PLANNING AND PARALLEL TRANSITION NETWORKS: ANIMATION'S NEW FRONTIERS

NORMAN I. BADLER
Center for Human Modeling and Simulation
Computer and Information Science Department
University of Pennsylvania
Philadelphia, PA 19104-6389 U.S.A.
E-mail: badler@central.cis.upenn.edu

and

BONNIE L. WEBBER*
Center for Human Modeling and Simulation
Computer and Information Science Department
University of Pennsylvania
Philadelphia, PA 19104-6389 U.S.A.
E-mail: bonnie@central.cis.upenn.edu

ABSTRACT

Animating realistic human agents involves more than just creating movements that look "real." A principal characteristic of humans is their ability to plan and make decisions based on intentions and the local environmental context. "Animated agents" must therefore react to and deliberate about their environment and other agents. Our agent animation uses various low level behaviors, sense-control-action loops, high level planning, and parallel task networks. Several systems we developed will illustrate how these components contribute to the realism and efficacy of human agent animation.

1. Introduction

Conventional animations often seek to re-create "life" through the artistic skills of an animator who transforms his or her observations, experience, and intuition into believable characters.[23] Even now, most of the tools designed to aid this craft provide manual control over images, shapes, and movements. Recently, more automated techniques for animation have been developed, often to ease some burden or other on the animator. For example, dynamics can be used to animate particles or objects,[12,26] or "flocking" considerations can be used to constrain interactions between a number of figures.[18,24] Partial success can be judged from the various physics-based techniques that use "real-world" mathematics to get the motions "right."[22,27,10]

Unfortunately, getting animated people (or human-like characters) seems to require more than the existing physical or manual toolset. One reaction to this difficulty

*Additional co-authors are Welton Becket, Chris Geib, Mike Moore, Catherine Pelachaud, Barry Reich, and Matthew Stone.

is the move toward "performance animation" where live actors go through the necessary motions while various sensing systems monitor various body landmarks labeled by markers or electromagnetic sensors.[1,20] While this provides motion data of unquestioned realism, it is only a specific instance of a performance and might still need massaging by an expert. For example, such motion cannot be directly used on a creature of markedly different body size than the original actor.

Something is still missing when one moves toward this performance-based animation. In fact, one can view performance animation as simply a way of guaranteeing that the physics is correct — without building and evaluating the formulas. Consider the following scenario:

> A pedestrian stands on a street corner, waiting for the light to change so that he can safely cross the street. Meanwhile a car is approaching the same intersection. What happens when the light changes?

First of all, the performance-based data might be useful for animating the character's walk, though it could also be simulated through a locomotion generator.[13] In a scripted animation, the animator would be responsible for initiating the walk at the time of the light change, and would also be controlling the car motions. So now suppose that we remove the animator: what would the pedestrian do? Well, if he were completely driven by physics, he would start to cross the street (he would have a forward force that propels him that way). The car would also be moved by physics. If they happen to arrive at the same place at the same time, the animation might be exciting but it would not be fun for either the car or the pedestrian. So what happened when we removed the animator? We removed the *decisions* that were made to have the pedestrian only cross the street when it was safe to do so, even if the car ran the red light. So there is an important clue to realistic animation here: *human movement realism includes decision-making in context.*

It is perhaps not so surprising that real humans engage in decision-making as one of their uniquely human qualities. What we note here is that *synthetic humans must also engage in decision-making if we want them to share human qualities.* Sensed human motions are not enough for realism because no choices can be made with them. Physics alone is not good enough for realism because there are no decisions outside the outcome of the mathematical laws. Humans operate differently from both of these strict models: we are neither puppets nor mannequins. We make choices, watch our surroundings, and plan for the future. We sense the world in order to accomplish intentions, validate expectations, avoid obstacles, and minimize surprises.[25]

