a slightly earlier version was published in: *From Animals to Animats 2: Proceedings of the Second International Conference on Simulation of Adaptive Behavior* (SAB92), Meyer, Roitblat and Wilson editors, ISBN 0-262-63149-0, MIT Press, Cambridge, Massachusetts, pages 384-392.
On the web at **http://www.red3d.com/cwr/papers/1993/sab92.html**

An Evolved, Vision-Based Behavioral Model of Coordinated Group Motion

Craig W. Reynolds

~~cwr@red.com~~ now: cwr@red3d.com

~~Electronic Arts / 1450 Fashion Island Boulevard / San Mateo, CA 94404 / USA~~

Abstract

Coordinated motion in a group of simulated *critters* can evolve under selection pressure from an appropriate fitness criteria. Evolution is modeled with the Genetic Programming paradigm. The simulated environment consists of a group of critters, some static obstacles, and a predator. In order to survive, the critters must avoid collisions (with obstacles as well as with each other) and must avoid predation. They must steer a safe path through the dynamic environment using only information received through their visual sensors. The arrangement of visual sensors, as well as the mapping from sensor data to motor action is determined by the evolved controller program. The motor model assumes an innate constant forward velocity and limited steering. The predator preferentially targets isolated "stragglers" and so encourages aggregation. Fitness is based on the sum of all critter lifetimes.

1 Introduction

In the work described here, a behavioral controller for a 2d *critter* (animat, artificial animal, autonomous agent, robot, or what-have-you) is obtained through simulated evolution. A stimulus-response controller for a type of *coordinated group motion* begins to emerge under selection pressure from an appropriate fitness measure. The evolutionary process starts with primitive computational elements that provide simulated perception and motor control, as well as the connections between them. The fitness of an individual controller is determined by installing it in a group of simulated critters, placing them in a simulated world and judging their collective performance.

The critters move forward at a constant rate and the controller must "steer" to avoid collisions with static obstacles, moving critters, and a pursuing predator. All collisions are considered fatal. A controller's fitness is based on the total number of simulation steps taken by all of the critters prior to their demise. Over time, under the patient guidance of fitness testing, the evolutionary process constructs an increasingly effective mapping from perception to motor control which allows the critters to get increasingly effective at avoiding the dangers in their simulated world.

This work is not intended to be a realistic model of the evolution of coordinated group motion in natural animals. Instead it provides an abstract example of how such behavior *can* arise in an evolutionary process. It provides a computational model for examining theories about how specific selection pressures and various environmental factors affect the evolution of coordinated group motion. This model provides a framework which can be used to investigate more sophisticated forms of coordinated group motion. The experiments reported here are a first step towards an eventual goal of evolving behavior reminiscent of the fluid, graceful, and visually fascinating motion seen in large groups of natural animals.

In nature, here are many reasons why animals congregate in groups, such as increased access to mates and the group's ability to forage over larger areas. This work focuses on simulating just one aspect of herding: the highly aligned, tightly packed, closely synchronized motion seen when a herd is exposed to a predator. The aim is to have this behavior emerge through artificial evolution given the conflicting selection pressures to: evade predators, avoid obstacles, and coordinate with herd-mates. At one extreme a critter could attempt to avoid everything in its world since all collisions are fatal. Predation pressure confers a survival advantage on those that cluster together. As a result the critters will tend to move close to their herd-mates while collectively avoiding obstacles and collectively evading the predator. Since all critters have the same behavior, coordinating motion is based on the principle that "if I don't run into my herd-mates, they won't run into me". This cooperation is a form of evolved altruism [McFarland 1981].

While admittedly unconnected to natural evolution of corresponding behaviors in real animals, this work aspires to a certain level of plausibility by the *closed* nature of its simulated world. The controller's action is fully determined by the information it obtains about the simulated world through its simulated perceptions. The fitness of a controller is fully determined by the performance of its simulated behavior in the simulated world. The critter's behavior is *grounded* in its perception of the world, and its perception directly reflect the consequences of its behavior.

2 Previous Work

Coordinated group motion exists in many forms in the natural world: flocks of birds, schools of fish, herds of mammals, and colonies of social insects. These behaviors presumably arose through the process of evolution. The work presented here seeks to investigate aspects of the relationship between various selection pressures and the evolution of coordinated group motion.

Zoologists and other scientists have studied coordinated group motion in nature for a long time, see for example [Partridge 1982] and [Potts 1984]. These phenomena are very difficult to study in an objective yet non-invasive way. In recent years, as computer modeling and simulation has provided a concrete way to test theories, many researchers have implemented models of certain aspects of natural coordinated group motion: [Amkraut 1985], [Meyers 1985], [Reynolds 1987], [Girard 1990], [Heppner 1990], [Drake 1991], and [Durkin 1991]. These behavior models were "written by hand" (not evolved) and were based on some type of global knowledge about the environment which is an unnatural model for an autonomous agent.

In contrast, Olivier Renault developed a vision-based behavioral model for obstacle avoidance in [Renault 1990]. Lawrence Dill described a model of coordinated animal motion based on abstracted visual signals in [Dill 1991]. And Helmut Lorek created a vision-based model of flocking that ran on a large parallel computer in real time [Lorek 1992]. These three were non-evolved vision-based implementations.

