May 1994

Terrain Navigation Skills and Reasoning

Hyeongseok Ko
University of Pennsylvania

Barry D. Reich
University of Pennsylvania

Welton Becket
University of Pennsylvania

Norman I. Badler
University of Pennsylvania, badler@seas.upenn.edu

Postprint version. Presented at *Proceedings of the Fourth Conference on Computer Generated Forces and Behavioral Representations*, May 1994, 9 pages.

Terrain Navigation Skills and Reasoning

Abstract
We describe a real-time model of terrain traversal by simulated human agents. Agent navigation includes a variety of simulated sensors, terrain reasoning with behavioral constraints, and detailed simulation of a variety of locomotion techniques. Our Kinematic Locomotion Generation Module (KLOG) generates various terrain navigation skills as well as both rhythmic and non-rhythmic variations of these skills. The terrain navigation skills include curved path walking, lateral or backward stepping, running, and the transitions between walking and running for motion continuity. Locomotion attributes such as pelvis rotation and translation and torso flexion and twist are used to modify the KLOG skills so that realistic looking rhythmic locomotion or non-rhythmic variations, such as ducking under a low hanging branch of a tree, can be achieved. The path through the terrain is incrementally computed by a behavioral reasoning system configuring a behavioral feedback network. A number of sensors acquire information on object range, passageways, obstacles, terrain type, exposure to hostile agents and so on. The behavioral reasoner weighs this information along with collision avoidance, cost, danger minimization, locomotion types and other behaviors available to the agent and incrementally attempts to reach a goal location. Since the system is reactive, it can respond to moving obstacles, changing terrain, or unexpected events due to hostile agents or the effects of limited perception.

Comments
Postprint version. Presented at *Proceedings of the Fourth Conference on Computer Generated Forces and Behavioral Representations*, May 1994, 9 pages.

Terrain Navigation Skills and Reasoning

Hyeongseok Ko, Barry D. Reich, Welton Becket, and Norman I. Badler
Center for Human Modeling and Simulation
University of Pennsylvania
Philadelphia, PA 19104-6389

Abstract

We describe a real-time model of terrain traversal by simulated human agents. Agent navigation includes a variety of simulated sensors, terrain reasoning with behavioral constraints, and detailed simulation of a variety of locomotion techniques. Our Kinematic Locomotion Generation Module (KLOG) generates various terrain navigation skills as well as both rhythmic and non-rhythmic variations of these skills. The terrain navigation skills include curved path walking, lateral or backward stepping, running, and the transitions between walking and running for motion continuity. Locomotion attributes such as pelvis rotation and translation and torso flexion and twist are used to modify the KLOG skills so that realistic looking rhythmic locomotion or non-rhythmic variations, such as ducking under a low hanging branch of a tree, can be achieved. The path through the terrain is incrementally computed by a behavioral reasoning system configuring a behavioral feedback network. A number of sensors acquire information on object range, passageways, obstacles, terrain type, exposure to hostile agents, and so on. The behavioral reasoner weighs this information along with collision avoidance, cost, danger minimization, locomotion types, and other behaviors available to the agent and incrementally attempts to reach a goal location. Since the system is reactive, it can respond to moving obstacles, changing terrain, or unexpected events due to hostile agents or the effects of limited perception.

1 Introduction

Consider the following scenario. An agent begins with an arbitrary position and orientation in an outdoor environment. There is a goal location. The agent moves toward the goal avoiding obstacles, ducking under low branches, climbing over objects, avoiding difficult terrain where feasible, and squeezing through tight spaces where necessary. In some situations the agent tries to avoid the sensory field of one or more hostile agents (hostiles).

Our aim is not to tell the agent how to achieve its goal. Instead, we make the agent aware of its environment and associate a set of actions with different perceptions. An interaction loop is created involving the agent and the environment which converges toward the goal. That is, the agent successfully navigates the terrain and arrives at the goal location.

Animating appropriate locomotive behaviors for all possible situations in an outdoor terrain is not a trivial task. The seemingly infinite number of cases are covered by a finite number of locomotion primitives including walking, running, and lateral stepping with primary and attribute parameters. Primary parameters are the choice of foot (left or right) and the footprint (desired foot location). This allows for arbitrary stepping goals (within certain limits). Attribute parameters control the movement of the torso, pelvis, and swing leg. They can cause the locomotion to be rhythmic, for visual realism, or non-rhythmic, for intentional deviation from normal gaits in such situations as ducking or stepping over low obstacles. In addition to visual realism we were also concerned with the efficiency of the locomotion algorithms since our simulations are real-time.

Simulated sensors are used to detect environmental features such as terrain type, and obstacle and hostile locations. The combined sensory input is analyzed by the behavioral feedback network described in Section 3 and a decision is made as to where the agent should move. This decision is made incrementally, after each step the agent takes, so that a dynamically changing environment can be supported. For example: chasms may open up, trees may fall down, pits or craters appear, or the goal may change.