It is the existence of choice that motivates much of our research. We see animation as an integration of a rich collection of interacting techniques, organized in a principled, structured representation. These techniques include planners and Parallel Transition Networks (PaT-Nets) (Sec. 2.2) to aid in overall task control, and goal-based sensing, response, and (as necessary) physics-based, kinematic or inverse kinematic behaviors to achieve environmentally-appropriate movements.[3]

In the remainder of this paper we will briefly review the two-level architecture to intelligent agents that we are investigating.[a] We will then discuss three sample domains that utilize this architecture to great advantage: "Stepper" (Sec. 3), "Hide and Seek" (Sec. 4), and "Gesture Jack" (Sec. 5). We close with a view of the future.

2. The Agent Architecture

An *agent* is an object that is able to take action by virtue of having a sense-control-action (SCA) loop to produce locally-adaptive behavior (Sec. 2.1). In general, behaviors are considered "low level" capabilities of an agent, such as locomoting [to], reaching [for], looking [at], speaking [to], listening [to], *etc.* An agent with only SCA loops determining its behavior is considered a purely "reactive" agent. Deliberative agents (such as human agents) also have higher-level control structures that affect more global, planned, or cognitive aspects of behavior, which in turn can affect the immediate formulation and parameters of an SCA loop. We use planners and PaT-Nets for this level of control (Sec. 2.2). Because our interest is in behavioral realism, we focus on human agents, whose behaviors we have the greatest familiarity with and expectations about, thus providing a grounding for judging realistic behavior.

An intelligent agent must interleave sensing, planning, decision-making, and acting. Accordingly, it is desirable to create an architecture that permits specification and exploration of each of these processes. Planning and decision-making can be accommodated through incremental, symbolic-level reasoning. When the agent decides to act, the symbolic actions must be instantiated in executable behaviors. Most behavioral systems use either state controllers or numerical feedback streams, but not both. By using both it is possible to obtain maximum flexibility and maintain appropriate levels of specification.[3,4]

We can characterize these two control levels as PaT-Nets and SCA loops.

- PaT-Nets are parallel state-machines that are easy for humans and automatic planning systems to manipulate. They are also good at sequencing actions based on the current state of the environment or of the system itself. They characterize the tasks in progress, conditions to be monitored, resources used, and any temporal synchronization.

- The SCA loop performs low-level, highly reactive control involving sensor feedback and motor control.

In this paradigm, the agent can instantiate explicit PaT-Nets to accomplish certain goals (*e.g.*, go to the supply depot and pick up a new motor), while low-level control can be mediated through direct sensing and action couplings in the SCA loop (*e.g.*, controlling where the agent's feet step and making sure that s/he doesn't run into or

[a] Much of the material that follows is condensed from a forthcoming book chapter.[2]

trip over any obstacles). Since the sensors can establish what the agent can perceive, the agent is able to react through the SCA loop, and if desired, use this information to confirm, adopt, or select higher-level (cognitive) actions: for example, if an obstacle cannot be surmounted, the current PaT-Net might need to be reconsidered. Since PaT-Net state transitions are explicitly represented, alternative behaviors may be easily embedded.

The rest of this discussion describes features of SCA loops and PaT-Nets. We will then have enough tools in place to illustrate the interactions between planning, PaT-Nets, and SCA behaviors in three domains: "Stepper," "Hide and Seek," and "Gesture Jack."

2.1. Low-Level Control: Sense-Control-Action Loops

The SCA or *behavioral loop* is a continuous stream of floating point numbers from the simulated environment, through simulated sensors providing the abstract results of perception, through control decisions independently attempting to solve a minimization problem, out to simulated effectors or motor actions (walking, *e.g.*), which enact changes on the agent or the world. This loop continuously operates, connecting sensors to effectors through a network of nodes which for descriptive convenience are divided into *sense* (S), *control* (C), and *action* (A) phases.

