Creating physically embodied agents using realistic human motion models

Gita Sukthankar, Michael Mandel and Katia Sycara

Simulation Gaming 2008; 39; 64
DOI: 10.1177/1046878107309686

Creating physically embodied agents using realistic human motion models

Gita Sukthankar
Carnegie Mellon University, USA

Michael Mandel
Apple Computer, USA

Katia Sycara
Carnegie Mellon University, USA

Many simulations are populated with physically embodied agents capable of taking physical actions in the virtual world. Creating these agents, or virtual humans, is demanding; not only must the agents demonstrate visual verisimilitude, but they must plan and act in a way that is consistent with that of humans, especially for training simulations in which the participants are attempting to learn real-world skills. This article discusses an approach for adapting agent decision-making techniques to accurately model the physical capabilities of human subjects. To achieve this, the authors rely on human movement data acquired with a motion capture apparatus to build physically realistic models of human movement. To aid agents' planning, the authors construct a physical capability model for the agents, an accurate estimate of the time required for a real human to perform various movement sequences. A cost map over the space of agent actions is calculated by creating and stochastically sampling motion graphs assembled from the human data exemplars. The agents can use this cost model during the planning process to select between equivalent goal-achieving plans. This technique leverages highly accurate movement information acquired from human subjects to create agents that plan in physically realistic ways.

KEYWORDS: *agent decision making; decision making; human data exemplars; human modeling; human movement; human physical behaviors; motion capture; motion graphs; physical capability model; physically embodied agents; physically realistic models; planning; real-world skills; training and rehearsal simulations; virtual humans; visual verisimilitude*

Much research attention (Best & Lebiere, 2003; Craig et al., 2002; Jones, Tambe, Laird, & Rosenbloom, 1993; Wray, Laird, Nuxoll, & Jones, 2002) has been devoted to the problem of cognitive modeling to create agents with reasoning skills similar to those of humans. For applications such as training environments, computer games, and military simulations with computer-generated forces, it is not enough that we

AUTHORS' NOTE: We would like to thank Jessica Hodgins, Moshe Mahler, and the Carnegie Mellon University Motion Capture Lab for their help during the data acquisition process. Rahul Sukthankar's help proofreading this document was invaluable. This work was supported in part by Office of Naval Research Grant No. N000140210438, Air Force Office of Scientific Research Grant No. F496200110542, and International Technology Alliance Grant No. W911NF0630001.

SIMULATION & GAMING, Vol. 39 No. 1, March 2008 64-82
DOI: 10.1177/1046878107309686
© 2008 Sage Publications

simply create agents that are successful at completing tasks; we must seek to develop agents that can perform in a humanlike fashion. Ultimately, the user must step outside the simulation environment and into the real world in order to 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, human reasoning is constrained and influenced by an internal model of 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 article 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. As such, in this article we focus on the following issues:

- modeling humanoid physical capabilities,
- incorporating knowledge of the agent's physical aptitude into planning and opponent modeling, and
- constructing agent simulations that enforce a realistic physical capability model.

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

This article is organized as follows: First, we discuss some of the related work on simulating human motion. Next, we introduce our algorithm for constructing nonparametric physical capability models from human data. Then we present some results on the usage of our physical capability model for planning and opponent modeling. Finally, we detail the implementation of the model within the TEAMBOTS robotic simulation environment (Balch, 2000).

Related work

The central motivation for our work is the problem of developing realistic agents to participate in human training, as computer-generated forces in military simulations or immersive virtual environments. Wray and Laird's discussion (2003) of the need to introduce variability into human behavior modeling motivated aspects of our research, as did the work by Rickel and Johnson (2002) on building virtual humans for team training. Much work on lifelike agents has been done by the conversational agents community for applications such as interactive drama (Mateas & Stern, 2003; for an overview, see Cassell, Sullivan, Provost, & Churchill, 2000); pedagogical agents can also benefit from enhanced believability (Lester & Stone, 1997).

Motion graphs were introduced (Arikan & Forsyth, 2002; Kovar, Gleicher, & Pighin, 2002; J. Lee, Chai, Reitsma, Hodgins, & Pollard, 2002) to provide a solution for the need to adapt motion capture data to new, more complex environments while still producing visually smooth motion data. However, simply performing a runtime global search of the motion capture graph does not scale well to large numbers of characters or complicated virtual environments. In response to these problems and to simplify the search process, several groups have precomputed other types of data structures, such as mobility maps (Srinivasan, Metoyer, & Mortensen, 2005) and motion patches (K. Lee, Choi, & Lee, 2006), based on human motion data. Search techniques on motion graphs have been used to paint desired positional and behavioral constraints on a timeline (Arikan et al., 2003); in this article, we focus on the use of motion graphs to deduce the physical capabilities of a human from a spanning data set of animation sequences.

