2.1. Data collection

Human motion data is captured using a Vicon optical motion capture system with twelve cameras, each 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'$\times$24'. A subject is shown in the motion capture laboratory in Figure 1. The motion is generally captured in long clips (over a minute) to allow the subjects to perform natural transitions between behaviors and is represented by a skeleton that includes the subject's limb lengths and joint range of motion (computed automatically during a calibration phase). 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.

We manually annotate the data to label individual domain-specific behaviors, such as walking, probing, inspecting and covering. Because we aim to capture a full spanning set of motion for a particular behavior, our manual labelling is made tractable with long sessions where each take is aimed at capturing the full range of a single behavior. Often capturing multiple behaviors in one take is desirable for future synthesis of natural transitions between behaviors or when a particular domain's fine-grained behaviors are difficult to capture individually. In these cases, semi-automated techniques have been proposed [2] to assist in annotation that require a relatively small portion of the database to be manually labelled.

2.2. Motion capture graph

To explore the full physical capabilities of a human for a particular behavior in a domain-appropriate, simulated environment, we must move beyond the raw motion data. The data alone merely allows playback of the subject's performance from the capture session in an environment that is fixed in size and layout. Often we would like to reuse the motion data in new environments with full control over the character's navigation of its environment and the order in which various actions are performed.

Motion graphs were introduced [1, 12, 13] to provide a solution to this need for control by automatically discovering the ways in which the original data could be reassembled to produce visually smooth motion. Instead of being restricted to a linear playback of the motion clips, the algorithm automatically produces choice points where streams of motion can be smoothly spliced to create novel motion sequences. Individual animation frames act as nodes in the graph, and the choice points act as directed arcs indicating a possible transition between two frames. Below, we discuss how searching the graph structure enables us to compute a cost for navigating between two points in a simulated

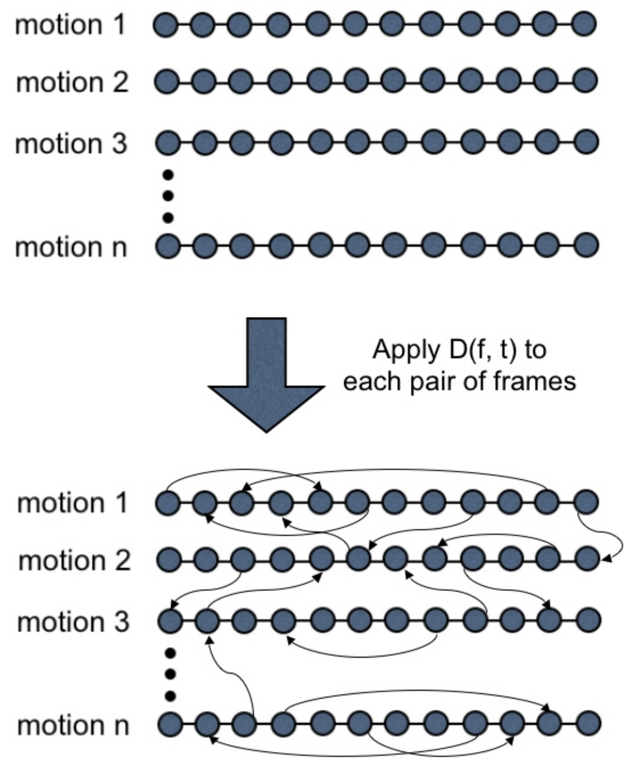

Figure 2. The motion graph is constructed by finding the distance between each frame in the database. Edges are added to the graph when $\mathbf{D}(\mathbf{f},\mathbf{t})$ is below a specified threshold. An edge between two nodes indicates that a transition may be made to smoothly splice the corresponding streams of data together.

environment.

2.2.1. Computing a distance metric The key to building a motion graph is defining an appropriate distance metric between pairs of frames in the database. The metric must ensure that the position and velocity of each body part be sufficiently similar for two pieces of motion to be joined. Since the data is captured in the global coordinate system of the capture area, some care needs to be taken when comparing motion captured in different regions of the space. It is important to note that data in the global coordinate system may be translated along the ground and rotated about the human's vertical axis without affecting any important qualities of the motion. Because poses should be compared in the canonical frame of reference, the algorithm must recover this alignment transformation.