Once the decision is made where to move next, other sensors are used to examine the local terrain more closely (Section 4). These sensors determine whether or not behavioral modifications must be made and, if so, what the specifics of the modifications are. The agent may have to crawl, crouch, or step over an obstacle.

The behavioral net considers input from all the sensors. It decides where to move the agent, and the appropriate locomotion behavior is invoked.

2 Terrain Navigation Skills

Locomotion is very complicated: the entire body participates and joint rotations generate coordinated rhythmic translation of the whole body. Creating realistic walking motion is far beyond the limit of the basic techniques such as key framing, and requires more special attention than other kinds of human motion. A few algorithms for animating human locomotion have been proposed in the computer animation field[13, 10, 11, 5]. Most of them, however, either do not provide reasonable realism, or have sufficient computational cost as to prohibit real-time simulation.

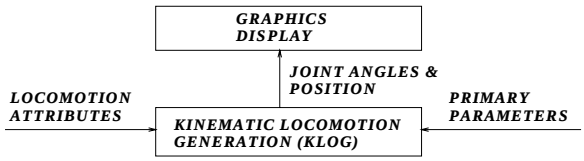

Figure 1: The motor-level locomotion architecture

Above all, they are not general enough to handle the variety of human locomotive behaviors. We present a method for realistic animation of the variety of human locomotion in real-time.

The cases we cover in this paper require more visual realism and computational efficiency than the dynamic soundness of the motion[1]. Thus our algorithm for generating motor level human locomotion primitives is based on kinematics and biomechanical experimental data. It extracts the characteristics of the original measurements on human walking and applies the extracted style in generating the locomotion of different anthropometry in taking an arbitrary step. Much precaution is taken so that the resulting motion is not a vague exchange of legs[2], but all the kinematic constraints imposed on the normal walking are satisfied. Also, the original motion style is preserved during the generalization above [17]

The architecture of our motor level locomotion consists of two major components as shown in Fig. 1: Kinematic Locomotion Generation (KLOG), and Graphics Display (DISPLAY). KLOG generates kinematic locomotion that achieves the current goal (footprint) specified by the high level parameters. The resulting joint angles and global position of the figure root are sent to DISPLAY for kinematic positioning and graphical display[3].

High level parameters are the driving input of the whole motor level locomotion system. The high level parameters consist of the next footprint and the designation of the stepping foot. This *footprint-driven* approach provides a high fidelity control of human locomotion.[4] The high level parameters can be supplied from various sources: reactive or non-reactive path planning systems, virtual reality applications, or interactively constructed spline curves.

[1] Dynamic soundness is considered in Hyeongseok Ko's PhD thesis [14]. The resulting locomotion is dynamically balanced and the motion comfort is maintained within a reasonable range suggested by the human body strength. However, dynamic computation takes non-negligible time in general. Therefore the dynamic control module is bypassed for the applications in this paper.

[2] It is imaginable, and has been demonstrated in other animations, that if the measurements are directly or not carefully used in animating a different anthropometry from the measured subject, which is likely in the most of the cases, visually non-desirable effects can result such as skidding or floating of the supporting foot.

[3] Kinematic positioning and graphical display is done through $Jack^{TM}$ [22].

[4] There are a few other ways of specifying locomotion, such as the path of the COM or pelvis. However, such path does not have the exact control of the foot prints so that the agent can step over a pit on the way.

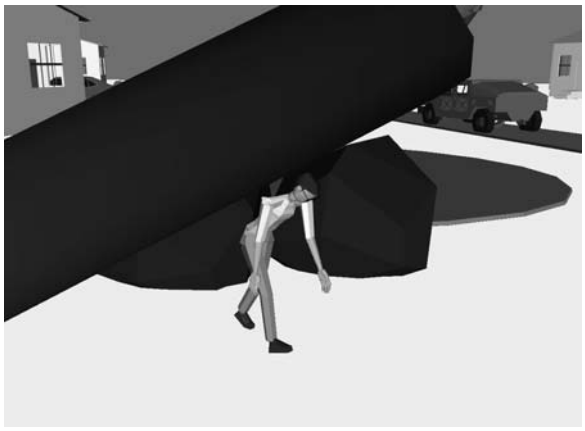

Figure 2: Ducking under a tree

KLOG is *goal-driven*, in the sense that the resulting motion achieves the goal footprint very accurately. Even though some other aspects of motion may be altered by later application of attribute parameters, the goal achievement is not altered. Eighteen attributes such as the pelvis rotation/translation and torso flexion/twist are used to modify the result of KLOG so that rhythmic locomotion style or non-rhythmic variations (e.g, ducking under the low hanging branch of a tree; Fig. 2) can be achieved.