Jenkins and Mataric (2003) 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's using domain knowledge. Funge, Tu, and Terzopoulos (1999) 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 is not readily available for some of the physical endeavors that we have examined (e.g., sneaking), walking and running are relatively well-understood behaviors. Hodgins (1996) developed a rigid body model of a human runner with 17 segments and 30 controlled degrees of freedom, which she compared to video footage of human runners and biomechanical data. Fajen and Warren (2003) 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. Motion capture data has also been used in combination with optimization techniques for physics-based simulations of human movement (Safonova, Hodgins, & Pollard, 2004).

Physical capability model

In selecting actions for agents, the following important question arises: How much time does it 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 backward without changing direction? Either sequence of actions will ultimately get the agent to the same location (x, y), so there is no reason for a naïve planner who is lacking a model of human behavior to prefer one plan over the other. A priori, the planner might assume that the plan that requires 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 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; and
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.

Step 1: Data collection

Human motion data is captured using a Vicon optical motion capture system with 12 cameras, each capable of recording at 120 Hz, with images of 1,000 × 1,000 resolution. We use a marker set with 43 14-mm 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 feet × 24 feet. Figure 1 shows a subject in the motion capture laboratory . The motion is generally captured in long clips (over a minute) to allow the subjects to perform natural transitions between behaviors, and it 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

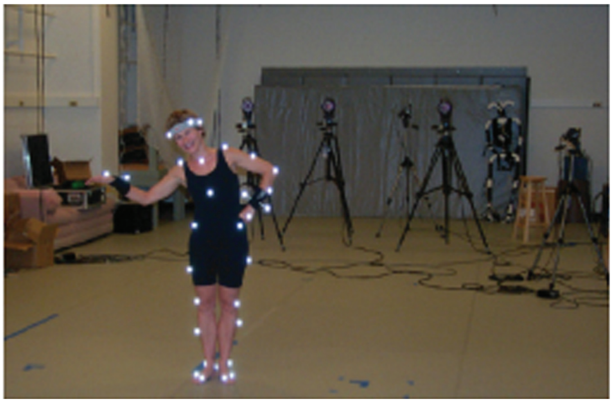


FIGURE 1: A Subject Wearing a Retro-Reflective Marker Set in the Carnegie Mellon University Motion Capture Laboratory

spanning set of motion for a particular behavior, our manual labeling is made tractable with long sessions where each take is aimed at capturing the full range of a single behavior. 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, semiautomated techniques have been proposed (Arikan, Forsyth, & O'Brien, 2003) to assist in annotation that only require a relatively small portion of the database to be manually labeled.

During a motion capture session, the subject is instructed to perform a set of physical behaviors, such as moving around the room, performing a physical skill, and manipulating objects. It is important to have the subject perform any variations on a behavior (e.g., picking up an object with the right hand and the left hand) that need to be incorporated in the physical capability model. Directly capturing transitions between behaviors produces higher-quality motion data than does interpolating motion segments between behaviors. Assessing the final quality of a motion data set is a difficult problem; Reitsma and Pollard (2004) have described an algorithm for embedding motion capture data into a target environment and calculating path quality and environment coverage. In practice, it is possible for an experienced technician to evaluate the data set by visually observing a synthetic character animated using the data and noting discontinuities in the character's motion.

Step 2a: 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 allow playback of the subject's performance from the

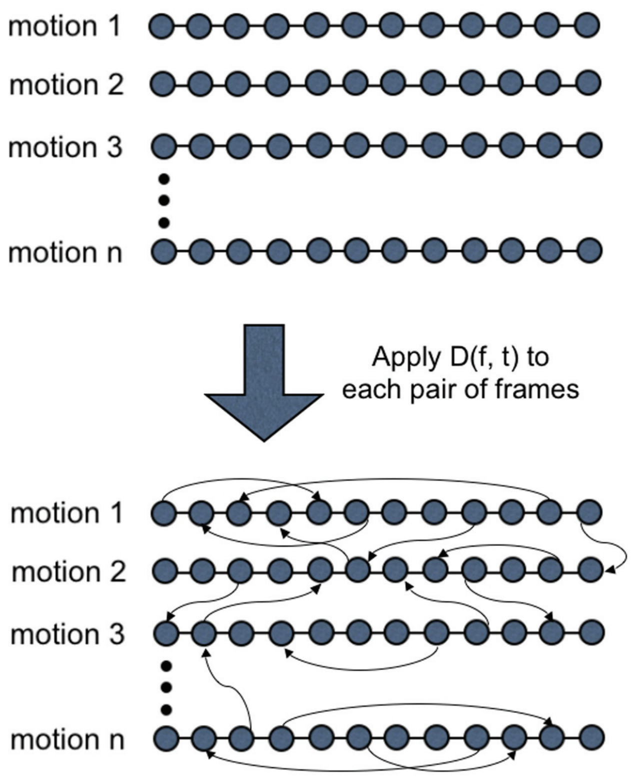


FIGURE 2: The Motion Graph—Constructed by Finding the Distance Between Each Frame in the Database

NOTE: Edges are added to the graph when $D(f, t)$ 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.

capture session in an environment that is fixed in size and layout. It would be desirable to adapt the captured behaviors to new, more complex environments in which an animator or computer simulation could intuitively direct the character's actions.