The behavioral loop is modeled as a network of interacting SCA processes connected by arcs across which only floating point messages travel. An individual, conceptual path from sensors to effectors is referred to as a *behavioral net*. It is analogous to a complete behavior in an "emergent behavior" architecture such as Brooks' *subsumption architecture*,[8] except that nodes may be shared between behaviors, and arbitration (competition for effector resources) may occur throughout the behavioral path and not just at the end-effector level. The behavioral loop is modeled as a network with floating point connections in order to allow the application of low-level, unsupervised, reinforcement learning in the behavioral design process. (This is being developed by Becket.[5]) Since our main use of SCA loops to date has been in locomotion reasoning, the remaining discussion will be in these terms.

2.1.1. Sensory Nodes

Sensory nodes model or approximate the abstract, geometric results of object perception. They continuously generate signals describing the polar coordinate position (relative to the agent) of a particular object or of all objects of a certain type within a specified distance and field of view. A few of the sensors used are:

object sensors: These provide the current distance from the agent and angle relative to the forward axis of the agent of a particular object in the environment. (Currently our sensors abstract over object recognition. A more sophisticated approach would simulate an agent's vision using Z-buffering hardware to create a depth map of what the agent can see.[17,19])

range sensors: A range sensor collects all objects of a certain type within a given range and field of view, and performs a weighted average into signals giving the distance and angle of a single abstract object representing all detected objects. Signals into the sensor define the range, field of view, and weighting parameters (defining relative weights of distance and angle) and may be altered continuously in order to focus the sensor.

terrain mapper: This sensor perceives an internal map of the terrain as if it were an external entity.

human sensor: If an object is a human, that information is detected by this sensor.

Other sensors can be developed and embedded in the architecture as needs arise.

2.1.2. Control Nodes

For locomotion reasoning we use two simple control nodes loosely based on Braitenberg's *love* and *hate* behaviors,[7] but formulated as explicit minimizations using outputs to drive inputs to a desired value (similar to Wilhelms'[26] use of Braitenberg's behaviors). Control nodes typically receive input signals directly from sensory nodes, and send outputs directly to action nodes, though they could be used in more abstract control situations. Our two control behaviors are:

attract: Create an output signal in the direction of the input signal, but magnified according to distance and angle scalar multipliers and exponents. This node works only when input signals exceed a threshold distance or angle.

avoid: Create an output signal in the opposite direction of the input, magnified according to scalar multipliers and exponents, whenever inputs fall below a threshold distance or angle.

These nodes incorporate both scalar multipliers and exponents, to allow modeling the non-linearities typically observed in animal responses to perceived inputs.[18]

2.1.3. Action Nodes

Action nodes connect to the underlying human body model and directly execute routines defined on the model (such as walking, balance, hand position, and torso orientation) and arbitrate among inputs, either by selecting one set of incoming signals or averaging all incoming signals. An example is the *walk controller*, which

Figure 1: A sample PaT-Net shown graphically

Figure 2: Attraction, avoidance, and terrain awareness

decides where to place the agent's next footstep and then connects to the locomotion generator[3] to achieve the step.

2.2. High-Level Control: PaT-Net Schemas

Low-level control is designed to connect to a general symbolic reasoning process, including a model of parallel automata (PaT-Nets)[6] and various planners. A sample PaT-Net is shown conceptually in Fig. 1. Each net description is a class in the object-oriented sense and contains a number of nodes connected by arcs. Nodes contain arbitrary Lisp expressions to execute as an *action* whenever the node is entered. A transition is made to a new node by selecting the first arc with a true condition (defined as a Lisp expression). Nodes may also support probabilistic transitions where the probability of a transition along an arc is defined rather than a condition. *Monitors* are supported that, regardless of which state the net is in, will execute an action if a general condition evaluates to `true`.