Our distance metric is modeled most closely after the one introduced by Kovar *et al.* [12]. The distance metric, $D(f,t)$, is computed between frame f and t in the database using the joint positions of the poses in a small transition time window starting at f and t. The purpose of computing the metric over a window of frames is to ensure that velocity mismatches between the two frames are penalized in the calculated metric. A coordinate transformation, $T(f,t)$, is computed to align f and t in the first frame of the window so each pose has matching translation in the ground plane and vertical axis rotation. The metric is computed as follows:

$$D(f,t) = \sum_{i=0}^{\mathrm{WS}} \sum_{j=0}^{J} w_j \left\| p(f+i,j) - (T(f,t)p(t+i,j)) \right\|^2 \quad (1)$$

where WS is the size of the transition window, J is the number of joints in the character, w_j allows for weighting the importance of each joint, $p(f,j)$ is the global position of joint j at frame f in x,y,z coordinates, and $T(f,t)$ is the coordinate transformation that maps frame t to frame f.

An edge connecting two nodes (frames) in the motion graph is added whenever the distance between the two frames is below a specified threshold. This threshold may be varied to balance the transition smoothness with the size and versatility of the graph. Typically, it is empirically set so that transitions exhibit no visual discontinuities nor physical anomalies. For rendering at runtime, transitions are blended over a small time window using a sinusoidal “ease-in/ease-out” function to smooth the discontinuity between the two motion streams.

Once the distance metric has been calculated between all frames, we employ a pruning strategy adapted from [12,13] to remove troublesome edges from the graph; this operation is to avoid the problem of generating paths through the motion graph that cause the character to get stuck in a small subset of the motion database. We remove sinks and dead-ends in the graph by keeping only the largest strongly-connected component (SCC) of the graph, and this can be performed efficiently [17].

2.3. Evaluating actions

We now present a metric to score the cost for a human to move through an environment while performing a particular behavior, where each path is generated by a motion graph. The full set of paths is sampled using a stochastic algorithm which converges on a final cost map for the space. An extension to the algorithm is also presented to handle obstacles that may be present in a real environment.

2.3.1. Scoring animation sequences Given a motion capture graph, we can generate candidate sequences of the character performing each behavior that are visually smooth and lifelike. These sequences, or paths, consist of a series of frames and transitions created by traversing the motion

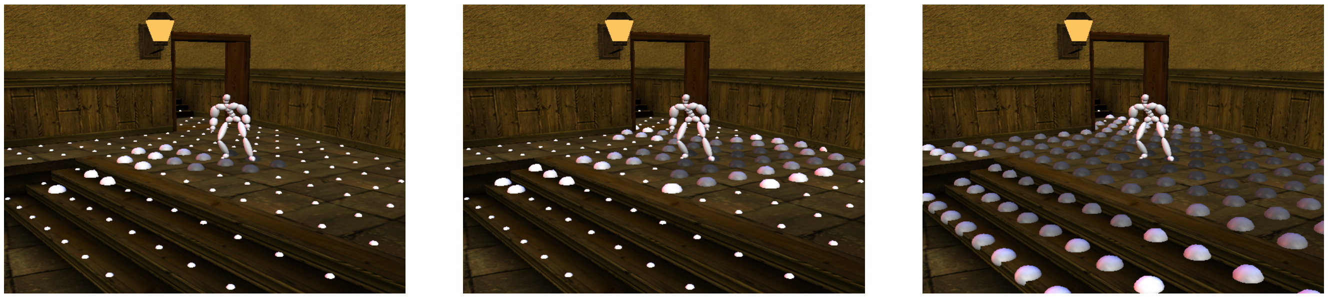

Figure 3. The Monte Carlo sampling of paths through the motion graph iteratively approximates the cost map. Light and dark color spheres indicate high and low physical cost according to our metric. The first panel shows a single path through a motion graph while the subsequent panels show how additional iterations contribute to convergence. As yet unexplored regions of the world are indicated by smaller spheres.