An evolved, vision-based approach similar to the work described in this paper was used to address the single-critter obstacle avoidance problem in [Reynolds 1993?]. Larry Yaeger's PolyWorld [Yaeger 1993?] is a conceptually vast Artificial Life simulator based (in part) on visual perception and non-goal-directed evolution. In some PolyWorld runs, species have evolved which display locomotion behavior similar to coordinated herding.

Maja Mataric has investigated (non-evolved) coordinated group motion in a group of 20 small robots based on the subsumption architecture [Mataric 1992]. Ronald Arkin has researched strategies for collective robotic action inspired by cooperation in ant colonies [Arkin 1992].

Evolved collective behavior for "central place foraging" in a type of grid-based artificial ant, has been investigated by Robert Collins and David Jefferson [Collins 1991] and by John Koza (see chapter 12 in [Koza 1992]).

A classical work in behavioral modeling is [Braitenberg 1984] which touched on many of the ideas here, but as thought experiments whose implementation was somewhat fanciful. The behavioral controllers described in this paper are very much in the spirit of the *subsumption architecture* described by Rodney Brooks [Brooks 1986]. These *reactive agents* base their behavior directly on the world as perceived through their sensors. They have little or no higher cognition and do not bother with complex mental models of their environment, preferring to use "the world is its own map" [Brooks 1991]. The work reported here is strongly influenced by Dave Cliff's manifesto on *computational neuroethology* [Cliff 1991a] as well as his hoverfly simulation [Cliff 1991b]. While the models described here are not based on neurons, the principles of using a closed, grounded simulation to test behavioral models are fundamental to this project.

This work grew out of conversations the author had with Andy Kopra at the Artificial Life II meeting in Santa Fe in February 1990. Many of the basic ideas presented here (evolved coordinated motion, using the 2d case, vision basing, the predator, evaluation in closed and grounded simulation) came directly from those talks. At almost the same time (or perhaps a little earlier) Stephen Smith and Johnny Ornelas of Thinking Machines Inc. were having amazing similar conversations, and had envisioned strikingly similar experiments. Unfortunately neither groups was able to pursue the project back then due to more immediately pressing matters.

3 Genetic Algorithms, Genetic Programming and the Steady State

At the heart of the work described here is the notion of simulated evolution. The basic evolution model used here is the venerable Genetic Algorithms ("GA"), originally developed by Holland [Holland 1975]. The Genetic Algorithm has been widely studied by many authors and applied to a myriad of practical problems in many different fields [Holland 1992]. Over the years many variations on the basic Genetic Algorithms have been proposed. A hybrid of two such variations are used to implement the simulated evolution described in this paper.

John Koza forged a link between the Genetic Algorithm and computer programming technology with his Genetic Programming paradigm [Koza 1989], [Koza 1992]. Genetic Programming ("GP") is a technique for automatically creating computer programs (traditionally, but not necessarily, in the Lisp language) that satisfy a specified fitness criteria. There is a very strong analogy between the operation of the Genetic Algorithm and Genetic Programming. The main difference is in the representation of genetic information: bit strings in GA, fragments of Lisp code in GP. The use of fitness-proportionate selection, reproduction, crossover, and mutation are all directly analogous. One significant difference is that while classic Genetic Algorithms work on fixed length bit strings, Genetic Programming deals with objects of inherently varying size. The complexity of programs created by GP tend to correspond to the complexity of the problem being solved. If simple programs do not satisfy the fitness function, the Genetic Programming paradigm creates larger programs that do. As a result, Genetic Programming does not require that the user know, or even estimate, the complexity of the problem at hand. This was an important consideration for the goal of coevolving sensor arrays and coordinated group motion. We did not want to specify how many sensors should be used. Genetic Programming did not require such a specification, instead that implicit parameter could be left to evolve its own preferred value.

(Karl Sims independently developed the idea of using Lisp code as genetic material. He used the concept in combination with a system much like *The Blind Watchmaker* ("BW") [Dawkins 1986]. An apt analogy might be to say that GP is to GA as Sims work is to BW. A discussion of crossover and mutation operations can be found in [Sims 1991]. These mutation operations were incorporated into the GP system described in this paper, but the experiments described here did not use mutation.)

Gilbert Syswerda described a variation on traditional GA he called *Steady State Genetic Algorithms* ("SSGA") in Appendix A of [Syswerda 1989], an analysis of their performance can be found in [Syswerda 1991] and [Davis 1991]. A similar technique had previously used in classifier systems and is described on pages 147-148 of [Holland 1975]. Darrell Whitley independently discovered this variation and described it in [Whitley 1989]. While the term "steady state" has apparently become accepted, the comparison of "traditional GA" versus "steady state GA" suggests terms like "batch" versus "continuous" to this author. In any case, the basic idea is to do away with the synchronized *generations* of traditional GAs. Instead there is a continuously updated population ("gene pool") of evolved individuals. Each step of the SSGA consists of fitness proportionate selection of two parents from the population, creation of new individual(s) through crossover and (occasional) mutation, removal of individual(s) from the population to make room, and finally insertion of the new individual(s) back into the population. An additional requirement is that all individuals in the population are required to be unique. (The step of creating new individuals loops until unique offspring are found.) The general observation is that SSGAs are more efficient than traditional GAs in terms of the number of fitness tests required before a given problem is solved. There seems to be some evidence that SSGAs are less prone to premature convergence on suboptimal solutions. In the work reported here, the concept of SSGA was applied to GP to produce a system for "Steady State Genetic Programming".