Fig. 3 shows the structure of our KLOG subsystem. KLOG receives high level locomotion parameters and locomotion attributes. There are default values for the attributes, which are for normal, unloaded locomotion. Based on the step distance, direction, and the previous history of stepping, it decides which of the locomotion primitives should be used. Locomotion primitives can be divided into the two groups, Curved Path Locomotion (CPL) which handles the rhythmic case, and Non-Rhythmic Intermittent Stepping (NRS). CPL consists of CFS (Curved First Step), CLS (Curved Later Step), and CES (Curved Ending Step). NRS consists of FWD (Forward Step), BWD (Backward Step), LATERAL (Lateral Step), and TURNAROUND. FWD is similar to CPL, but is static, especially in the foot angle variation. Whenever a new locomotion primitive is added, such as running ([23]) or crawling, we augment the Locomotion Type Determination module appropriately.

Curved path locomotion is obtained through an efficient modification of straight line walking [16]. This generalization method can be applied to any existing straight line walking algorithm. Therefore the straight line walking can be improved independently from the implementation of curved path locomotion. Experiments demonstrate the robustness of this algorithm. Considering that locomotion study has focused on straight line walking, this algorithm adds generality, while maintaining the original realism of straight line walking [18].

Linear path locomotion in turn is obtained by a

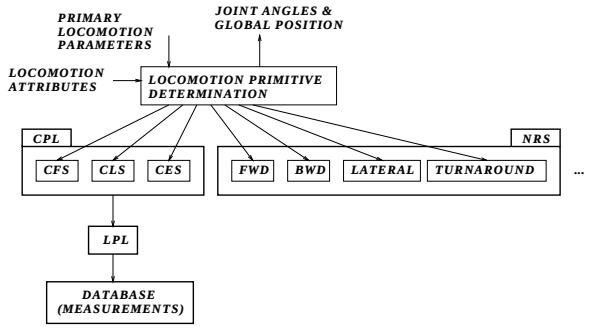

Figure 3: The structure of KLOG

kinematic generalization technique. The motion characteristics are extracted from a set of measured data on human walking, and then applied to generate the walking step of different anthropometry in different step lengths. For all the differences in the bodies and step lengths, the resulting motion of our LPL is very realistic and resembles the style of the originally recorded motion [17]. The Database can have multiple sets of walking data, and LPL can render different walking styles to different agents in the animation scene.

People occasionally take one or two steps intermittently, in a constricted space. Direction is random in this local adjustment. This non-rhythmic stepping behavior is the exceptional case in curved path walking. Forward, backward, lateral stepping, and turning around are considered as the primitives of the local adjustments. Fig. 4 shows the situation when lateral steps are required to pass along a narrow path. (A big log is lying on the ground in parallel and close to a jeep.) Figs. 5-11 show snapshots during a lateral walk. The tree is drawn transparently to show the leg movements. The focus here is not to create a stylistic rhythmic motion, but rather to provide many high level parameters, so that the above four non-rhythmic stepping primitives may cover all random stepping behaviors. One example is the *step height factor*, which is set to one for normal stepping, and can be set to a larger number to step over a low obstacle.

Running is frequently required to navigating in a terrain. It can be used in running over low obstacles or running away from an eminent danger, etc. Rhythmic running and walking is quite different in the timing of the support phase. Therefore, if a running step follows right after a walking step, there will be a discontinuity between the steps. We have implemented[5] a transition mechanism from walking to running and running to walking as well as rhythmic running [23].

3 Behavioral Control

Much of the planning necessary for terrain traversal can be modeled using *emergent behavior*, where complex observed behavior of the agent is the result of several independently modeled behaviors within the agent interacting with the environment and competing with one another for control of the agent. Agent control relying on emergent behavior has had tremendous success in robotics and artificial intelligence (AI) because it can accomplish essential, low-level agent behavior in complex, dynamic environments robustly and in real time. When used on low-level tasks such as navigation, sensor processing, and motor control, the traditional symbolic reasoning techniques tend to be *brittle* in that they will not apply to unforeseen situations or new environments [9], and non-*real-time*: symbolic reasoning tends to be extremely slow and in fact the general symbolic reasoning problem has been shown to be NP-complete [12]. In robotics and AI, the *emergent behavior* approach has been championed by Brooks [8, 9] and by many others (see Maes' collection of papers [20]). In computer graphics, this technique has been proposed independently as *behavioral control* [25, 24, 27]. In this paper we refer to this general technique as the *behavioral* approach.

The pure behavioral approach, however, focuses on low-level control and does not, in general, exhibit long-range planning or memory. The *behavioral architecture* [2] attempts to consider graphics *and* robotics behavioral approaches within the context of a larger symbolic architecture. This allows integrating the benefits of symbolic reasoning, in particular long-range planning, action sequencing, memory due to internal models of the world, and ease of specification with behavioral approaches. Our approach here will be to use a form of this combined approach along with an explicit internal map of the terrain. In this section, we briefly describe the behavioral architecture and the components within it that we use for terrain reasoning.

