Evolving Legged Locomotion of Virtual Creatures

Daniel Nikovski

Department of Computer Science
Southern Illinois University at Carbondale

Abstract

Virtual reality (VR) systems require realistic simulation of artificial worlds inhabited by virtual creatures. The complex behavior of these creatures has to be guided by intelligent control systems. The paper discusses experiments on combining an explicit planning method with evolutionary computation techniques for the creation of a control system for a six-legged virtual creature modeled in a physically-based simulation environment. The chosen approach permits the incorporation of both explicit and learned knowledge in the control strategy of the virtual autonomous agent.

1 Introduction

A wide variety of Artificial Intelligence (AI) techniques and methods can be applied to the problem of creating control and navigation systems for autonomous agents. A traditional AI planning system consists of a symbolic representation of the world and a search mechanism that operates in this symbolic environment (Korf, 1990). The planning system gives the autonomous agent the ability to perceive the external world, reason consciously about its optimal action policy and implement this policy by controlling its effectors.

A number of serious difficulties exist with the application of this approach to controlling autonomous agents in either real or virtual worlds. First, the sensors and effectors of the autonomous agent operate with numerical data, rather than with symbolic representations. Second, learning in symbolic systems is typically difficult, especially learning that results in structural changes of the representation of the world. Third, a centralized symbolic planning system is slow and non-reactive. All these shortcomings render the traditional AI methodology unsuitable to the problem of controlling virtual creatures in VR environments.

One of the most influential alternative approaches to understanding and simulation of cognition and intelligence is the one that treats intelligence as adaptive behavior. This approach defies the traditional view that cognition is a form of deliberative reasoning that can be studied and simulated through symbolic models of conscious problem-solving processes (Beer, 1990). Instead, the adaptive behavior approach to cognition and intelligence studies the intelligent agent as an autonomous system that interacts in an unconstrained manner with a realistic environment (Brooks, 1989). The task of the cognitive architecture is to connect receptors and effectors so that the autonomous agent performs well in the environment.

This new type of environment, very similar in nature to the environment in VR worlds, poses novel requirements to the cognitive architecture that controls the operation of the autonomous agent. Learning and reactive operation are of primary importance. Numerical learning paradigms such as neural and evolutionary computation can be applied to the problem of learning controlling autonomous agents (Espenschied et al., 1993; Cliff et al., 1993; Ram et al., 1994). This approach results in a significantly different type of cognitive architectures than the traditional symbolic planning systems. The new cognitive architectures are reactive in nature, process information in numerical rather than in symbolic form, and their complex behavior emerges naturally from the interactions of low-level behaviors.

However, these models perform poorly in situations when sequential, non-reactive behavior is required. Along with most high level cognitive functions, even a behavior as simple as walking is essentially sequential. Research has been devoted to demonstrating that sequential behavior can nevertheless emerge as an interaction between low-level parallel interactive processes - neural networks have been evolved that produced a stable gait in simulated hexapods (Beer and Gallagher, 1992).

Is this, however, the best way to implement sequential behavior? A neural network is a *tabula rasa*, a structure that contains initially no knowledge about the nature of the function it is to perform. It is only through long training that it can capture the sequential character of the required behavior. Isn't it better to embed somehow into the control system the explicit knowledge about the modeled function that has already been obtained analytically? For example, if legged locomotion is to be modeled, there is useful explicit knowledge available about the order in which the legs of the autonomous agent should move, so that a stable gait can be produced. It would be advantageous if this explicit knowledge can be incorporated in the control system of the agent prior to modifying and fine-tuning it through learning techniques.

Representation schemes other than neural networks might prove to be more suitable for combining explicit and implicit (resulting from learning) knowledge. The representation scheme chosen for the experiments reported in this paper is a collection of motor programs. McKenna and Zeltzer (1991) used the same representation scheme, however without any learning capabilities. Related representation schemes are Augmented Finite State Machines (Brooks, 1989) (no learning capabilities either), and Classifier Systems (Holland, 1992).

2 Description of the simulation environment

The context in which the cognitive agents operate is essential to the adaptive behavior approach to artificial intelligence. This context is a computer simulated world that provides to the learning system fast feedback about how successful its actions have been. Intuitively, it is clear that the closer the simulation to the real world, the better the chances that the emerged behavior will correspond to the behavior of naturally-intelligent agents. Recent advances in virtual reality research suggest that physically-based modeling can be employed to provide detailed and realistic simulations of the effects of the actions of an autonomous system on the external world (Sims, 1994).