4 Coordinated Group Motion as Genetic Programming

In order to solve a problem with Genetic Programming we must restate the problem in a canonical form. We must specify a list of *functions*, a list of *terminals*, and a *fitness function*. The Genetic Programming paradigm will evolve programs according to the judgment of the fitness function. The programs themselves are nested Lisp expressions: a function applied to subexpressions. The subexpressions are either one of the terminals or (recursively) another such expression. (These hierarchies are known variously as "s-expressions", "lists", "Lisp fragments", "parse trees" and so on.)

The terminals used in the evolved group motion problem are just an assortment of numerical constants: 0, 0.01, 0.1, 0.5, and 2. The list of functions is:

```
+
-
*
%
abs
iflte
turn
look-for-friend
look-for-obstacle
look-for-predator
```

The functions +, -, and * are the standard Common Lisp [Steel 1990] arithmetic functions for addition, subtraction, and multiplication. Each of them take an arbitrary number of arguments. The function abs is the standard Common Lisp absolute value function which take one argument. The functions % and iflte are suggested in [Koza 1992]. Koza calls % "protected divide", a function of two arguments (A and B) which returns 1 if B=0 and A/B otherwise. The conditional iflte combines the standard Common Lisp functions if and <= into "if less than or equal". In the implementation described here, iflte is a Lisp macro which makes this source-level transformation:

(iflte a b c d) ‘ (if (<= a b) c d)

Using iflte instead of if and <= produces a set of functions which are easily interchangeable: all of the values being passed around are numeric. The Boolean value returned by <= would cause an error if supplied where a number was expected. These “nonviable” programs would not survive to reproduce. Evolution could eventually discover how to plug <= into if by itself, but by smoothing off this rough edge beforehand, we can get GP to focus its effort more directly on the problem at hand. We want survival to be based on increases in fitness rather than details of syntax.

The functions turn, look-for-obstacle, look-for-friend, and look-for-predator are specific to the coordinated group motion problem. (Note that “friend” refers to another critter.) Each of them take a single argument, an angle relative to the current heading. Angles are specified in units of *revolutions*, a normalized angle measure: 1 revolution equals 360 degrees or 2π radians. These functions will be explained more fully below, but basically: turn steers the critter by altering its heading by the specified angle (which is returned as the function's value). The look-for-... functions “look” in the given direction and return a measure of how strongly (if at all) a friend, obstacle, or predator is “seen through the fog”.

We must also provide the *fitness function* that the Genetic Programming paradigm will use to judge the quality of the programs it creates. The fitness function takes one argument, an evolved program, and returns a numerical fitness value. In the implementation described here, fitness values are normalized to lie in the range between zero and one inclusively. A fitness of zero means “totally unfit”, a category that can include programs that get errors during execution. A fitness of one signifies a perfect solution to the problem at hand. For more details see the “Fitness” section below.

Finally there are a few other parameters required to specify a Genetic Programming run. The maximum size of programs in the initial random generation is set to 50. The size of the “steady-state gene pool” (which is roughly comparable to the population in a traditional “batch” generation GA) was set to various values for different runs, but ranged between 20 and 200 individual programs. The mutation rate is zero.

5 The Critter and its World

The critter model used in these experiments is a computer simulation based on widely-used principles of computational geometry and computer graphics. Its simplicity and abstraction make it an equally good (or by the same token, equally bad) model of a living creature, a physical robot, or simply an abstract synthetic creature.

The critter moves on a two dimensional surface. It is essentially equivalent to the LOGO *turtle* [Abelson 1981]. Its state consists of a position and an orientation. In the accompanying illustrations, the critter's body is depicted as a triangle to indicate its heading. For purposes of collision detection, however, its body is modeled as a disk of unit diameter.

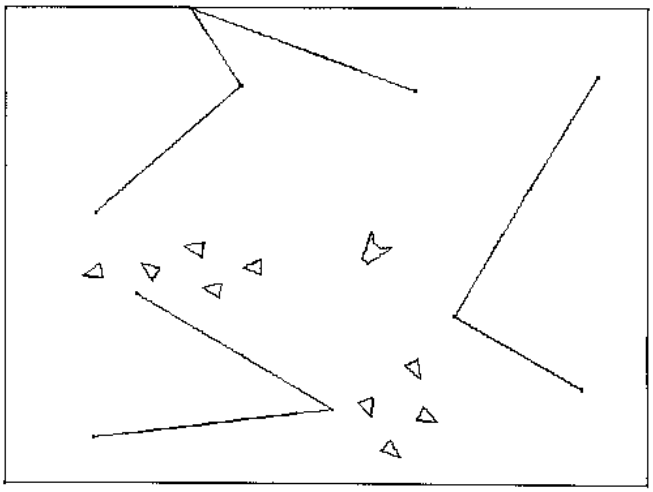

Figure 1: the critter’s world

The simulated world consists of a group of critters, some static *obstacles*, and a *predator*. A critter will die if it *collides* with another critter, an obstacle, or the predator. Collision is defined by a simple distance criteria. In the case of critter/critter or critter/predator collisions, the criteria is overlap of the two bounding disks. In the case of critter/obstacle collisions, the criteria is overlap of the critter's bounding disk with any of the line segments that make up an obstacle. Figure 1 shows some critters, the predator, and the obstacle course (named “Box-and-Fences”) which was used in all of the experiments described here.