graph. Each sequence represents a possible chain of motions that the human could have executed to reach the goal position. To compute the cost of executing a sequence, we examine two criteria: (1) time cost, which is directly proportional to the path length in frames; (2) goal achievement—how well the path achieves the desired goal state. The cost metric is

$$C = F + \alpha \| r(x,y) - g(x,y) \|^2 \tag{2}$$

where C is cost, F is path frame length, $r(x,y)$ is the character's position, and $g(x,y)$ is desired goal position. The path cost is dominated by F which represents the time cost required for the character to reach a given location; the second term penalizes paths that terminate farther away from the desired goal. Increasing the discretization of the environment reduces the magnitude of the penalty term since all paths that fall within a grid cell are very close to the goal but increases the time required to generate the cost map. The experiments described in Section 3 were run with $\alpha = 0.0$. Note that this cost is complementary to the distance metric that we use when assembling the motion graph; because we know that every sequence is guaranteed to be smooth and human-like within a certain threshold, we omit smoothness from our path scoring criterion.

2.3.2. Calculating the cost map To extract the cost of performing a behavior for a given set of constraints, we "unroll" the motion graph to create a cost map over the environment for a given behavior. The map size should be large enough to accommodate the region over which the planner may require cost estimates, and sampled at sufficient resolution to ensure that discretization errors do not eliminate solutions. For example, a map covering a 50'×50' area at a resolution of 10×10 corresponds to a grid with 100 equally-spaced cells, each 5'×5' in size.

The algorithm stochastically samples the set of valid paths to move through the environment using the selected behavior and annotates each cell with the cost of the best path through that cell. Edges in the graph may optionally be disabled if the planner wishes to enforce constraints such as a maximum velocity for the character. The basic steps of the algorithm are as follows:

- Disable edges in the graph according to constraints provided by the planner, eliminating illegal actions.
- Estimate a reasonable maximum search depth of the graph (dependent on desired map size) to bound the search. Paths greater than the estimated maximum depth are not considered.
- Perform a Monte Carlo sampling of the motion path space, updating each cost map cell's score (according to the metric described in Section 2.3.1) along the candidate paths. Random paths are repeatedly generated until the cost map converges to a stable configuration.

We adopted this strategy due to the high branching factor of the motion graph and the necessity to simultaneously evaluate every potential cell in the cost map's space. Since a real human's movement is highly variable, making general early pruning decisions is difficult. Exhaustively searching the graph using breadth-first search is prohibitively expensive, and more directed search strategies (e.g., A*) are inappropriate since a search would need to be initiated for each cost map cell. If computation time is limited, the Monte Carlo search also has the desirable property that it may be terminated early to produce an approximation of the final cost map. Figure 3 shows a visualization of the search process from within our simulator.

Our previously computed cost maps are invariant to an agent's position and orientation because they can be embedded with the agent anywhere in an environment. However, they do not reflect the true physical cost for the agent in the presence of obstacles. In our solution, if there are obstacles in the environment, candidate paths that enter obstructed regions are pruned to eliminate physically impossible paths.

For a complex environment, the simulator must enforce behavioral constraints to ensure that candidate paths do not violate the terrain requirements (e.g., chasms must be crossed with jumping or crawlways traversed with crawling). These constraints are checked during the search to eliminate invalid paths.

In presence of obstacles, the cost map is no longer invariant to starting position and orientation since its costs are only correct for a character in the same position relative to the obstacles. One solution would be to cheaply compute a small low resolution cost map at run-time to address the current state of the character that the planner is considering; another idea is to pre-compute cost maps at choice points in the environment where the planner is unlikely to be able to use the previously computed generic cost maps. For a very complicated environment, the cost model should be computed at run-time for a few regions of interest rather than building a cost map over the entire environment.

3. Results

Here we present cost models from different domains and demonstrate possible applications to the agent's decision-making process.

3.1. Comparative Cost Models

One of the domains we are currently investigating is MOUT (Military Operations in Urban Terrain) where soldiers perform small-unit tactical maneuvers. For our preliminary cost model of the MOUT soldier, we captured data from a human subject (male college student) performing various behaviors required for the execution of team tactical maneuvers: walking, running, sneaking, taking cover, rising from the ground, using communications equipment, hand signalling team members, inspecting areas, and probing for booby-traps. Using the human data and the algorithms described in Sections 2.3.1 and 2.3.2, we built a motion capture graph that generates smooth animations of the MOUT soldier performing various behaviors.