A running network is created by making an instance of the PaT-Net class. Because a running net is actually an encapsulated, persistent object, it may have local state variables available to all actions and conditions, and may also take parameters on instantiation. The running PaT-Net instances are embedded in a Lisp operating system that time-slices them into the overall simulation. This operating system allows PaT-Nets to spawn new nets, kill other running nets, communicate through semaphores and priority queues and wait (sleep) until a condition is met (such as waiting for another net to exit, for specific time in the simulation, or for a resource to be free). Running nets can, for example, spawn new nets and then wait for them to exit (effectively a subroutine call), or run in parallel with the new net, communicating if necessary through semaphores. Because PaT-Nets are embedded in an object-oriented structure, new nets can be defined that override, blend, or extend the functionality of existing nets.

3. Stepper

Stepper is an instance of the two-level (PaT-Net and SCA loop) architecture providing locomotion reasoning and control for simulated human agents in simulated environments.[16] Locomotion reasoning determines the manner in which the agent moves through the world: *i.e.* what types of attractions, avoidances, and posture changes will achieve the goal (Fig. 2).

Figure 3: An Influence

At the low level, Stepper uses an SCA loop to generate human locomotion. A set of influences (combinations of simulated sensors and attraction or avoidance) determine an agent's behavior. At the high level, a set of PaT-Nets schedule and control these influences.

3.1. The Sense-Control-Action Loop for Human Locomotion

Stepper makes use of a framework for general object locomotion (embedded in the *Jack*® software system[3]), which in turn makes use of an SCA loop that performs "anticipatory sensing". That is, in the sense phase, sensors "anticipate" the environment at each potential next foot position, in order to determine in the control phase where the agent should step. The agent takes the chosen step in the action phase. A human steps at discrete positions in a continuous space, and cannot change the targeted step location while a step is in progress.

Individual *influences* bound to the agent effect anticipatory sensing. An influence captures both what aspects of the environment are relevant to monitor and the degree to which they should attract or repel the agent from a particular position. An agent can be influenced by natural features of the terrain (*e.g.,* muddy ground, bodies of water), man-made features of the environment (*e.g.,* sidewalks), locations for which it is headed, *etc.*

An influence determines how an agent acts. In our system, the combination of a sensor and a control behavior (attraction or avoidance) is an influence (Fig. 3). An influence maps a foot position to the *stress* of stepping there. From among the possible choices, the control phase of the SCA loop leads the agent to take the least stressful step.

An influence activates when "bound" to an agent. While active, its output contributes to the stress calculations. Influences may be bound or unbound at any time during a simulation, and hence activated or deactivated. Locomotion is performed by binding influences to humans. The SCA loop, constantly monitoring the environment, immediately initiates the appropriate locomotion.

3.2. PaT-Nets for Human Locomotion

PaT-Nets introduce decision-making into the agent architecture. They monitor the SCA loop (which may be thought of as modeling instinctive or reflexive behavior) and make decisions in special circumstances. For example, the observed behavior resulting from the combined use of different influences can sometimes break down. The agent may get caught in a dead-end or other local minimum. Actions sometimes fail and unexpected events sometimes occur. PaT-Nets can recognize these situations,

Figure 4: ChaseNet State Diagram

Figure 5: Hiders hiding — seeker counting

modify the agent's behavior by binding and unbinding influences, and then return to a monitoring state. During a simulation, PaT-Nets bind and unbind influences in Stepper, thereby altering agent behavior.

Consider an example with Tom chasing Jerry. The ChaseNet shown in Fig. 4 begins in *state 1*. An attraction to Jerry binds to Tom. As Tom begins to run toward Jerry the net passes to *state 2*; the ChaseNet enters the monitoring state. When Jerry ceases to be visible to Tom (Jerry may have run around a corner or behind an object), the net enters *state 3*. An attraction to the location where Jerry is most likely to be found, generally Jerry's last known location, binds to Tom. Tom begins to run toward this location as the ChaseNet transitions to *state 4*. If Tom arrives at this location and does not see Jerry, the ChaseNet transitions to *state 5* and Tom searches in the direction Jerry was last known to be heading.