There are two kinds of motor actions in the critter's repertoire: *move forward* and *turn*. In the particular problem being studied here, we will assume that forward motion is constant and innate. The critter will always move forward (that is, along its own heading) by one half of its body length each simulation step.

Beyond these fixed innate properties, the critter's *volition* comes from its evolved control program. Typically the controller will use the perception primitives (look-for-obstacle, look-for-friend, and look-for-predator) to get information about the environment, do some conditional and arithmetic processing, and then a turn based on the result.

Various schemes have been used in these experiments to enforce the idea of a non-zero turning radius. In some cases, turning too much is considered fatal. In other cases, large turning angles are allowed but are truncated to a "reasonable" range. Hybrids of these two approaches have also been tried. The maximum per-step turning angle threshold used in these runs is ±0.08 revolutions (29 degrees or 0.50 radians). Both of these approaches will ensure that turning rates are bounded, but the more forgiving "truncation" approach seems less likely to block evolutionary progress by ruthlessly killing off promising new variations which happen to turn a little too much.

The predator is controlled by a hand-crafted program which does not evolve. In general, the predator will select the nearest critter as its prey. The "targeting criteria" is actually a combination of three factors: distance, relative heading, and isolation. Critters that are heading away from the predator are harder to catch and so are considered less desirable. Similarly, the predator prefers isolated "stragglers" and so considers critters in close proximity to others to be less desirable. The predator chases its prey trying to get close enough to "kill" it. Because the predator is only 95% as fast as the critters, they can escape by running directly away from the predator. Rather than heading towards its prey's current location, the predator uses a simple linear predictor and heads to where it "thinks" the prey will be at the time of capture, based on the prey's current heading.

6 Perception

The look-for-obstacle, look-for-friend, and look-for-predator functions simulates the critter's perception and so are the controller's only source of information about its world. All adaptive, time-varying behavior must be derived somehow from the variations of these perceptions as the critter moves through the world. When (for example) look-for-obstacle is called, a *ray-tracing* operation is performed. That is, a geometric ray ("half line") is constructed from the critter's position in the direction specified by the sum of the critter's heading and the argument to look-for-obstacle. The intersection (if any) of that ray with each object in the environment is calculated. The object whose ray intersection is closest to the critter's center is deemed the *visible object*. Note that all objects are treated as *opaque* and so for example, a critter can "hide" from the predator by moving behind an obstacle. Similarly, a critter that is surrounded by other critters can see only them and not any obstacles or predators that lie beyond them.

In order to provide the controller with an indirect clue about distances, we have postulated that the critter's world is a very foggy place. Visual stimuli are attenuated by distance. Certainly this aspect of the simulation can be criticized as being *ad hoc*. But the alternatives are daunting: without attenuation, a critter surrounded by obstacles is in a featureless environment, whatever direction it looks it sees an obstacle. Postulating more complicated sensory mechanisms (such as stereo vision or texture recognition) seemed too complex and would have introduced a new crop of *ad hoc* details to be explained away. The "foggy world" model is somewhat plausible: real fish in murky water face a similar perceptual realm, and the phenomenon known as "aerial perspective" refers to attenuation in air caused by dust and water vapor. If desired, say for robotic experimentation, technology exists (in the special effects industry) for filling rooms with "smoke" of precisely controlled density. The last rationalization for the foggy world model is that it is only slightly different in effect from perception based on active sonar as used by bats and dolphins.

It has been assumed that the critter's visual system can immediately distinguish between the three kinds of object (obstacles, friends, and predators) in its world. While this provides a vast simplification, one plausible explanation is to assume that critters have a form of *color vision* and that obstacles, friends, and predators all have unique primary hues. The implementation described here glosses over these details, but we can imagine that perceived color is decomposed into hue (indicating the type of object) and saturation (indicating distance through the fog).

The value returned from (for example) look-for-obstacle is a number between zero and one. A value of one would indicate that the obstacle is coincident with the critter, but this does not occur in practice since this would imply a collision had occurred. As the distance between the critter and the obstacle increases, the visual signal drops off quadratically in strength. At a certain threshold value (15 body lengths in these experiments) the signal will have reached zero. Hence a value of zero returned from look-for-obstacle indicates that the closest obstacle in the given direction (if any) is at least 15 units away.