Motion graphs were introduced (Arikan & Forsyth, 2002; Kovar et al., 2002; J. Lee et al., 2002) 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.

Next, we discuss how to construct a motion graph and how searching the graph structure enables us to compute a cost for navigating between two points in a simulated environment.

Step 2b: 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. Because the data are 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. For many behaviors, the 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. An alignment transformation can be computed to create a canonical frame of reference for comparison of pose sequences.

Our distance metric is modeled most closely after the one introduced by Kovar et al. (2002). 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 that each pose has matching translation in the ground plane and vertical axis rotation. The metric is computed as follows:

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

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 neither 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 an adapted pruning strategy (Kovar et al., 2002; J. Lee et al., 2002) 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 of the graph, which can be performed efficiently (Tarjan, 1972).

Step 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 that 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.

Step 4: 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 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: first, time cost, which is directly proportional to the path length in frames; second, goal achievement, or how well the path achieves the desired goal state. The cost metric is

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

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, given that all paths that fall within a grid cell are close to the goal, but it increases the time required to generate the cost map. The experiments described in this article were run with $\alpha = 0$. 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 humanlike within a certain threshold, we omit smoothness from our path-scoring criterion.

Step 5: 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 it should be sampled at sufficient resolution to ensure that discretization errors do not eliminate solutions. For example, a map covering a 50 feet × 50 feet area at a resolution of 10 × 10 corresponds to a grid with 100 equally spaced cells, each 5 feet × 5 feet in size.

The algorithm stochastically samples the set of valid paths to move through the environment using the selected behavior, and it 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 above) along the candidate paths. Random paths are repeatedly generated until the cost map converges to a stable configuration.

We adopted this strategy because of the high branching factor of the motion graph and the necessity to simultaneously evaluate every potential cell in the cost map's space. Given that 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 because 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; Figure 4 shows the number of iterations required for cost value convergence.

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.

To perform the Monte Carlo sampling, we selected a reasonable maximum search depth based on the average velocity of the subject. The cost maps used in this article were constructed with search depths of 500-2,000 frames. The algorithm executes a random walk of the motion graph, starting from the character's current pose and orientation. At every choice point in the motion graph, the algorithm stochastically selects a choice point to follow after pruning physically impossible paths. Each grid cell in the cost map is initialized as having cost greater than the maximum search depth; if a cheaper path to that location is discovered, then the cost of that cell is set to the cost of the path, and that path is saved for future use.

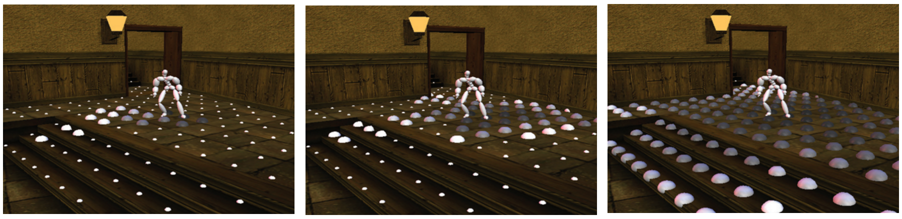


FIGURE 3: The Monte Carlo Sampling of Paths Through the Motion Graph Iteratively Approximates the Cost Map

NOTE: 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, whereas the subsequent panels show how additional iterations contribute to convergence. As yet unexplored regions of the world are indicated by smaller spheres.