3.1 The Agent Architecture

The behavioral architecture proposed in [2] is depicted in Fig. 12. There are two primary control paths shown:

the behavioral loop: This is a continuous stream of floating point numbers from the simulated environment, through simulated sensors providing the abstract results of perception, through control behaviors independently attempting to solve a minimization problem, out to simulated effectors or effector behaviors (in our case, walking), which enact changes on the world. This loop continuously operates, connecting sensors to effectors through a network of behavioral nodes which for descriptive convenience are divided into *perceptual behaviors*, *control behaviors*, and *motor behaviors*.

the cognitive pipeline: Symbolic reasoning, or other symbolic or non-behavioral control techniques monitor the behavioral loop and at strategic times, reconfigure the stream of control by altering connectivity, changing connectivity weights, or modifying node parameters. The *behavioral abstractions* manage the transition from floating point signals to discrete symbols and also package behavior into dependable units. This

[5] Running was developed by the collaboration with Byong Mok Oh.

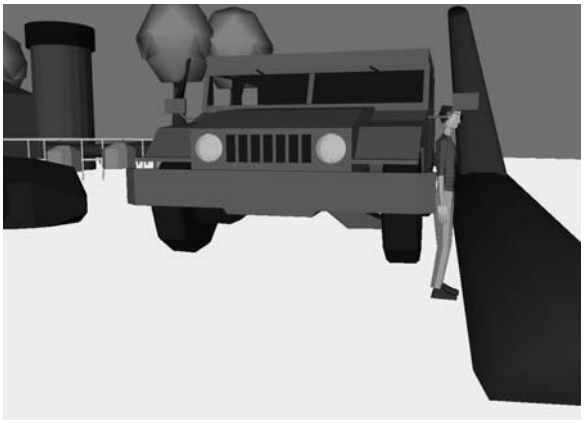

Figure 4: A situation requiring lateral stepping

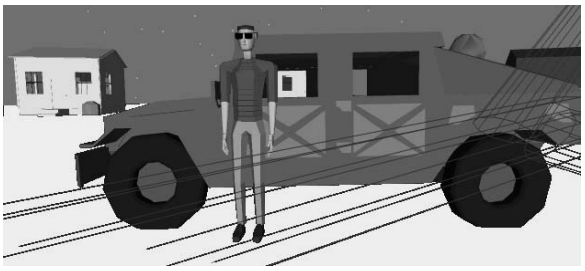

Figure 5: Snapshot during lateral walking

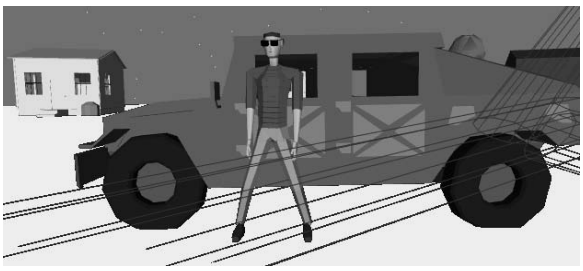

Figure 6: Snapshot during lateral walking

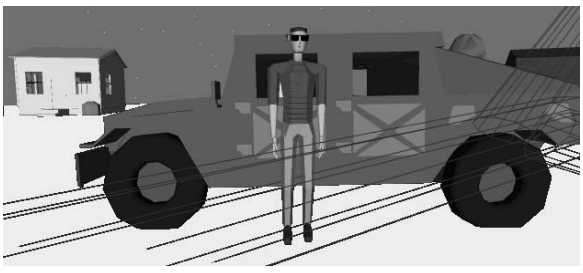

Figure 7: Snapshot during lateral walking

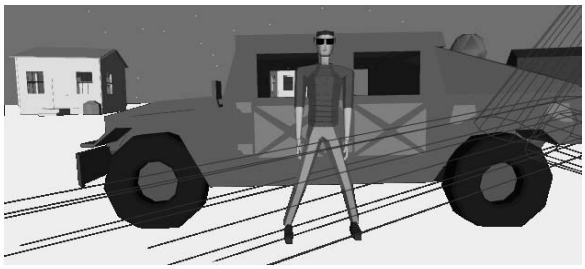

Figure 8: Snapshot during lateral walking

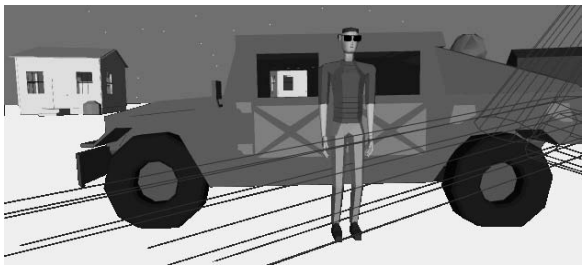

Figure 9: Snapshot during lateral walking

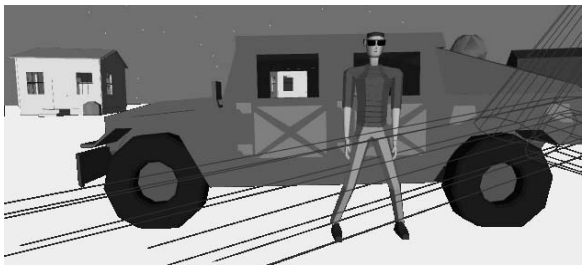

Figure 10: Snapshot during lateral walking

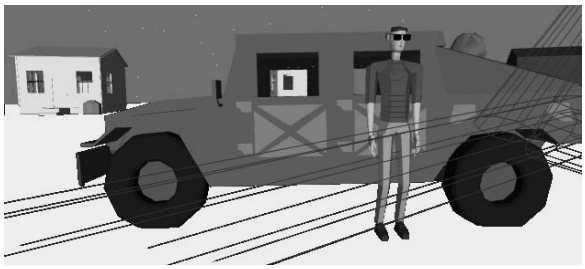

Figure 11: Snapshot during lateral walking

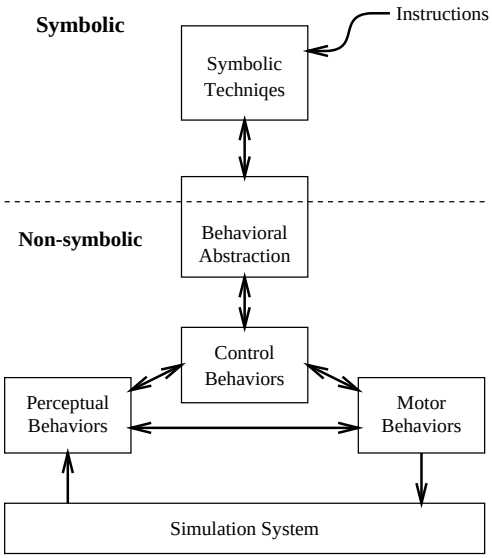

Figure 12: *Behavioral architecture* control hierarchy.

split allows addressing the *symbol grounding problem* (see [19]) and keeps symbolic operations at a sufficiently abstract level.

3.2 Behavioral Nets

The *behavioral loop* is modeled as a network of interacting behaviors, where perceptual, control, and motor behaviors are nodes 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*, and is analogous to a complete behavior in the traditional emergent behavior architectures 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 in [3]).

3.2.1 Perceptual behaviors