Note that the argument to the look-for-… functions are not restricted to be constants and so the process of evolution will often derive bizarre formulations that calculate a dynamic value and use that to specify the direction in which to look. Originally the author had assumed that evolution would create a sort of *retina*, with fixed sensors pointed in the directions of interest. In hindsight it becomes clear that a better model is a simple form of animate vision [Cliff 1991b] where the controller "aims" its visual fovea at the area of interest.

7 Fitness

In order to test the fitness of a newly evolved control program we place the critters in the world, start them running and measure how long they can avoid collisions and preda-

tion. Specifically, the critters are told to take 200 steps. The number of steps taken by each critter (before it dies) is divided by 200 and those values are averaged together to produce the controller's normalized raw score.

The raw score is modified by multiplication by some "style points" (less objective criteria). In order to encourage critters to use generally smooth paths they are penalized for "excessive turning". In order to encourage symmetrical behavior and discourage looping paths, the critters are penalized for "unbalanced turning". Statistics on turning are accumulated during the run, a sum of all turn angle magnitudes, a sum of signed turn angles, and a count of left-versus-right steps. A controller that made the maximum legal turn each time step would receive the "most harsh" excessive turn penalty of 0.5, whereas a controller that never turned would get value of 1.0 (i.e.: no penalty). The penalty for "unbalanced turning" is analogous.

Using a single fitness test will produce a controller for solving the one exact situation represented by the fitness test. However this solution may turn out to be very fragile and opportunistic. It may well have incorporated expectations of incidental properties of the specific fitness test.

To strive for *robust* behavior we need an alternative to a single fitness test. One solution would be to randomize some aspect of the world. This is an appealing approach since fitness trials in nature are effectively randomized. When randomization was attempted in these experiments, it became clear that the noise injected into the fitness values made it very hard for both the human experimenter and for the evolution software to determine if progress was actually being made. Instead each controller was tested on two sets of initial conditions, the starting positions and orientations of the critters and predator are "randomized" using a restartable pseudo-random number generator which is reset for each new controller being tested. This has the effect of discouraging fragile solutions, without introducing uncorrelated noise into the fitness measure.

8 Results

The results of two runs of cooperative group motion evolution are discussed below. Note that the fitness values specified below are normalized to the size of the herd and so are not directly comparable except for identical herd sizes.

In the run called Herd-D there were 20 critters in the herd. The SSGP "gene pool" had a population of 200 programs. Minimum turn radius was enforced by the strict method: critters died if the turned too much. After about 6600 new individuals were created and fitness tested, the program that had attained the best fitness of 12% was:

```
(- (look-for-obstacle 0.01)
   (look-for-predator (turn (look-for-obstacle 0.01)))
   (iflte (turn (look-for-friend 0.1))
          (look-for-predator 0)
          (- (look-for-friend (turn (look-for-obstacle 0.1)))
             (look-for-obstacle 0.1)
             (look-for-friend 0.5))
          0))
```

Astonishingly this program has not been "cleaned up" to make it more readable as is often done when using the Genetic Programming paradigm. This is the unretouched program exactly as evolved. Despite this program's apparent simplicity, analyzing its operation is challenging. Its is apparently trying to avoid collisions with obstacles and friends, based somehow on its relative perception of friends and predators. Figures 2 and 3 show the "trail" of all simulation steps for this program for the two sets of initial conditions.

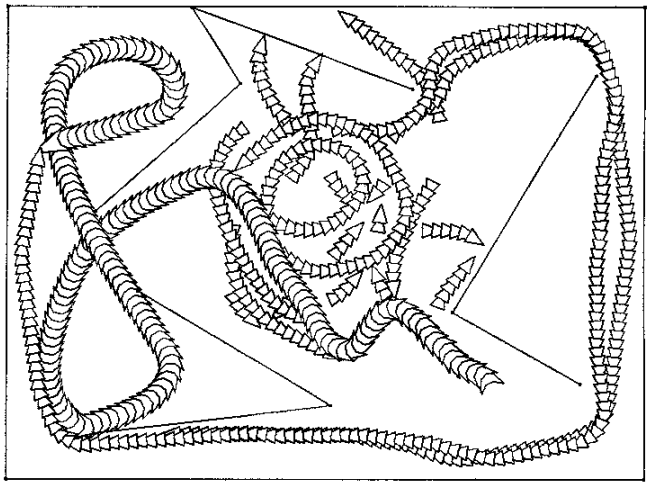

Figure 2: Herd-D, after 6600 runs, first of two trials

The opaque symbols are laid down sequentially, so when two paths cross the one that appears to be on top happened later in time. The wider path is the predator. Sharp turns in the predator's path generally correspond to a "prey capture and retarget" event. Wide, smooth turn usually indicate it is in "prowl mode".