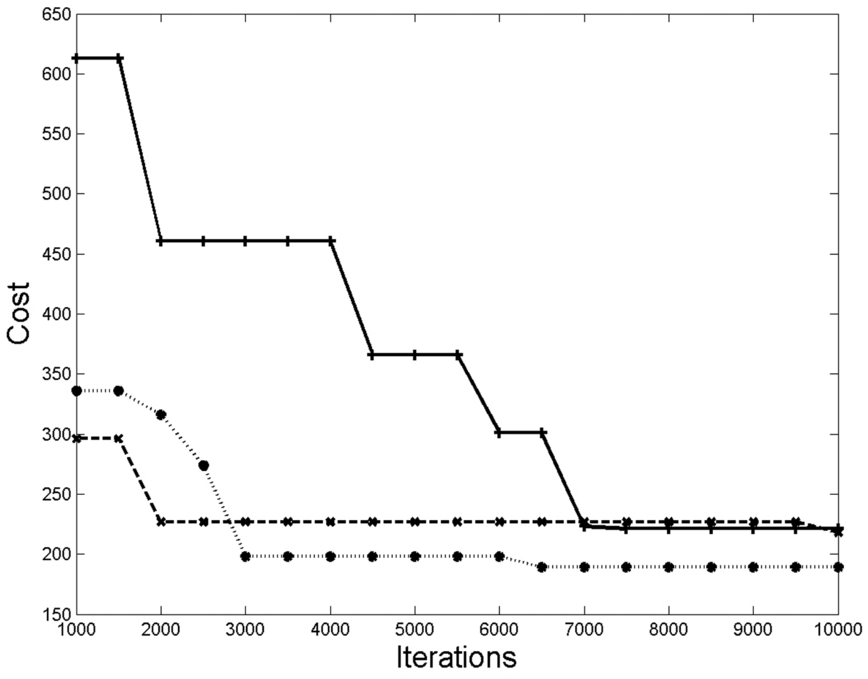


FIGURE 4: Convergence of Physical Capability Model

NOTE: This graph shows the cost value from three different cells in the cost map generated using the sneak motion graph. We see that the values have converged after 7,000 iterations.

Results

Here we present results from the usage of our physical capability models for planning and opponent modeling.

Comparative cost models for planning

One of the domains that 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 signaling, inspecting areas, and probing for booby traps. Using the human data and the algorithms described above, we built a motion capture graph that generates smooth animations of the MOUT soldier performing various behaviors.

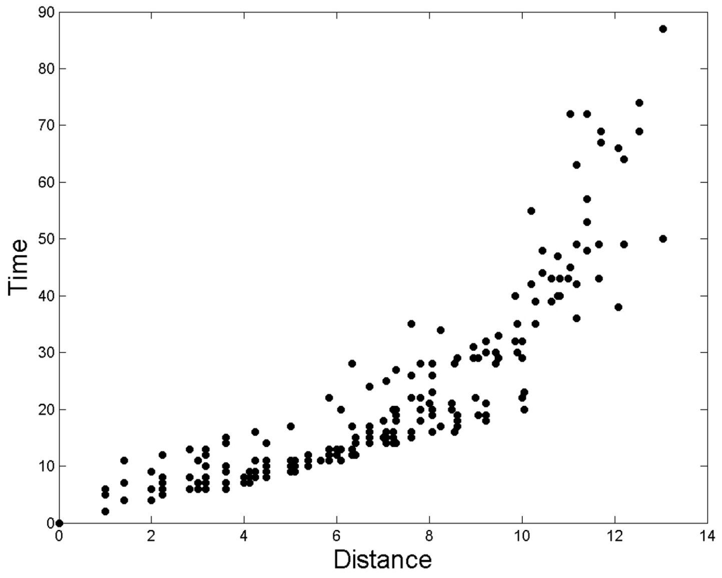
In the first scenario, we model the cost of the MOUT soldier running around an obstacle, to determine the trade-offs between having the soldier run around a hazardous area versus executing other plans of action (sneaking or probing for booby traps). We restrict the animation system to using behaviors labeled with the *running* descriptor, and we 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 grassfire robotic path planners. Figure 5 shows the distribution of costs (time) as a function of distance traversed by the agent for the physical capability model. Unlike grassfire, which would always predict that cost is proportional to traversed distance, we see that our model predicts a range of possible times, dependent on the relative position of the goal with respect to the starting pose of the agent.

Figure 6 shows a cost map generated using grassfire path planning (Blum, 1967) 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.

Opponent modeling using physical capability models

Not only can the agent improve its plans with a refined cost model, but it can also anticipate its opponents' actions using the cost model. We now discuss how our model can be applied in the basketball domain for a single agent who is 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 nondribbling 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. From which location should the offensive agent shoot? Clearly, shooting from its current position



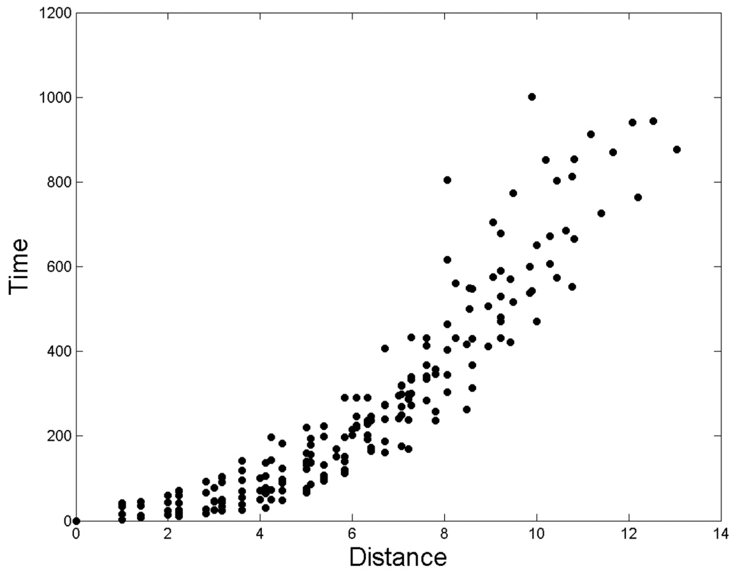


FIGURE 5: Predictions of the Physical Capability Model for Running (*top*) and Sneaking (*bottom*)
NOTE: The graphs are displayed as distance (feet) versus time (frames, 1/30 sec). Unlike grassfire, which would predict that the time required to cover a particular distance is constant, we observe that the time predicted by our physical capability model is path-dependent and highly variable.