In the first scenario, we model the cost of the MOUT soldier running around an obstacle, to determine the tradeoffs between having the soldier run around a hazardous area vs. other plans of action (sneaking or probing for booby-traps). We restrict the animation system to using behaviors labeled with the "running" descriptor and represent the hazardous area as an obstacle to prevent the system from simulating paths that cross the area.

We compare our motion-capture based cost model to the popular distance-based cost model used by robotic path planners. Figure 4 shows a cost map generated using grassfire path-planning [4] on a discretized version of the space, along with the "running" cost map created with the human motion capture data and the cost map of the "sneaking" behavior. The cost map generated by grassfire is shaded uniformly dark to light with distance from starting position. This reflects a belief that moving in any direction is equally feasible. Unfortunately this approximation is not consistent with human behavior, and agents that use this model are easily anticipated by human opponents.

3.2. Modeling Opponent Agents

Not only can the agent improve its own plans with a more refined cost model, but it can also anticipate its opponents' actions using the cost model. We discuss how our model could be applied in the basketball domain for a single agent attempting to bypass a blocker and score. To generate our motion graphs, we captured data from a human subject (male college student) performing basketball maneuvers such as dribbling, pivoting, spinning, and running. The offensive agent is required to use the dribbling behavior to move the ball down the court, whereas the defender can use any form of non-dribbling movement. The offensive agent's goal pose is the shooting stance; it can shoot from any square with a varying chance of success, which we model as being inversely proportional to the distance to the basket. Which location should the offensive agent shoot from? Clearly shooting from its current position minimizes the chance of being blocked by the opponent agent but also has the lowest chance of shooting success since the character's location is far away from the basket. The closer spots raise the chance of shooting success but also increase the risk of being intercepted by the blocker. How can the shooting agent balance the tradeoffs?

One solution to this problem is to have the offensive agent model the time cost for the opponent to reach different potential shooting positions. By comparing this cost to its own cost to reach the shooting position, the offensive agent can balance the risk and reward of various locations. Combined with an estimate of the time required to make the shot, the shooting agent should choose a square such that:

$$C_o > C_d + T_s + \varepsilon \tag{3}$$

where C_o is cost required for an opponent to reach the square, C_d is the agent's dribbling cost, T_s is the agent's shooting time (constant for all locations), and ε is an error margin. Among the squares that fit this criterion, the offensive agent selects the square that maximizes its shooting success. Cost maps for the offensive and defensive agents are pre-computed; the decision can be performed at execution time in $O(n)$ time, where n is the number of squares under consideration.