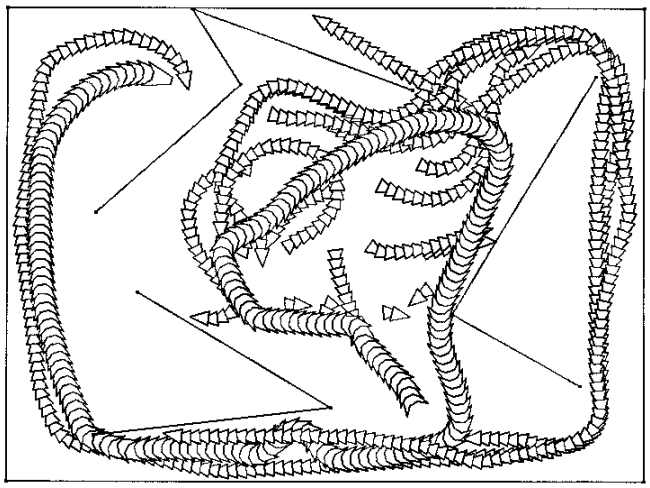

Figure 3: Herd-D, after 6600 runs, second of two trials

It can be seen (in animation if not in these static diagrams) that most of the critters parish early due to collisions or predation while two or three survivors manage to escape into the upper right hand corner and swoop around out of sight of the predator. Figure 4 shows a plot of best-of-run fitness and population-average fitness after every 200 individuals.

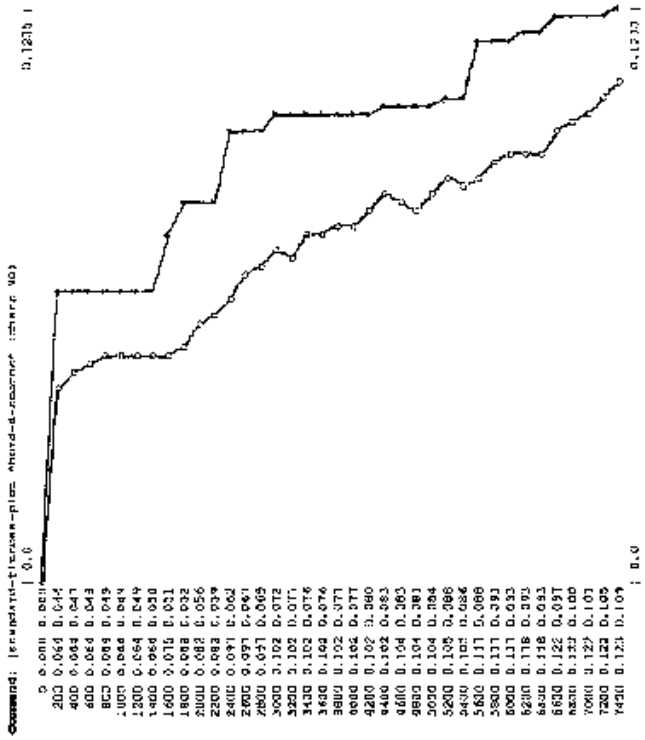

Figure 4: Herd-D, fitness versus time

In the run called Herd-G there were 16 critters in the herd. The SSGP "gene pool" had a population of 30 programs which is quite small. Minimum turn radius was enforced by the permissive method: critters could "ask" for any amount of turn angle, but they would be allowed only a certain amount each simulation step. After about 2000 new individuals were created and fitness tested, the program that had attained the best fitness of 13% was:

```
(iflte (look-for-obstacle (iflte (look-for-obstacle 0.1) (look-for-friend (look-for-friend 0)) (turn (look-for-friend (+ (iflte (look-for-friend 0.1) (look-for-obstacle (look-for-friend 0)) (look-for-predator (look-for-obstacle (turn (+ 0.01 (look-for-obstacle 0.1)))) ) (turn (turn 0))) (look-for-predator (look-for-obstacle 0.1))))) 0)) (* 0.001 (look-for-friend 0.01)) (+ 1 (iflte (look-for-obstacle (iflte (look-for-obstacle 0.1) (look-for-friend (look-for-friend 0)) (turn (look-for-friend (+ (iflte (look-for-friend 0.1) (look-for-obstacle (look-for-friend 0)) (look-for-predator (look-for-obstacle (turn (+ 0.01 (look-for-obstacle 0.1) )))) (turn (turn 0))) (look-for-predator 0)))) 0)) (* 0.001 (look-for-friend 0.01)) 0 (turn (look-for-obstacle 0.1)))) (turn (look-for-obstacle 0.1)))
```

This code is pretty convoluted and essentially impossible to understand. It was not clear to the author that it could be simplified significantly. Worth noting perhaps is that it contains several small "core" strategies such as:

```
(turn (look-for-obstacle 0.1))
(turn (+ 0.01 (look-for-obstacle 0.1)))
(turn (look-for-friend ...))
```

These fragments implement interacting obstacle and critter avoidance. (For example, the first fragment causes turning in the positive direction based on how strongly an obstacle is perceived at an angle of 0.1 revolutions from the current heading. The closer the obstacle is, the sharper the turn will be.) It is not immediately clear if this controller implements predator evasion at all. Figures 5 and 6 show the "trail" of all simulation steps for this program for the two sets of initial conditions.