The perceptual behaviors used for our terrain reasoning model 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. Two of the perceptual behaviors 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. Note that this sensor directly abstracts over object recognition (which we believe is a fair assumption for a simulated human agent in our circumstance).

range sensors: 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.

Another perceptual behavior is used that perceives an internal map of the terrain as if it were an external entity – this is discussed in detail in Section 4.

These geometric sensors are sufficient for the current abstraction level of our work. However, a more sophisticated approach is to simulate the high-level results of vision of the agent using Z-buffering hardware to create a depth map of what the agent can see. This is the approach used by Renault and Thalmann [24] and Reynolds [26], and we intend to incorporate this approach as our terrain reasoning model progresses.

3.2.2 Control behaviors

For terrain reasoning we use two simple control behaviors loosely based on Braitenberg's *love* and *hate* behaviors [7], but formulated as explicit minimization nodes using outputs to drive inputs to a desired value (similar to Wilhelms' [27] use of Braitenberg's behaviors). Control behaviors typically receive input signals directly from perceptual behaviors, and send outputs directly to motor behaviors, 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 behavior 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 behaviors incorporate both scalar multipliers and exponents, to allow modeling the non-linearities typically observed in animal responses to perceived inputs [25].

3.2.3 Motor behaviors

Motor behaviors 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 decides where to place the agent's next footstep and then connects to the locomotion generator discussed in Section 2 to achieve the step.

A human steps at discrete positions in a continuous space. We assume that the agent cannot change the

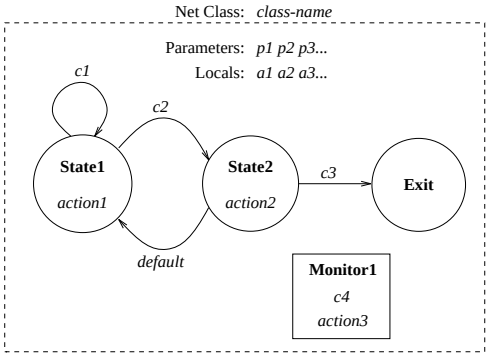

Figure 13: A sample PaT-Net shown graphically

targeted step location while a step is in progress. In order to solve sampling problems that may occur by only sampling at the agent's center at the beginning of every step (especially due to thresholds on avoidance behaviors), the walk controller interprets the incoming attract and avoid signals as a potential field, then steps at a position within the range of possible foot positions with minimum field strength. The field strength for a possible location of the agent is constructed as the sum of absolute values of all angle and distance inputs – which provides a stress value for the position. Discrete positions along a predefined number of arcs in front of the agent are sampled in order to find a suitable minimum position.