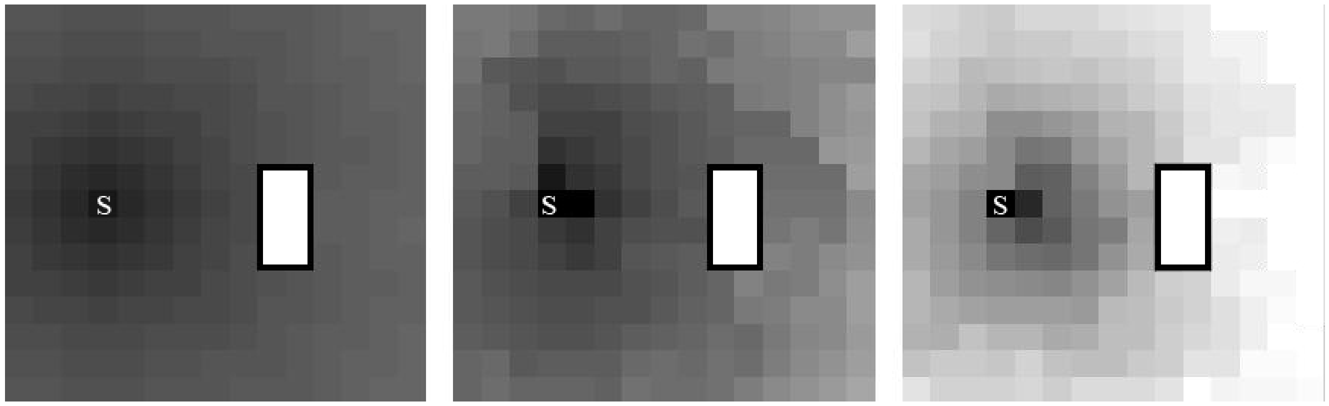

Figure 4. Comparative predictions of different cost models in a region with one obstacle. *S* **denotes the character's starting square; the obstructed region is white outlined in black. Costs increase as squares change from dark to light. The left panel shows the simple distance-based model of running generated with grassfire path-planning. The middle panel shows the cost model of the running MOUT soldier generated with human motion capture data. The right panel shows the cost map generated for the sneaking behavior using the motion capture data; the sneaking behavior produces a lighter cost map due to the greater time cost associated from sneaking and is not related to the running cost map in a strictly linear manner. Note the cost asymmetries in the human data, compared the grassfire transform which predicts that moving the same absolute distance in any direction will cost the same.**

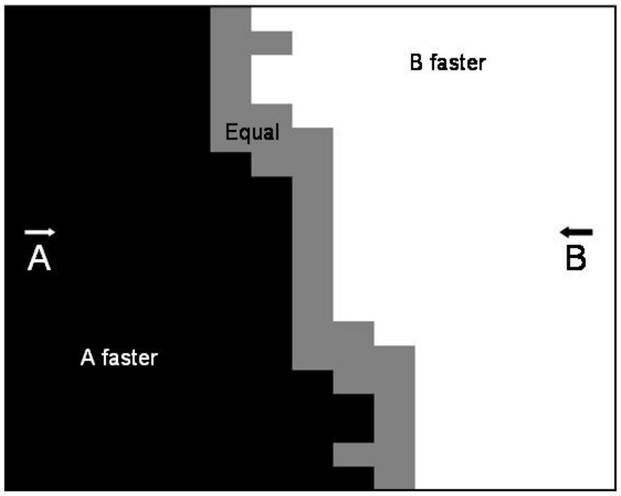

Figure 5. By examining the difference between its cost map and the opponent's, the agent can determine which squares it can reach before the opponent. Without prior knowledge, the agent simply models the opponent using the same cost map. This cost map was generated in 200,000 iterations considering paths of 50 frames or less (approximately 2 seconds).

3.3. Modeling Heterogeneous Agents

The previous scenario raises an important research question. Should the offensive agent apply its own cost model when reasoning about other agents, or should it maintain separate cost models for every agent? For the domains we are investigating, we build a separate cost model for every class of agent; for example, in the basketball domain, agents are divided into guards and forwards. Agent heterogeneity is modeled at the data collection phase with the following techniques: 1) collecting data from different individuals, preferably subject matter experts; 2) giving the same actor different instructions; 3) providing the actors with different equipment (encumbering clothing and gear will affect the actors' movement).

Variability between agents can also be expressed by directly modifying the motion graph. Eliminating or adding paths in the motion graph subtly modifies the final cost model. Behaviors are added or removed simply by enabling or disabling transitions; also ranges of behavior can be modified through changes in the graph. For instance, by disabling exemplars of tighter turns in the motion graph, the agent's turning radius can be changed, even though the agent hasn't been explicitly parameterized with variables such as running speed, turning radius, or maximum acceleration.

Cost models for heterogeneous agents are of key importance to team planners that have to make decisions about agent role-allocation; for instance a basketball team planner could incorporate the players' cost models into its role assignment process to determine which player has the best chance of dribbling and taking a particular shot.

3.4. Modeling Physical States

One potential flaw in our technique is that it doesn't do a good job of modeling endurance limits; our cost model generated by the animation graph suggests that agents can run, sneak, or dribble forever with no limit imposed by fatigue. Changes in physical state have to be explicitly introduced into the motion capture graph, but once introduced the cost model for implicit properties (speed, agility) can be derived by the graph. Using additional motion capture data, we model a wounded MOUT soldier with reduced capabilities. The new cost model is generated by: 1) adding new data for the wounded soldier walking and turning; and 2) removing certain behaviors (jumping) by disabling arcs in the graph. The resulting cost model varies in a way subtly different than merely rescaling the subject's speed and using the shortest distance model. The wounded soldier performs certain tasks at the same speed as the normal soldier, although its running performance is greatly reduced. An open question is whether it's easier to edit the motion graph or to tune explicit parameters. We believe that with an appropriate graphical interface, editing the motion capture graph to add new behaviors and disable inappropriate transitions (e.g., having the wounded soldier sprint or leap) will be reasonably intuitive.