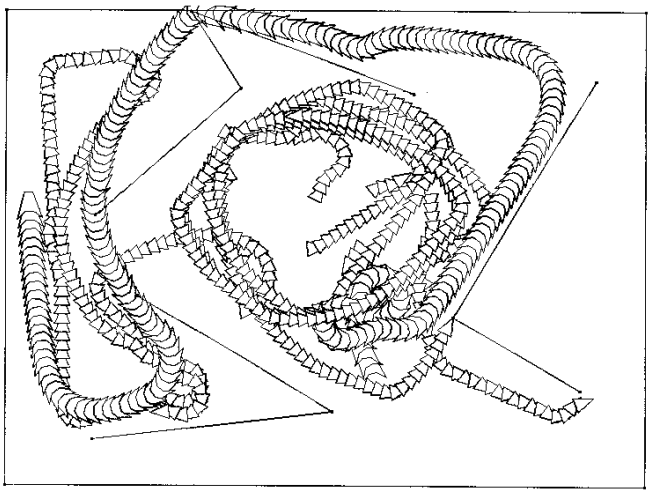

Figure 5: Herd-G, after 2000 runs, first of two trials

Note that in Figure 5 the predator becomes wedged in the concavity at the top center. Since the predator does not die

when it collides with an obstacle, it continues to "tunnel through" to the other side of the obstacle. This is not an evolution problem, it is just a bug in the author's design of the predator's control program.

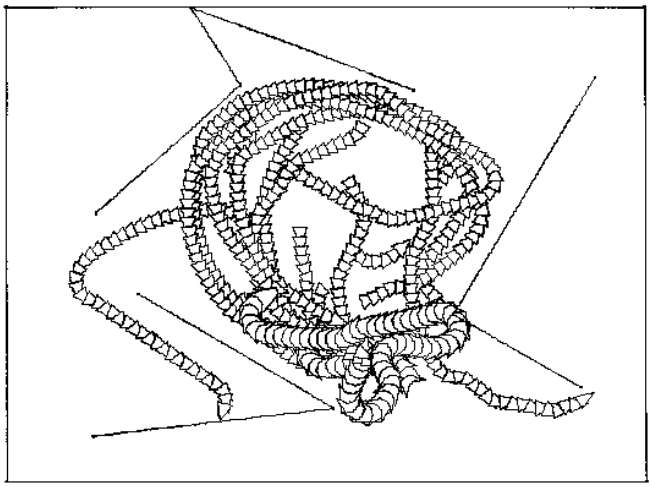

Figure 6: Herd-G, after 2000 runs, second of two trials

Figure 7 shows a plot of best-of-run fitness and population-average fitness after every 30 individuals.

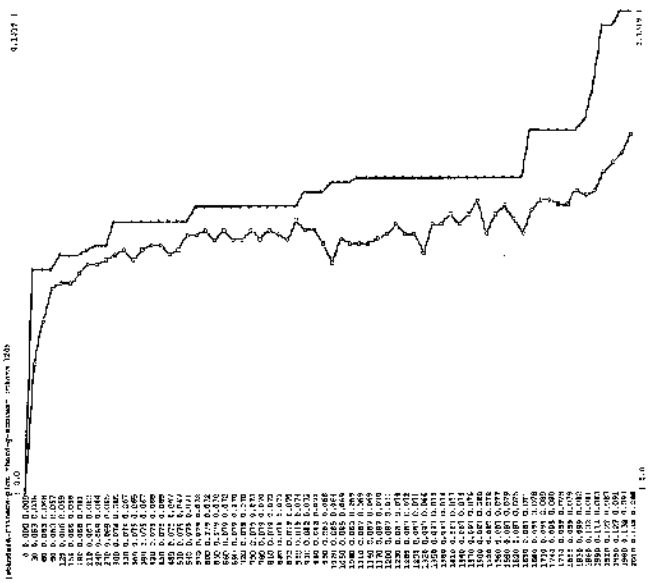

Figure 7: Herd-G, fitness versus time

9 Conclusions

The preliminary results reported here represent only partial solutions to the problem of robust coordinated group motion in the presence of obstacles and under threat of predation. None of the behaviors evolved in this work are anywhere near as robust and general purpose as herding behaviors seen in natural animals. Thoughts about how to proceed beyond these limitations are given in the next section.

On the other hand, there is no question that vision-based coordinated group motion strategies have begun to emerge, given only the requisite primitives, an appropriate fitness measure, and the action of Darwinian evolution through natural selection and survival of the fittest.

10 Future Work

Generally the future goals of this work are to find increasingly competent, robust, and graceful coordinated group motion. In the current model, where the predator is slower than the prey, a herd of really skillful critters should be able to evade the predator indefinitely. We would like to see critters that can form into groups (and so discourage predation) while coordinating their motion to avoid collisions with each others. In the results presented here we can see the beginnings of these behaviors, but they are not nearly as vivid and graceful as the examples seen in natural herds.

One problem with the current model is that while critters can perceive the position of herd-mates and the predator, they can not sense the other's orientation. Presumably this would be very significant information. One simple way to provide this cue would be to consider the "front" and "rear" of the critters (and predator) to have distinct colors. This would allow critters to evolve a different reaction to seeing a herd-mate directly ahead based on its orientation. If a critter sees another critter's "front" directly ahead, it indicates a potential collision which must be avoided. Whereas seeing another critter's "back" directly ahead is normal while herding and requires no reaction.