Clearly, chasing requires reasoning and decision-making beyond the scope of the SCA loop alone. PaT-Nets provide this reasoning, and schedule and control the low-level influences to direct the agent to act in the desired manner.

4. Hide and Seek

Moore, Geib and Reich[15] are building a planning system for synthetic players in a game of "Hide and Seek." It is vertically integrated into a system called ZAROFF that selects reactive behaviors to execute in an animated simulation. By interleaving planning and acting, the players dynamically react to changes in the environment and changes in information about where the other players may be hiding. Adaptivity is also supported through least-commitment planning, as the planner only looks ahead one action at each level of its abstraction hierarchy. The implementation follows the two-level agent architecture: the Intentional Planning System (ItPlanS)[11] interacts with a Search Planner[14] to perform the "high-level" reasoning for the system, and these two components in turn interact with a set of "low-level" SCA nodes based on Stepper (Fig. 5).

4.1. System Architecture

Our division of the control of a player between a *planning component* and a *reactive behavior component* reflects a distinction between deliberative actions (ones requiring non-local reasoning about the past, the present, and possible futures) and non-deliberative actions. In this sense, keeping track of where you are located in a complex environment and what hiding places have been checked requires deliberate effort, while walking from one place to another generally does not. Together, these two components create realistic animations of human decision-making and locomotion while playing hide and seek.

Fig. 6 depicts information flow in ZAROFF. To control the player in the role of seeker, the system starts by initializing the plan with the input goal (finding a hiding human), populating the database with the initial locations of all the objects and human figures in the simulation, and creating a partial map from what the player can see around him. ItPlanS and the SCA loop start processing simultaneously. The planner queries the state of the database through the Filtered Perception module to decide how to elaborate the plan and select an action. If necessary, the Search Planner is consulted to assist in planning how to find things. When ItPlanS decides on an action, it instructs Action Execution to carry it out. Further planning is suspended until the action has terminated (successfully or unsuccessfully).

In making decisions about what to do next, each component makes use of its own internal simulation, which differs from the graphical animation of the environment. ItPlanS uses abstract descriptions of the effects of each action to choose one which will move closer to the specified goal. The Search Planner simulates the movements of an agent on its internal map of the environment. Stepper simulates taking the next step in several alternate locations. At each level of decision making, an internal simulation is used *at an appropriate granularity.*

4.2. Action in ZAROFF

Actions chosen by ItPlanS are carried out by an Action Execution module (see Fig. 6). Both components are well matched to the dynamic environment in which ZAROFF acts: the planner quickly selects the next action to perform based on comparison between the perceived world state and a partial hierarchical plan that is regularly revised. The action execution module controls locomotion via the Stepper system, enabling it to react to unexpected events such as moving obstacles, changing terrain, or a moving goal.[16]

4.3. Planning in ZAROFF

ItPlanS is a hierarchical planner, in which hierarchical expansion only takes place to the degree necessary to determine the next action to be carried out. It consists of an incremental expansion of the frontier of the plan structure to successively lower levels of abstraction. The incremental nature of the plan allows the system to make commitments at the appropriate level of detail for action while not committing the

Figure 6: Information flow in ZAROFF

system to future actions that might be obviated by changes in the world.

4.4. Search planning

A consequence of limited perception is the occasional need to find objects. Our approach is to isolate this reasoning in a specialized module, a Search Planner that translates information acquisition goals to high-level physical goals to explore parts of the environment.

Searches are planned by first identifying known regions where an object may be located and systematically exploring this space. A plan is developed for exploring each region in turn. After such an exploration plan is executed, the environment is observed to determine whether the agent can see an object with the desired properties. During this observation phase, new potential regions may be seen by the agent. These new regions are considered for future exploration as needed.

4.5. Distinctions between the Upper and Lower Level

ZAROFF's stratification into higher-level and lower-level components is a reflection of differences in informational needs. For example, the low-level locomotion behavior requires very detailed information — *e.g.*, foot positions, distances and angles — and a rather fast cycle time (the stepping rate).