In a physically based system, the modeled object is described by a set of first order ordinary differential equations (ODE) that reflect the dynamics of the parts of the object as formulated by Newtonian and Lagrangian mechanics (Witkin, 1991). One relatively computationally inexpensive possibility is to avoid rigid-body simulation and instead describe the simulated object as a set of particles connected with springs.

The autonomous agent used in the experiments consists of 14 particles and 40 springs (Figure 1). Each of the particles has corresponding 6 first order ODEs (3 for the position in 3D and 3 for the velocity of the particle). This results in a set of 84 coupled first order ODEs. Along with spring forces, gravity and collision forces act upon the particles of the system. The set of ODEs is integrated forward by explicit Euler integration with step 0.0025 time units. The small step-

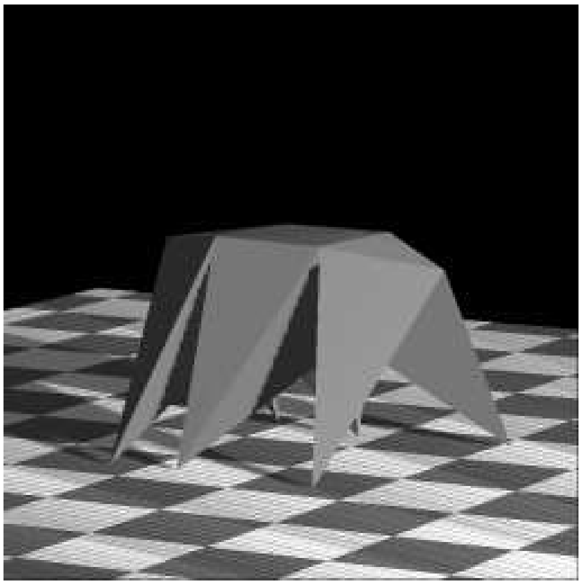

Figure 1: Particles and springs, covered by polygons, form the corpus of the virtual creature.

size is required to provide accurate integration (Raibert and Hodgkins, 1991).

3 Control System

The problem at hand is to make the simulated hexapod walk on a flat surface in a realistic manner. This can be achieved by changing the resting lengths of the springs that connect the particles. In particular, the 20 springs that form the legs of the hexapod need to be contracted or expanded in a certain order in the course of time. Exactly how this must be done is the task of the control system.

While simulating a mass-spring system is relatively simple and computationally inexpensive, controlling such a system presents a real challenge. Classical control system design methods (Raibert and Hodgkins, 1991) will either fail, or will require enormous design efforts, for two reasons. The first is the size of the system of ODEs - 84 equations. All simulated legged locomotion systems reported in the literature have been described by far less equations (Beer, 1990; Raibert and Hodgkins, 1991; Zeltzer and McKenna, 1991). The second, more important problem, lies in the fact that the desired trajectories of the particles are not known in advance, as required by classical control design methods. Indeed, if these trajectories were known, kinematic modeling could be used instead of dynamic one. The primary purpose of using dynamic modeling is to obtain these trajectories in a realistic manner.

A neural network control system could be tried (Beer and Gallagher, 1992), but, as discussed in the Introduction, this would not permit the incorporation of the qualitative knowledge about the desired gait that is available in advance. In this case, the hexapod can first lift and move forward the rear leg on the right hand side, then proceed with the middle and front legs on the same side, and then repeat the same with the three legs on the other side. This would result in a gait pattern called *metachronal wave*, common to many insects (Beer, 1990). At the same time when one of the legs lifts and moves forward, the other legs should move too, in backward direction.

The sequence of actions can be captured in the form of a sequential motor program with the help of a set of six leg positions. Each of the legs of the hexapod has to follow these six positions in turn. The positions form the following movement:

1. Lift the leg
2. Move the leg up and forward
3. Put the leg down
4. Move the leg backward; since the leg is now on the ground, this movement propels the creature forward.
5. Go to step 1.

The motor program consists of a sequence of 19 such elementary instructions. This sequence remains fixed in the course of learning, i.e. learning changes only the resting lengths of the motor springs, but not the sequence of the executed moves.

The sequence of actions can be described as triples of the form *(time, leg, position)*. Even though the changes of the resting lengths of the springs are made at discrete moments in time, the inertial properties of the dynamically simulated particle system make the motion smooth.

Approximate values for the resting lengths of the six chosen positions can be easily calculated geometrically if it is assumed that when a leg is lifted, no other part of the body of the hexapod moves. Of course, this is not exactly so, because in such a case a redistribution of the weight takes place and all springs change their lengths. However, this assumption gives a very useful initial approximation to the correct resting lengths that can be further refined through optimization.

The motor program reflects correctly the synchronization of the actions needed to produce the gait, but it cannot produce a stable gait over a long period of time. After several cycles, the hexapod loses balance and falls down.