3.3 Parallel Automata: PaT-Nets

Although the *behavioral architecture* outlined above is designed to connect to a general symbolic reasoning process, we currently control the behavioral pipeline with a model of parallel automata called *Parallel Transition Nets* (PaT-Nets) [4]. A sample PaT-Net is shown conceptually in Fig. 13. 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 net is time-sliced within the behavioral pipeline. It runs concurrently and constantly monitors the flow of numbers to watch for conditions that will trigger a change of state and possibly a reconfiguration or modification of the pipeline.

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.

4 Sensor-Based Navigation

In order to make intelligent decisions based on the local environment and terrain it is necessary to represent the world on a human scale. A standard must be adopted. Microterrain is too coarse a model for this purpose. The standard chosen is based on a regular grid of squares, one meter on a side. We call this *nanoterrain.*

Each nanoterrain grid element includes the terrain type such as water or grass. Only the terrain is gridded. There are no restrictions on the size, shape, position, or orientation of objects or obstacles in the environment.

4.1 Sensors

The agent utilizes a set of sensors in order to interact with its environment. These sensors can be divided into several classes:

- Attractors
- Repulsers
- Terrain Sensors
- Hostile Agents' Sensory-Field Sensors

4.1.1 Attractors

An attractor draw the agent toward an object. The primary use for this type of sensor is to attract the agent to the goal. Another possible use is to attract the agent to objects to hide behind such as walls. This is important in some simulations where the agent is trying not to be seen. The force this sensor exerts must be strictly less than the force the goal sensor exerts at all times. Otherwise the agent may get stuck in a local minimum near large structures. In a multi-agent simulation, attractors can also be used to keep the agents together in a group.

4.1.2 Repulsers

A repulser exerts a force away from an object. They are particularly useful for collision avoidance. Trees, for example, are detected by repulsers with an infinitely high avoidance force inside, and zero avoidance force beyond a threshold equal to the radius of the tree. This allows the agent to come close to a tree in a simulation, but does not allow the agent to step

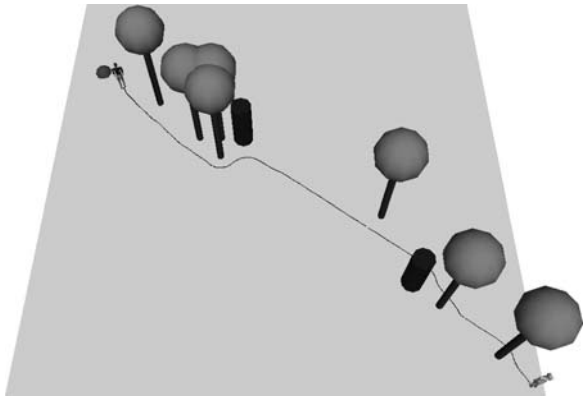

Figure 14: An agent avoiding obstacles

inside one. In a multi-agent simulation, repulsers are also used to prevent agents from colliding with each other.

Fig. 14 is a simple example. The agent begins in the lower right corner. It is fitted with a sensor which attracts it to the goal in the upper left corner. It also senses trees and obstacles and avoids them. The agent walked in real-time towards the goal along the path shown.

4.1.3 Terrain sensor

The terrain sensor evaluates a position and orientation in the environment based on the local terrain-type. Each nanoterrain grid element has an associated weight. Easily navigable terrain such as grass is given a small weight. Regions that are difficult to traverse are given a high weight.

The terrain sensor does more than simply exert a force proportional to this weight. Instead it considers several steps in the direction the agent is facing. The weights for each of these steps are multiplied by a function that drops with distance and summed. The exerted force is proportional to that result. The advantage of this method is that the agent will tend to step into a region of difficult terrain if the region beyond is clear. For example, although the agent tends to avoid water, it will cross a stream if there is a grassy field on the other side.

4.1.4 Hostiles' sensory-field sensor

The agent attempts to achieve the goal location while avoiding the gaze of hostiles. Hostiles' views may be obscured by obstacles, structures, and terrain, and lessened by distance and atmospheric effects. A sensor is used to detect regions visible to the hostile agents and avoid them if possible. This avoidance desire is stronger when the region is closer to a hostile or when it is visible to more than one hostile.

Another consideration is that, although the agent may be standing in a region considered safe, the agent's head may be visible to one or more hostiles. If this is the case a sensor will detect the condition and the agent will crouch or drop to the ground and crawl.

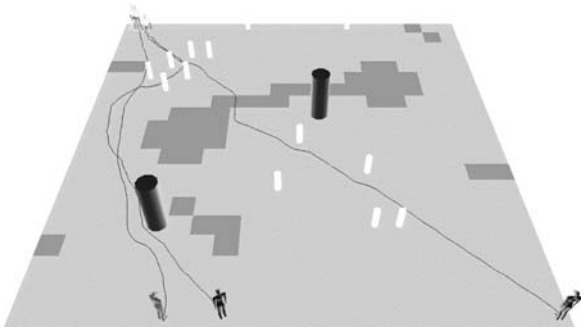

Figure 15: Avoid, attract, and terrain sensors

4.2 Behavioral modifications

Sometimes a path which encounters obstacles is chosen for the agent. When this is the case the agent is forced to make behavioral changes to successfully navigate through the region.