4. Discussion

We compare our framework to related approaches that have emerged from different research communities:

- Many simulation agents use motion planning algorithms derived from the robotics community to produce rapid, optimal paths. Unfortunately, the resulting motion is not very human-like, and opponent agents created with this approach can be predictably outwitted in certain domains. Our method creates asymmetric cost models that are less easily anticipated by human trainees.
- The computer game industry has developed simple methods for parameterizing the behavior of AI bots using a small set of qualities such as speed and aggressiveness. Although this is a useful way to quickly create a large population of bots, the bots lack variability, since there are only a small number of attributes that can be tuned to adjust behaviors. Our approach of editing the motion capture graphs allows the developer to create many variations on the same bot by enabling and disabling edges in the graph.
- Biomechanical data has been collected for a limited set of endeavors but is not easily incorporated into decision-making algorithms without a model that predicts how changing low-level parameters affects high-level behaviors (dribbling, sneaking). Our model is easily incorporated into planning or learning algorithms since we directly compute the effects of using the high-level behavior on the world.

Our technique requires collecting a complete set of basis behaviors for each domain to be modeled, since data omissions can be interpreted as higher cost regions in the cost map. Also our method does not model phenomena such as fatigue or injury unless the modeler explicitly introduces them as separate behavior states. Like many exemplar-based, non-parametric models, the data collection costs could be quite high; we envision our framework only being used to model a small set of behaviors of high importance. We are still working on the problem of validating the cost maps against human behavior since experimental data is not available for many of the domains that we are interested in; our current plan is to compare the cost map time predictions with time predictions extracted directly from motion capture sequences not used in our graphs (k-fold cross-validation).

5. Related Work

The central motivation for our work is the problem of developing realistic agents to participate in human training, either as computer generated forces in military simulations or immersive virtual environments. Wray and Laird's [18] discussion of the need to introduce variability into human behavior modeling motivated certain aspects of our research, as well as Rickel and Johnson's work [16] on building virtual humans for team training. Much work on lifelike agents has been done by the conversational agents community, see [5] for an overview; pedagogical agents can also benefit from enhanced believability [14].

Jenkins and Matarić [10] have researched the problem of automatically deriving behaviors from a motion capture stream, whereas we assume that the behaviors are specified by the system designer using domain knowledge. Search techniques on motion graphs have been used to synthesize smooth human motion that closely follows a given path [12, 13] and to paint desired positional and behavioral constraints on a timeline [2]; in this paper we focus on the use of motion graphs to deduce the physical capabilities of a human from a spanning dataset of animation sequences. Reitsma and Pollard [15] first introduced the idea of unrolling a motion graph. Their work can assist in the capture process by suggesting data that might help fulfill the requirements of a particular environment or indicate that certain data is redundant and unnecessary. Funge, Tu, and Terzopoulos [8] couple cognitive modeling with animation techniques to reduce the burden on human animators. By cognitively empowering the characters, they make the animation task easier for the animator who need only specify a sketch plan for the animation sequence; in contrast, we focus on the task of improving decision-making and planning by incorporating costs extracted from motion capture sequences.

Other researchers have attempted to simulate or model motion based on human data. Although data isn't readily available for some of the physical endeavors that we've examined (e.g., sneaking), walking and running are relatively well understood behaviors. Hodgins [9] developed a rigid body model of a human runner with seventeen segments and thirty controlled degrees of freedom which she compared to video footage of human runners and biomechanical data. Fajen and Warren [7] have proposed a biological model of walking obstacle avoidance based on computations of goal angle and obstacle angle in which goals and obstacles are represented as attractors and repulsors for a second order dynamical system.