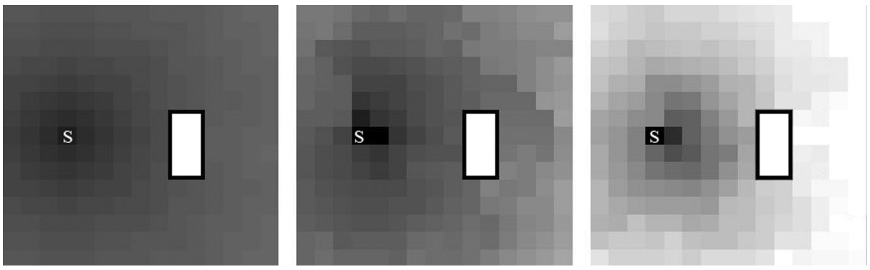


FIGURE 6: Comparative Predictions of Different Cost Models in a Region With One Obstacle
NOTE: *S* denotes the character's starting square; the obstructed region is white and outlined in black. Costs increase as squares change from dark to light. The left panel shows the simple distance-based model of running generated with grassfire path planning. The middle panel shows the cost model of the running MOUT (Military Operations in Urban Terrain) 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 because of the greater time cost associated from sneaking and is thus not related to the running cost map in a strictly linear manner. The human data exhibit asymmetric costs, unlike the grassfire transform, which predicts that moving the same absolute distance in any direction should cost the same.

minimizes its chance of being blocked by the opponent agent, but doing so also has the lowest chance of success because the character's location is far from the basket. The closer spots raise the chance of shooting success but increase the risk of being intercepted by the blocker. How can the shooting agent balance the trade-offs?

One solution 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 precomputed; the decision can be performed at execution time in $O(n)$ time, where n is the number of squares under consideration.

Simulating agents with physical capability models

To demonstrate how physical capability modeling can be incorporated within a simulation environment, we modified the Java-based TEAMBOTS to allow agents to use cost maps in their movement planning process, and we precomputed motion data paths

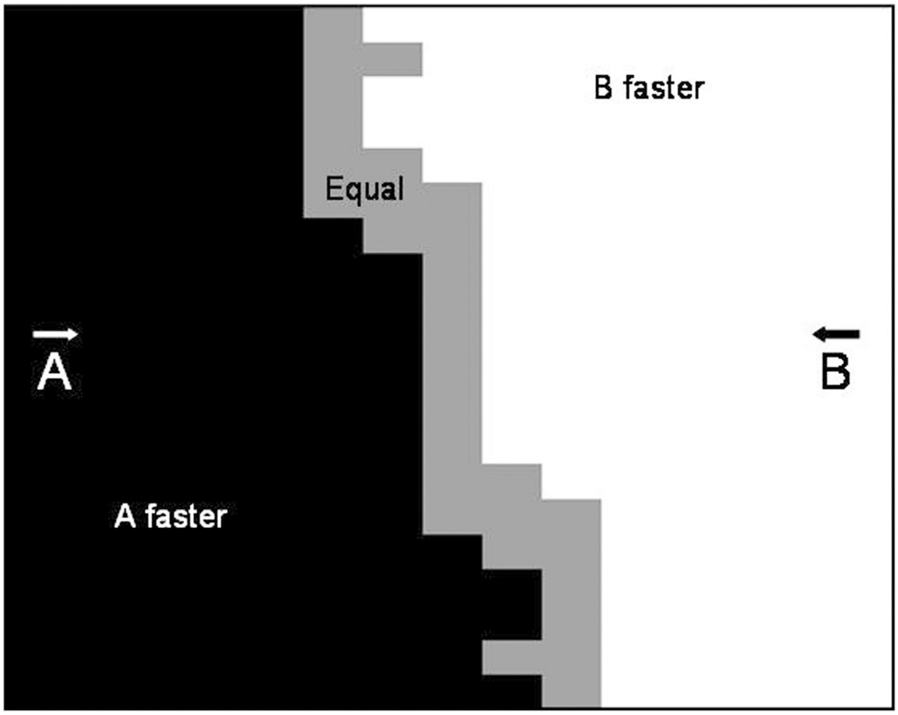


FIGURE 7: Opponent Modeling Using Cost Maps
NOTE: By examining the difference between its cost map and the opponent's, the agent can determine which squares it can reach before the opponent can. Without prior knowledge, the agent simply models the opponent using the same cost map. This cost map was generated in 200,000 iterations considering paths of 50 frames or less (approximately 2 sec).

to generate their locomotion. TEAMBOTS was originally developed as a simulator and real-time robotic execution environment for simulating teams of soccer-playing robots for the Robocup competition. In our experiments, we modified TEAMBOTS to simulate a basketball game and so implemented a two-player agent team.