In contrast, ItPlanS is responsible for sequencing locomotion actions with actions to open doors in order to explore various hiding places within the game field. The information needed to build plans at this level is at a different level of abstraction as well as a coarser temporal scale. ItPlanS needs to know "Is the door open?", "Am I at the door?", *etc.* While such information can be derived from lower-level information, neither of the modules has need of the information that the other uses.

Separating low-level motor control from high level planning decisions is valid: a symbolic planner is inappropriate for making decisions about foot placement, and likewise local potential field calculations are inappropriate for making long-range plans. While the benefits of adding a reactive controller to a planner are well known, the relationship between these two components is symbiotic. While the reactive controller adds flexibility and an ability to handle local disturbances to the plan, if properly constructed the high level planning can result in the reduction of the complexity of the problem that the controller must solve.

5. Gesture Jack

"Gesture Jack" is a demonstration system that consists of two embodied agents holding a conversation where one agent has a specific goal and the other tries to help achieve it.[9] All parts of the conversation have been automatically synthesized and animated: intonation, gesture, head and lip movements, and their inter-synchronization.

Figure 7: Architecture of each conversational agent

Gesture Jack combines a dialogue planner (which moderates the communicative acts between the agents) with PaT-Nets (which control the speaker/listener roles and various non-verbal aspects of the intercourse). Motor actions drive the face, head, lips, and eyes. PaT-Net schemas control head and eye movements, as these relate directly to the agent's role in the conversation. The face and lips are controlled directly from behavior inputs to the SCA loop, but the absence of direct sensory inputs means that the SCA loop is much simplified in comparison to ZAROFF.

5.1. Gesture Jack Structure

In the Gesture Jack system, we have attempted to adhere to a model of face-to-face interaction suggested by the results of empirical research.[21] In particular, each conversational agent is implemented as an autonomous construct that maintains its own representations of the state of the world and the conversation, and whose behavior is determined by these representations. (For now, the two agents run copies of the same program, initialized with different goals and world knowledge.) The agents communicate with one another only by the symbolic messages whose content is displayed in the resulting animation. (If their SCA loops were fully modeled, they would actually be able to *interpret* the speech and gestures of the other agent!) The architecture of a conversational agent is shown in Fig. 7.

The selection of content for the dialogue by an agent is performed by two cascaded planners. The first is the domain planner, which manages the plans governing the concrete actions which an agent will execute; the second is the discourse planner, which manages the communicative actions an agent must take in order to agree on a domain plan and in order to remain synchronized while executing a domain plan.

5.2. Using PaT-Nets in Gesture Jack

Interaction between agents and synchronization of gaze and hand movements to the dialogue for each agent are accomplished using PaT-Nets, which allow coordination rules to be encoded as simultaneously executing schemas. Each agent has its own PaT-Net: probabilities and other parameters appropriate for an agent are set for the PaT-Net, given its current role as listener or speaker. Then as agents' PaT-Nets synchronize the agents with the dialogue and interact with the unfolding simulation, they schedule activity that achieves a complex observed interaction behavior.

The Gaze and Gesture PaT-Net schedule motions as necessary, given the current context, in semi-real time. They send information about timing and type of action to the animation system. The animation itself is carried out by *Jack*.

Each of the four dialogic functions (planning, comment, control and feedback)[9] appears as a sub-network in the PaT-Net, represented by a set of nodes, a list of conditions and their associated actions. Each node has an associated probability, based on an analysis of two-person conversations, noting where and when a person is gazing, smiling and/or nodding. Each of these signals is binary-valued — *e.g.*, gaze is equal to 1 when a person is looking at the other person, 0 when looking away. The conversation is annotated every tenth of a second. Six turn-states are considered, three per agent. When an agent hold the floor she can be speaking while the other agent is pausing (normal turn) or speaking (overlapping talk or a backchannel signal), or they can be pausing simultaneously. For each of these turn-states, we compute the co-occurrence of signals (nod, smile and gaze) and their probability. Separate PaT-Net nodes correspond to each different turn-state and signal occurrence; for example, the occurrence of a "within-turn signal" corresponds to the action: agent1 looks at the agent2 while having the floor and pausing.