6. Conclusion and Future Work

In this paper, we demonstrate a general framework for building a non-parametric model of an agent's physical capabilities using human motion capture data. By precomputing cost maps for actions and areas of interest, the algorithm provides the high-level decision-making algorithm with a sound basis for selecting one behavior from a set of equivalently goal-achieving actions. In future work, we will explore other methods for incorporating the cost model predictions directly into a planner, beyond the current approach of locally selecting the best action at every step.

Acknowledgements

We thank Rahul Sukthankar for his insightful comments and the CMU Motion Capture Lab for capturing and processing our data. This work has been supported by ONR grant N00014-02-1-0438 and NSF grant EIA-0196217.

References

[1] O. Arikan and D. A. Forsyth. Interactive motion generation from examples. In *Proceedings of the 29th Annual Conference on Computer Graphics and Interactive Techniques*, pages 483–490, 2002.

[2] O. Arikan, D. A. Forsyth, and J. F. O'Brien. Motion synthesis from annotations. *ACM Trans. Graph.*, 22(3):402–408, 2003.

[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. Blum. A transformation for extracting new descriptors of shape. In *Proceedings of Symposium Models Perception Speech Visual Form*, 1964.

[5] J. Cassell, J. Sullivan, S. Provost, and E. Churchill, editors. *Embodied Conversational Agents*. MIT Press, 2000.

[6] K. Craig, J. Doyal, B. Brett, C. Lebiere, E. Biefeld, and E. Martin. Development of a hybrid model of tactical fighter pilot behavior using IMPRINT task network model and ACT-R. In *Proceedings of the Eleventh Conference on Computer Generated Forces and Behavior Representation*, 2002.

[7] B. Fajen and W. Warren. Behavioral dynamics of steering, obstacle avoidance, and route selection. *Journal of Experimental Psychology*, 29(2), 2003.

[8] J. Funge, X. Tu, and D. Terzopoulos. Cognitive modeling: Knowledge reasoning, and planning for intelligent characters. In *Proceedings of SIGGRAPH 99*, 1999.

[9] J. Hodgins. Three-dimensional human running. In *Proceedings of IEEE Intenational Conference on Robotics and Automation (ICRA)*, 1996.

[10] O. Jenkins and M. Matarić. Automated derivation of behavior vocabularies for autonomous humanoid motion. In *Proceedings of International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 2003.

[11] R. Jones, M. Tambe, J. Laird, and P. Rosenbloom. Intelligent automated agents for flight training simulators. In *Proceedings of the Third Conference on Computer Generated Forces and Behavior Representation*, 1993.

[12] L. Kovar, M. Gleicher, and F. Pighin. Motion graphs. In *Proceedings of the 29th Annual Conference on Computer Graphics and Interactive Techniques*, pages 473–482, 2002.

[13] J. Lee, J. Chai, P. S. A. Reitsma, J. K. Hodgins, and N. S. Pollard. Interactive control of avatars animated with human motion data. In *Proceedings of the 29th Annual Conference on Computer Graphics and Interactive Techniques*, pages 491–500, 2002.

[14] J. Lester and B. Stone. Increasing believability in animated pedagogical agents. In *Proceedings of the First International Conference on Autonomous Agents*, pages 16–21, 1997.

[15] P. Reitsma and N. Pollard. Evaluating and tuning motion graphs for character animation. Technical Report CS04-04, Brown University, 2004.

[16] J. Rickel and W. L. Johnson. Extending virtual human to support team training in virtual reality. In G. Lakemeyer and B. Nebel, editors, *Exploring Artificial Intelligence in the New Millenium*. Morgan Kaufmann Publishers, 2002.

[17] R. Tarjan. Depth first search and linear graph algorithms. *SIAM Journal of Computing 1*, pages 146–160, 1972.

[18] R. Wray and J. Laird. Variability in human behavior modeling for military simulations. In *Proceedings of Behavior Representation in Modeling and Simulation Conference (BRIMS)*, 2003.

[19] R. Wray, J. Laird, A. Nuxoll, and R. Jones. Intelligent oponents for virtual reality trainers. In *Proceedings of the Interservice/Industry Training, Simulation and Education Conference(I/ITSEC)*, 2002.