Movement generation

When an agent initiates a move to a new location, the modified TEAMBOTS simulator uses precomputed paths from the human motion capture data to move the agent between points, instead of using the simple kinematic model included in the default TEAMBOTS implementation. Agents plan their moves by checking the precomputed cost maps to find the least-cost goal-achieving location. For each location in the cost map, there is a corresponding clip of motion data that represents the shortest plausible motion that a human would use to reach the given location; see Figure 7.

Some care is necessary to embed the motion data into the TEAMBOTS simulator for 2D visualization. The original data provide a 3D global position, global orientation, and then relative orientations for the hierarchy of joints that represent human's skeleton. The position and orientations (represented as unit quaternions) are also relative to the coordinate frame of the original motion capture studio. Given that the TEAMBOTS simulator is 2D and allocates each agent its own coordinate system, the data must be interpreted in the right way to get proper results. The data are exported from the 3D simulator with all positions relative to the origin, but rotations are left in their original coordinate frame. To generate 2D motion paths, the data is projected into the *xz* plane. Joint angles for the skeleton are ignored because they are not displayed in the TEAMBOTS 2D visualization. To render and execute the paths at runtime, the positions must be scaled to fit the game units in TEAMBOTS, and a coordinate transformation must be applied to align the orientation with the steering angle of the agents. Each time that an agent selects a move action, the coordinate transformation is computed to rotate the original motion data into the coordinate system of the moving agent. For playback, the Gamebots global timer is used to enforce a playback speed of 30 frames per second to match the original data. Figure 8 shows an example of a spin maneuver being simulated within our modified TEAMBOTS implementation.

Agent strategies

Using our modified TEAMBOTS simulator, we developed a simple basketball team consisting of two offensive players and one defensive player to illustrate the application of our physical capability model to planning and opponent modeling. Focus was given to providing strategies for the offensive players; the defensive agent simply tries to move toward the ball. The offensive agents can select from the following four actions:

Maneuver toward goal: The simpler model moves straight toward the goal with a constant velocity, whereas the cost map version uses cost map to calculate the cheapest heading, as well as motion data to move.

Ready for pass: The model moves toward goal but actively tries to avoid the teammate to remain open for a pass.

Shoot ball: The model shoots the ball at the basket.

Pass to teammate: The simpler model passes when the defender is within a fixed radius, whereas the cost map version estimates the opponent's arrival time using the technique described above.

Figure 9 displays the conditional plan followed by the offensive players.

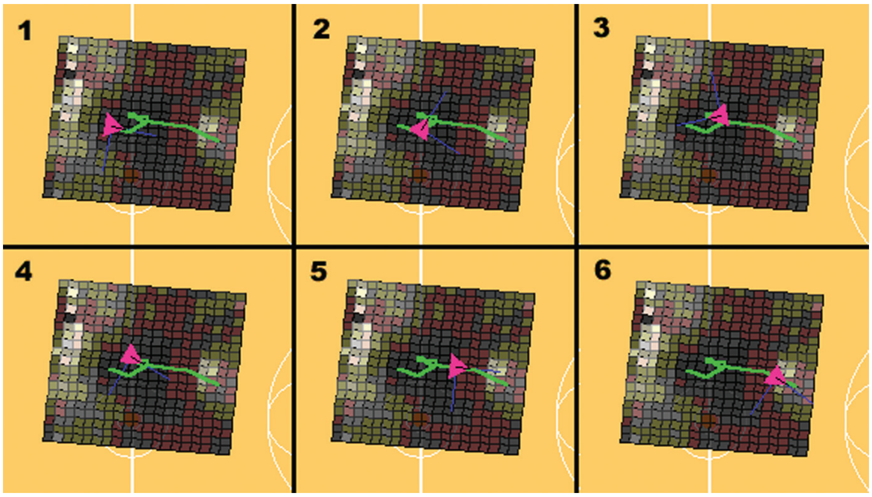


FIGURE 8: A Spin Maneuver Executed Within the TEAMBOTS Simulator
NOTE: The triangle marks the agent's current position; the line marks the path that the agent is currently following. The cost map used to calculate the path is shown in the rectangles. A coordinate transformation is applied to the motion data to align it with the agent's initial steering angle.