4 Evolving the motor program

One possibility to obtain a stable gait is to perturb appropriately the resting lengths for the positions of the walking. It is very difficult to decide which positions should be perturbed and with how much, so a manual adjustment is not feasible. Instead, a suitable search technique can be employed to find the necessary corrections for the six positions of each leg.

Genetic Algorithms (GAs) are search techniques particularly well suited to this problem (Goldberg, 1989). GAs search large spaces quickly by making use of a single objective function, called *fitness* function. No derivative information or continuity of the fitness function are required, unlike other learning techniques such as backpropagation, reinforcement learning, or classical optimization. GAs perform efficient search for a global maximum of the fitness function. This function has to be supplied by the environment.

A natural fitness function for the problem of controlling legged locomotion is the distance that the hexapod travels in a certain amount of time (Sims, 1994). Thus, if the gait is unstable, the hexapod will fall down and penalize the control system that produced this gait. If, on the contrary, the gait is stable, the value of the fitness function will be high. In order to ensure forward motion, only the x-component of the traveled distance was taken for the fitness function. Each control program was tested for 1000 time units (about 15 gait cycles). (Note that with the original unoptimized motor program the hexapod loses balance as early as the end of the third cycle.)

Because of the bilateral symmetry of the hexapod, only half of the 20 springs have to be evolved. With 6 positions per spring, this results in 120 evolvable parameters. Each parameter was encoded with 4 bits as in integer in the interval $(-8; 7)$. Thus, each chromosome consisted of 480 bits.

Over only 10 generations with a population of 4 individuals, the fitness function increased from 1064 to 4070 - more than triple. The roulette-wheel rule was used to generate offspring from the parent population (Goldman, 1989). The fitness reached a level corresponding to stable gait over the whole duration of the testing period. Of course, this result does not mean that the gait would be stable from that point on, because the motor program performs open-loop control.

5 Conclusion

It was shown that a dynamic system of substantial complexity could be easily controlled by incorporating knowledge from two sources - explicit, domain-specific structural knowledge on one hand, and implicit, learned change in parameters on the other hand. This approach can be used where classical control is difficult to apply. In addition to that, the presented solution might be more efficient than one based on neural networks, because the evolutionary search uses explicit qualitative knowledge in the form of a motor program, instead of rediscovering it.

6 Acknowledgements

The BIOBLOCK modeling system and the dynamic simulation toolkit used in the experiments have been developed by Dr. Michael Wainer, Department of Computer Science, Southern Illinois University at Carbondale.

References

[1] Beer, R.D. (1990). *Intelligence as Adaptive Behavior.* Boston: Academic Press.

[2] Beer, R.D. and J. C. Gallagher (1992). Evolving Dynamical Neural Networks for Adaptive Behavior. *Adaptive Behavior*, vol. 1:91-122.

[3] Brooks, R. A. (1989). A Robot that Walks: Emergent Behaviors from a Carefully Evolved Network. *Neural Computation* vol. 1:253-262.

[4] Cliff, D., I. Harvey, and Ph. Husbands (1993). Explorations in Evolutionary Robotics, *Adaptive Behavior*, vol. 2, No. 1:73-110.

[5] Espenschied, K. E., R. D. Quinn, H. J. Chiel, and R. D. Beer (1993). Leg Coordination Mechanisms in the Stick Insect Applied to Hexapod Robot Locomotion, *Adaptive Behavior*, vol. 1, No. 4:455-468.

[6] Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimiztion, and Machine Learning.* Reading: Addison-Wesley.

[7] Holland, J. H. (1992). *Adaptation in Natural and Artificial Systems.* Cambridge: MIT Press.

[8] Korf, R. E. (1990). Real-Time Heuristic Search. *Artificial Intelligence*, vol. 42:189-211.

[9] McKenna, M., and D. Zeltzer (1990). Dynamic Simulation of Autonomous Legged Locomotion. *Computer Graphics*, vol. 24, No. 4:29-38, August 1990.

[10] Raibert, M.H. and J. K. Hodgkins (1991). Animation of Dynamic Legged Locomotion. *Computer Graphics*, vol. 25, No. 4:349-358.

[11] Ram, A., R. Arkin, G. Boone, and M. Pearce (1994). Using Genetic Algorithms to Learn Reactive Control Parameters for Autonomous Robotic Navigation, *Adaptive Behavior*, vol.2, No. 4:277-305.

[12] Sims, K. (1994). Evolving Virtual Creatures,*Computer Graphics*, No.8:15-22.

[13] Witkin, A. (1991). Particle Systems Dynamics, *SIGGRAPH'91 Course Notes on Particle-Based Systems*, Section C. ACM Press.