6. Conclusion

We have demonstrated through these three systems that the combination of high level control through planners and PaT-Nets with low level SCA loops yields interesting, human-like, "intelligent" behaviors. Removing any one of these three components would incapacitate performance in notable and indeed crippling ways.

For example, if the planners were removed from Hide and Seek, then all decision-making and actions would need to be encoded in PaT-Nets, including the opportunistic generation of new goals (go to a found hider rather than the location the seeker was heading toward) and a hider's choice of where to hide and a seeker's choice of where to seek next. In the case of the dialogue planner in Gesture Jack, its symbolic reasoning (such as backward chaining or question generation) to determine a series of intermediate steps toward an overall goal would have to be encoded in PaT-Nets. Overloading PaT-Nets with these sorts of reasoning and planning would require a full programming language capability to specify arc transitions and a loss of locality that would be an end to any perspicuity. Indeed, the burden would once again be returned to the animator who would need to virtually "program" the entire agent to select actions and anticipate any contingencies based on whatever features the immediate environment presented.

If the PaT-Net schemas were omitted, the planners would be forced to do too much work to see that sequential or coordinated tasks were carried out. Rather than making the planner disambiguate overlapping or interleaved activities, the schemas can manage resource allocation and coordinated activities. Thus the PaT-Nets in Gesture Jack can manipulate the head nods and eye movements needed for speaker/listener turn-taking without imposing a load on the actual dialog content planner.

If the SCA loop were omitted, the burden of managing all the environmental complexity must be foisted off onto some higher-level (symbolic reasoning) controller.

It appears unrealistic to expect that a symbolic planner worry about where to place a foot during locomotion (Stepper). Likewise, a PaT-Net should not be used to explicitly manage the sensory feedback and decision-making that can check for and avoid hazardous terrain features or other obstacles. The low level SCA loop provides a kind of quick-turnaround "reflex" action which can adapt to situations without requiring cognitive overhead.

We believe that ongoing research into embodied human-like simulated agents will find, as we have, that this architecture of high level schemas and planners combined with low level SCA loops will achieve increasing success in producing intelligent and realistic behavior.

7. Acknowledgements

This research is partially supported by DMSO DAAH04-94-G-0402; U.S. Air Force DEPTH through Hughes Missile Systems F33615-91-C-0001; ARPA DAMD17-94-J-4486; National Library of Medicine N01LM-43551; ARO DURIP DAAH04-95-1-0023; Army AASERT DAAH04-94-G-0220; ARPA AASERT DAAH04-94-G-0362; and NSF CISE CDA88-22719.

8. References

1. N. I. Badler, M. J. Hollick and J. Granieri, "Real-Time Control of a Virtual Human using Minimal Sensors," *Presence* **2(1)** (1993) 82–86.
2. N. I. Badler, B. L. Webber, W. Becket, C. Geib, M. Moore, C. Pelachaud, B. Reich and M. Stone, "Planning for Animation," to appear in *Computer Animation*, ed. N. Magnenat-Thalmann and D. Thalmann (Prentice-Hall, 1995).
3. N. I. Badler, C. W. Phillips and B. L. Webber, *Simulating Humans: Computer Graphics Animation and Control* (Oxford University Press, New York, 1993).
4. W. M. Becket and N. I. Badler, "Integrated Behavioral Agent Architecture," in *The Third Conference on Computer Generated Forces and Behavior Representation* (Orlando, FL, 1993).
5. W. M. Becket, *Reinforcement Learning for Reactive Navigation of Simulated Autonomous Bipeds* (PhD thesis, University of Pennsylvania, 1995).
6. W. M. Becket, The *Jack* Lisp API (Technical Report MS-CIS-94-01, University of Pennsylvania, 1994).
7. V. Braitenberg, *Vehicles: Experiments in Synthetic Psychology* (MIT Press, Cambridge, MA, 1984).
8. R. A. Brooks, "A Robust Layered Control System for a Mobile Robot," *IEEE Journal of Robotics and Automation* (1986) 14–23.
9. J. Cassell, C. Pelachaud, N. Badler, M. Steedman, B. Achorn, W. Becket,