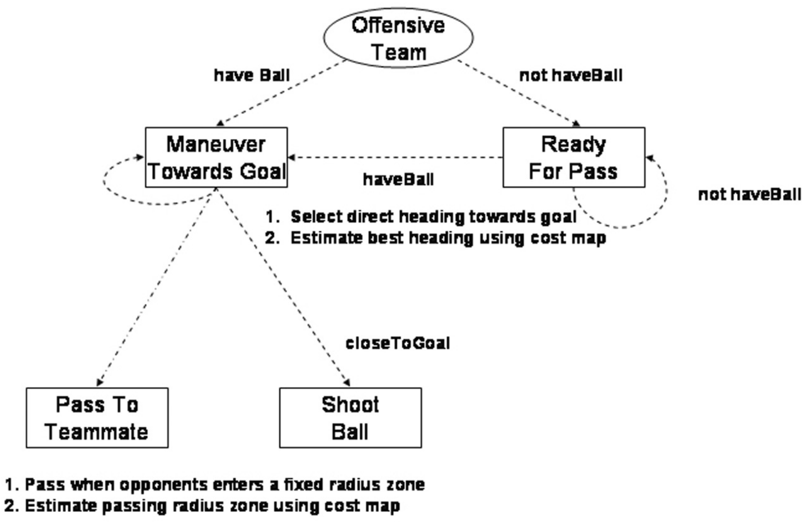

Offensive Team
have Ball
not haveBall
Maneuver Towards Goal
Ready For Pass
haveBall
not haveBall
1. Select direct heading towards goal
2. Estimate best heading using cost map
closeToGoal
Pass To Teammate
Shoot Ball
1. Pass when opponents enters a fixed radius zone
2. Estimate passing radius zone using cost map

FIGURE 9: Conditional Plan Used by Offensive Agents
NOTE: *Maneuver toward goal* → *Pass to teammate* yields two options, depending on whether the agent is utilizing its physical capability model.

Discussion

We now 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 humanlike, 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 artificial intelligence 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, they lack variability because there are only a few attributes that can be tuned to adjust behaviors.
- Biomechanical data have been collected for a limited set of endeavors but are not easily incorporated into decision-making algorithms without a model that predicts how changing low-level parameters affect high-level behaviors (dribbling, sneaking). Our model is easily incorporated into planning and learning algorithms because 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, given that data omissions can be interpreted as higher-cost regions in the cost map. Also our method does not model phenomena such as fatigue and injury unless the modeler explicitly introduces them as separate behavior states. Like many exemplar-based nonparametric models, the data collection costs can be quite high; we envision our framework as being used only to model a small set of behaviors of high importance. The cost function used in this research is primarily based on minimizing behavior times; other types of cost functions (e.g., smoothness) have been used to model human motor functions (Todorov, 2004). We believe that our algorithms can be adapted to different cost functions; because our primary concern was to model time-stressed human behaviors, the use of a minimum time criterion was appropriate for our simulation.

Conclusion

In this article, we demonstrate a general framework for building a nonparametric 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. Although we do not quantify the degree to which our plans match human motion profiles, our simulated agents move in a pattern that qualitatively matches human expectations.

References

Arikan, O., & Forsyth, D. A. (2002). Interactive motion generation from examples. In *Proceedings of the 29th annual conference on computer graphics and interactive techniques* (pp. 483-490). New York: ACM Press.

Arikan, O., Forsyth, D. A., & O'Brien, J. F. (2003). Motion synthesis from annotations. *ACM Transactions on Graphics, 22*(3), 402-408.

Balch, T. (2000). TEAMBOTS [Code for simulator]. Available at http://www.teambots.org/. (IIC, 85 Fifth Street Atlanta, Georgia, 30308)

Best, B., & Lebiere, C. (2003, May). *Spatial plans, communication, and teamwork in synthetic MOUT agents*. Paper presented at the Behavior Representation in Modeling and Simulation Conference, Mesa, AZ.

Blum, H. (1967). A transformation for extracting new descriptors of shape. In W. Wathen-Dunn (Ed.), *Symposium on models for the perception of speech and visual form* (pp. 362-380). Cambridge, MA: MIT Press.

Cassell, J., Sullivan, J., Provost, S., & Churchill, E. (Eds.). (2000). *Embodied conversational agents.* Cambridge, MA: MIT Press.

Craig, K., Doyal, J., Brett, B., Lebiere, C., Biefeld, E., & Martin, E. (2002, May). *Development of a hybrid model of tactical fighter pilot behavior using IMPRINT task network model and ACTR*. Paper presented at the Eleventh Conference on Computer Generated Forces and Behavior Representation, Orlando, FL.

Fajen, B., & Warren, W. (2003). Behavioral dynamics of steering, obstacle avoidance, and route selection. *Journal of Experimental Psychology, 29*(2), 343-362.

Funge, J., Tu, X., & Terzopoulos, D. (1999, August). *Cognitive modeling: Knowledge reasoning, and planning for intelligent characters*. Paper presented at SIGGRAPH 99, Los Angeles.

Hodgins, J. (1996, April). *Threedimensional human running*. Paper presented at the IEEE Intenational Conference on Robotics and Automation, Minneapolis, MN.

Jenkins, O., & Mataric, M. (2003, July). *Automated derivation of behavior vocabularies for autonomous humanoid motion.* Paper presented at the International Conference on Autonomous Agents and Multiagent Systems, Melbourne, Australia.

Jones, R., Tambe, M., Laird, J., & Rosenbloom, P. (1993, May). *Intelligent automated agents for fight training simulators*. Paper presented at the Third Conference on Computer Generated Forces and Behavior Representation, Orlando, FL.