4.2.1 Low obstacles

When the agent encounters low obstacles such as boulders or fallen trees one of three things happen. The agent climbs or jumps over or steps on the obstacle depending on its size and the agent's current behavior. The agent will not jump unless it is currently running.

4.2.2 High obstacles

In the case of a high obstacle such as a tree branch or a door frame the choices are to duck or crawl. The choice is made based on the height of the obstacle.

4.2.3 Narrow widths

When the agent encounters a narrow width it will squeeze through it. This may require turning sideways.

4.3 PaT-Nets for decision making

PaT-Nets are a mechanism for introducing decision-making into the agent architecture. They monitor the behavioral loop (which may be thought of as modeling instinctive or reflexive behavior) and make decisions in special circumstances. For example, the behavior resulting from the combined use of different types of sensors can sometimes break down. The agent may get caught in a dead-end or other local minimum. PaT-Nets recognize these situations, override the "instinctive" behavioral simulation by reconfiguring connectivity and modifying weights, and then return to a monitoring state.

5 Example

Fig. 15 is an example of real-time terrain traversal by three simulated agents. The agents' starting positions at the bottom of the scene and their paths through the terrain are shown.

For this example a combination of three sensor-types was used. Attractors guide the agents toward the goal in the upper left corner, repulsers cause the agents to avoid obstacles and each other, and terrain-sensors cause the agents to avoid water whenever possible. The effect of the terrain-sensor can be seen in this simulation where the rightmost agent chooses to cross the stream at the narrowest point rather than simply walk straight toward the goal.

The winding behavior exhibited by the agent in the center is due to the fact that it was trying to avoid the leftmost agent which was slightly ahead of it in the simulation. Toward the end of the simulation the leftmost agent was standing between a tree and a puddle (upper left). The combination of obstacle and agent avoidance and terrain-sensor output caused the central agent to decide to go around the tree.

6 Discussion

The current trend in designing autonomous agents is away from the *deliberative thinking* paradigm and toward a more direct coupling of perception to action. Flexibility, distributedness, parallelization, and dynamic interaction with the environment are emphasized more and more [21]. Our approach exhibits these features.

Sensor-based navigation as a method of path planning has several desirable properties. The iterative approach allows for a dynamically changing environment which includes changing terrain, obstacle, and goal positions. The fact that only a few local decisions need to be made for each step make the algorithm fast and implementable in real-time. Moreover, it is versatile. New types of sensors may be designed and easily integrated; additional layers of behavioral control may be overlaid on the architecture.

Simulation of locomotive behaviors in arbitrary nanoterrain requires the ability to navigate uneven surfaces, stairs, snow, ice, sand, mud, grass, water, swamp, etc. For some cases, quite different locomotion primitives may be required such as crawling, climbing walls, and swimming. The development of such locomotion skills are in progress.

Acknowledgments

This research is partially supported by ARO DAAL03-89-C-0031 including U.S. Army Research Laboratory (Aberdeen); U.S. Air Force DEPTH through Hughes Missile Systems F33615-91-C-0001; Naval Training Systems Center N61339-93-M-0843; Sandia Labs AG-6076; DMSO through the University of Iowa; NASA KSC NAG10-0122; MOCO, Inc.; Robotics Research Harvesting, Inc.; NSF IRI91-17110, CISE CDA88-22719, and Instrumentation and Laboratory Improvement Program #USE-9152503. We wish to thank Brett Douville for comments and contributions to editing.

Biographies

Hyeongseok Ko received his B.S. and M.S. in computer science from Seoul National University, in 1985 and 1987. He is currently a Ph.D. candidate in Computer Science at the University of Pennsylvania. His research interests include computer animation, human factor analysis, human locomotion, dynamic torque and strength analysis on human motion, and virtual reality.

Barry D. Reich is a Ph.D. candidate in Computer Science at the University of Pennsylvania. His current research includes the design and simulation of virtual sensors. He received his B.S. degree in Mathematics and Computer Science in 1989 from the University of Maryland and his M.S.E. degree in Computer Science in 1991 from the University of Pennsylvania.

Welton Becket is a staff systems programmer at the Center for Human Modeling and Simulation and also a Ph.D. candidate in Computer Science at the University of Pennsylvania. His research areas include reactive planning, machine learning, task-description languages, and maintainability analysis. He received B.S.E. and M.S.E. degrees in Computer Science from the University of Pennsylvania in 1989 and 1990.

Dr. Norman I. Badler, Cecilia Fitler Moore Professor and Chair of Computer and Information Science at the University of Pennsylvania, has been on that faculty since 1974. Active in computer graphics since 1968 with more than 90 technical papers, his research focuses on human figure modeling, manipulation, and animation. He received the B.A. degree in Creative Studies Mathematics from the University of California at Santa Barbara in 1970, the M.S. in Mathematics in 1971, and the Ph.D. in Computer Science in 1975, both from the University of Toronto. Dr. Badler also directs the Center for Human Modeling and Simulation.