B. Douville, S. Prevost and M. Stone, "Animated Conversation: Rule-Based Generation of Facial Expression, Gesture and Spoken Intonation for Multiple Conversational Agents," *Computer Graphics* (Annual Conference Series, ACM, 1994) 413–420.

10. M. F. Cohen, "Interactive Spacetime Control for Animation," *Computer Graphics* **26(2)** (1992) 293–302.
11. C. W. Geib, *The Intentional Planning System: ItPlanS* (PhD thesis, University of Pennsylvania, 1995).
12. J. K. Hahn, "Realistic Animation of Rigid Bodies," *Computer Graphics* **22(4)** (1988) 299–308.
13. H. Ko, *Kinematic and Dynamic Techniques for Analyzing, Predicting, and Animating Human Locomotion* (PhD thesis, University of Pennsylvania, 1994).
14. M. B. Moore, Search Plans (Technical Report MS-CIS-93-56, University of Pennsylvania, 1993).
15. M. B. Moore, C. W. Geib and B. D. Reich, "Planning and Terrain Reasoning," in *AAAI Spring Symposium on Integrated Planning Applications* (also, Technical Report MS-CIS-94-63, University of Pennsylvania, 1995).
16. B. D. Reich, H. Ko, W. Becket and N. I. Badler, "Terrain Reasoning for Human Locomotion," in *Proceedings of Computer Animation '94* (Geneva, IEEE Computer Society Press, 1994) 996–1005.
17. O. Renault, N. Magnenat-Thalmann and D. Thalmann, "A Vision-Based Approach to Behavioral Animation," *The Journal of Visualization and Computer Animation* **1(1)** (1990) 18–21.
18. C. W. Reynolds, "Flocks, Herds, and Schools: A Distributed Behavioral Model," *Computer Graphics* **21(4)** (1987) 25–34.
19. C. W. Reynolds, "Not Bumping into Things," *SIGGRAPH Course 27 Notes: Developments in Physically-Based Modeling* (ACM SIGGRAPH, 1988) G1–G13.
20. B. Robertson, "Caught in the Act," *Computer Graphics World* **17(9)** (1994) 23–28.
21. K. R. Scherer, "The Functions of Nonverbal Signs in Conversation," in *The Social and Physiological Contexts of Language*, ed. H. Giles and R. St. Clair (Lawrence Erlbaum, New York, 1980) 225–243.
22. K. Sims, "Evolving Virtual Creatures," *Computer Graphics* (Annual Conference Series, ACM, 1994) 15–22.
23. F. Thomas and O. Johnson, *Disney Animation: The Illusion of Life* (Abbeville Press, New York, 1981).
24. X. Tu and D. Terzopoulos, "Artificial Fishes: Physics, Locomotion, Perception, and Behavior," *Computer Graphics* (Annual Conference Series, ACM, 1994) 43–50.
25. B. Webber, N. Badler, B. Di Eugenio, C. Geib, L. Levison and M. Moore,

"Instructions, Intentions and Expectations," *Artificial Intelligence Journal* **73** (1995) 253–269.
26. J. Wilhelms and R. Skinner, "A 'Notion' for Interactive Behavioral Animation Control," *IEEE Computer Graphics and Applications* **10(3)** (1990) 14–22.
27. A. Witkin and M. Kass, "Spacetime Constraints," *Computer Graphics* **22(4)** (1988) 159–168.