Kovar, L., Gleicher, M., & Pighin, F. (2002). Motion graphs. In *Proceedings of the 29th annual conference on computer graphics and interactive techniques* (pp. 473-482). New York: ACM Press.

Lee, J., Chai, J., Reitsma, P. S. A., Hodgins, J. K., & Pollard, N. S. (2002). Interactive control of avatars animated with human motion data. In *Proceedings of the 29th annual conference on computer graphics and interactive techniques* (pp. 491-500). New York: ACM Press.

Lee, K., Choi, M., & Lee, J. (2006, August). *Motion patches: Building blocks for virtual environments annotated with motion data*. Paper presented at SIGGRAPH 2006, Boston.

Lester, J., & Stone, B. (1997). Increasing believability in animated pedagogical agents. In *Proceedings of the First International Conference on Autonomous Agents* (pp. 16-21). New York: ACM Press.

Mateas, M., & Stern, A. (2003, March). *Facade: An experiment in building a fully-realized interactive drama*. Paper presented at the Game Developers Conference, San Jose, CA.

Reitsma, P., & Pollard, N. (2004, August). *Evaluating motion graphs for character navigation*. Paper presented at ACM SIGGRAPH/Eurographics Symposium on Computer Animation, Grenoble, France.

Rickel, J., & Johnson, W. L. (2002). Extending virtual human to support team training in virtual reality. In G. Lakemeyer & B. Nebel (Eds.), *Exploring artificial intelligence in the new millenium* (pp. 217-238). Burlington, MA: Kaufmann.

Safonova, A., Hodgins, J., & Pollard, N. (2004, August). *Synthesizing physically realistic human motion in lowdimensional, behaviorspecific spaces*. Paper presented at SIGGRAPH 2004, Los Angeles.

Srinivasan, M., Metoyer, R., & Mortensen, E. (2005, May). *Controllable realtime locomotion using mobility maps*. Paper presented at Graphics Interface, Victoria, British Columbia, Canada.

Tarjan, R. (1972). Depth first search and linear graph algorithms. *SIAM Journal of Computing, 1,* 146-160.

Todorov, E. (2004). Optimality principles in sensorimotor control. *Nature Neuroscience, 7*(9), 907-915.

Wray, R., & Laird, J. (2003, May). *Variability in human behavior modeling for military simulations.* Paper presented at the Behavior Representation in Modeling and Simulation Conference, Mesa, AZ.

Wray, R., Laird, J., Nuxoll, A., & Jones, R. (2002, December). *Intelligent oponents for virtual reality trainers.* Paper presented at the Interservice/Industry Training, Simulation, and Education Conference, Orlando, FL.

Gita Sukthankar is a doctoral student in robotics at Carnegie Mellon University. Previously, she received a bachelor of arts (cum laude) in psychology from Princeton University and a master of science in robotics from Carnegie Mellon. From 2000 to 2003, she worked as a researcher at HP Labs (CRL) in the handheld computing group before returning to Carnegie Mellon for her doctorate. She is the 2005 winner of the Best Student Paper Award at the International Conference on Autonomous Agents and Multi-Agent Systems and a 2006 Microsoft Graduate Research Fellowship finalist.

Michael Mandel graduated from Carnegie Mellon University with a bachelor of science and a master of science in computer science. He works for Apple Computer, developing 3D technology for future products. His research interests include data-driven motion synthesis, physical simulation of human motion, interactive interfaces for character animation, and learning human capabilities from motion capture data.

Katia Sycara is a research professor in the School of Computer Science at Carnegie Mellon University and director of the Advanced Technology Lab. She has been the principal investigator for many multimillion-dollar grants from Defense Advanced Research Projects Agency, Air Force Office of Scientific Research, Advanced Research and Development Activity, Office of Naval Research, NASA, National Science Foundation, and industry. Dr. Sycara was General Chair of Autonomous Agents 1998, founding editor-in-chief of the Journal of Autonomous Agents and Multiagent Systems, *and is on the editorial board of four other journals. She has authored over 200 technical articles and book chapters on multiagent systems. Dr. Sycara is an AAAI Fellow and recipient of the ACM Agents Research Award.*

ADDRESSES: Gita Sukthankar: Robotics Institute, Carnegie Mellon University, 5000 Forbes Avenue, Pittsburgh, PA 15213, USA; telephone: +1-412-268-5245; fax: +1-412-268-5569; e-mail: gitars@cs.cmu.edu. Katia Sycara: Robotics Institute, Carnegie Mellon University, 5000 Forbes Avenue, Pittsburgh, PA 15213, USA; Telephone: +1-412-268-8825; fax: +1-412-268-5569; e-mail: katia@cs.cmu.edu. Michael Mandel: Apple Computer, 4720 Forbes Avenue, Suite 420, Pittsburgh, PA 15213, USA; telephone: +1-412-420-1357; fax: +1-412-420-1357; e-mail: mike@mmandel.com.