References

[1] Norman I. Badler, Cary B. Phillips, and Bonnie L. Webber. *Simulating Humans: Computer Graphics Animation and Control.* Oxford University Press, 1993.

[2] Welton Becket and Norman I. Badler. Integrated behavioral agent architecture. In *The Third Conference on Computer Generated Forces and Behavior Representation*, Orlando, Florida, March 1993.

[3] Welton M. Becket. PhD thesis, University of Pennsylvania, 1994. In preparation.

[4] Welton M. Becket. The *Jack* lisp api. Technical Report MS-CIS-94-01/Graphics Lab 59, University of Pennsylvania, Philadelphia, PA 19104-6389, 1994.

[5] Ronan Boulic, Nadia Magnenat-Thalmann, and Daniel Thalmann. A global human walking model with real-time kinematic personification. *The Visual Computer*, 6:344–358, 1990.

[6] Ronan Boulic and Daniel Thalmann. Combined direct and inverse kinematic control for articulated

figure motion editing. *Computer Graphics Forum*, 11(4):189–202, 1992.

[7] Valentino Braitenberg. *Vehicles: Experiments in Synthetic Psychology*. The MIT Press, 1984.

[8] Rodney A. Brooks. A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, pages 14–23, April 1986.

[9] Rodney A. Brooks. Elephants don't play chess. In Pattie Maes, editor, *Designing Autonomous Agents*, pages 3–18. MIT Press, 1990.

[10] Armin Bruderlin and Thomas W. Calvert. Goal-directed, dynamic animation of human walking. *Computer Graphics*, 23(3):233–242, July 1989.

[11] Armin Bruderlin and Tom Calvert. Interactive animation of personalized human locomotion. In *Proceedings of Graphics Interface '93*, pages 17–23, Toronto, Canada, May 1993.

[12] David Chapman. Planning for conjunctive goals. In James Allen, James Hendler, and Austin Tate, editors, *Readings in Planning*, pages 537–558. Morgon Kaufmann Publishers, Inc., 1990.

[13] Michael Girard and A. A. Maciejewski. Computational modeling for the computer animation of legged figures. *Computer Graphics*, 19(3):263–270, July 1985.

[14] Hyeongseok Ko. *Kinematic and Dynamic Techniques for Analyzing, Predicting and Animating Human Locomotion*. PhD thesis, University of Pennsylvania, 1994. Will be available in 1994 May.

[15] Hyeongseok Ko and Norman I. Badler. Intermittent non-rhythmic human stepping and locomotion. In *Proceedings of the First Pacific Conference on Computer Graphics and Applications, Pacific Graphics '93*, pages 283–300, Seoul, August 1993. World Scientific.

[16] Hyeongseok Ko and Norman I. Badler. Curved path human locomotion that handles anthropometrical variety. Technical Report MS-CIS-93-13, University of Pennsylvania, Dept. of Computer and Information Science, Philadelphia, PA 19104-6389, January 1993.

[17] Hyeongseok Ko and Norman I. Badler. Straight line walking animation based on kinematic generalization that preserves the original characteristics. In *Graphics Interface '93*, Toronto, Canada, May 1993.

[18] Hyeongseok Ko, Byong Mok Oh, Barry D. Reich, Welton Becket, Leanne J. Hwang, and Norman I. Badler. Terrain reasoning for human locomotion - animation. In *Film Festival of Geneva*, May 1994. Animation showing a terrain traversal.

[19] Libby Levison. PhD thesis, University of Pennsylvania, 1994. In preparation.

[20] Pattie Maes, editor. *Designing Autonomous Agents*. MIT Press, 1990.

[21] Pattie Maes. Situated agents can have goals. In Pattie Maes, editor, *Designing Autonomous Agents*, pages 49–70. MIT Press, 1990.

[22] Cary B. Phillips and Norman I. Badler. Interactive behaviors for bipedal articulated figures. *Computer Graphics*, 25(4):359–362, July 1991.

[23] Barry D. Reich, Hyeongseok Ko, Welton Becket, and Norman I. Badler. Terrain reasoning for human locomotion. In *Proceedings of Computer Animation '94*, Geneva, Switzerland, May 1994. IEEE Computer Society Press.

[24] Olivier Renault, Nadia Magnenat Thalmann, and Daniel Thalmann. A vision-based approach to behavioral animation. *The Journal of Visualization and Computer Animation*, 1(1):18–21, 1990.

[25] Craig W. Reynolds. Flocks, herds, and schools: A distributed behavioral model. *Computer Graphics*, 21(4):25–34, 1987.

[26] Craig W. Reynolds. Not bumping into things. SIGGRAPH course 27 notes: Developments in Physically-Based Modeling, 1988. pages G1–G13.

[27] Jane Wilhelms and Robert Skinner. A 'notion' for interactive behavioral animation control. *IEEE Computer Graphics and Applications*, 10(3):14–22, May 1990.