It is anticipated that better, more robust, coordinated group motion could be be obtained by pitting the critters sequence of ever more sophisticated predators. But that approach puts a significant burden on the human designer of the predators. As the critters become proficient at predator avoidance, it would become increasingly difficult to come up with predators that would challenge them. These considerations suggest that perhaps the predator should coevolve with the critters. This kind of coevolution of problem-solvers and problem-poser has been examined in [Hillis 1990]. The fitness of an evolved predator would be judged by how well it could catch the critters. In the confrontation between a given critter and predator, a capture early in the simulation would reflect poorly on the critter and well on the predator. Conversely, a capture late in the simulation (or an escape) reflects well on the critter and poorly on the predator.

Acknowledgments

The research reported here was pure amateur science. It was not an officially sanctioned project of any corporation, university, or other funding agency. The individuals and companies listed below helped the author out of the goodness of their hearts and their interest in the subject matter. For generous grants of computer facilities, I am deeply indebted to Tom McMahon and Bart Gawboy of Information International Incorporated, Koichi Kobayashi of Nichimen Graphics Incorporated, and Richard Hollander of Video Image Associates. Special thanks go to Andrea Lackey whose workstation I repeatedly commandeered. Heartfelt thanks to Andy Kopra who cheerfully put up with my intrusions to his busy production schedule, and who helped brainstorm many of these ideas back at ALife II. Thanks to: Larry Malone, Dave Dyer, Jay Sloat, Dave Aronson, Joseph Goldstone, Glen Neufeld, DJ, Antoine Durr, and whoever else I forgot. And finally, this work would probably not have happened without John Koza's trail-blazing work on the Genetic Programming paradigm. Many thanks to John and to James Rice for their helpful and encouraging comments during this project.

I'd also like to thank Electronic Arts for ending that stint of "unsupported research" by hiring me recently.

References

Harold Abelson and Andrea diSessa (1981) *Turtle Geometry: The Computer as a Medium for Exploring Mathematics*, MIT Press, Cambridge, Massachusetts.

Susan Amkraut, Michael Girard, and G. Karl (1985) "Motion studies for a work in progress entitled *Eurhythmy*" in SIGGRAPH Video Review, Issue 21.

Ronald C. Arkin (1992) Cooperation Without Commu-nication: Multi-Agent Schema Based Robot Navigation, in Journal of Robotic Systems, 9(3), pages 351-364.

Valentino Braitenberg (1984) *Vehicles: Experiments in Synthetic Psychology*, MIT Press, Cambridge, Massachusetts.

Rodney Brooks (1986) A Robust Layered Control System for a Mobile Robot, in *IEEE Journal of Robotics and Automation*, 2(1).

Rodney A. Brooks (1991) Intelligence Without Representation, in *Artificial Intelligence* 47, pages 139-160.

Dave Cliff (1991a) Computational Neuroethology: A Provisional Manifesto, in *From Animals To Animats* proceedings of SAB90, Meyer and Wilson editors, MIT Press, Cambridge, Massachusetts.

Dave Cliff (1991b) The Computational Hoverfly; a Study in Computational Neuroethology, in *From Animals To Animats* proceedings of SAB90, Meyer and Wilson editors, MIT Press, Cambridge, Massachusetts.

Robert Collins and David Jefferson (1991) AntFarm: Towards Simulated Evolution, in *Artificial Life II*, Langton *et al.* editors, Addison-Wesley.

Lawrence Davis, editor (1991) *Handbook of Genetic Algorithms*, Van Nostrand Reinhold.

Richard Dawkins (1986) *The Blind Watchmaker*, Harlow Logman.

Lawrence Dill (1991) Predicting the 3D Structure of Animal Aggregations from Function Considerations: the Role of Information, (not yet published) presented at the NSF workshop on *Animal Aggregations: 3D Measurement and Modeling*, Monterey Bay Aquarium, October 1991.

Tom Drake and Julia Parrish (1991) Computer Simulation of Fish Schooling, (not yet published) presented at the NSF workshop on *Animal Aggregations: 3D Measurement and Modeling*, Monterey Bay Aquarium, October 1991.

James Durkin (1991) An Implementation of a Behavioral Model of Aggregate Animal Motion, (not yet published) presented at the NSF workshop on *Animal Aggregations: 3D Measurement and Modeling*, Monterey Bay Aquarium, October 1991.

Michael Girard and Susan Amkraut (1990) Eurhythmy: Concept and Process, *The Journal of Visualization and Computer Animation*, 1:15-17.

Frank Heppner and Ulf Grenander (1990) A Stochastic Non-Linear Model for Coordinated Bird Flocks, in the *Ubiquity of Chaos* (Saul Krasner editor), AAAS Publications, Washington, pages 233-238.

W. Daniel Hillis (1990) Co-Evolving Parasites Improve Simulated Evolution as an Optimization Procedure, in *Emergent Computation*, Stephanie Forrest editor (a special issue of *Physica D* 42) pages 228-234, MIT Press/North-Holland.

John Holland (1975) *Adaptation in Natural and Artificial Systems*, The University of Michigan Press, Ann Arbor, Michigan.

John Holland (1992) Genetic Algorithms, in *Scientific American*, July 1992.

John R. Koza (1989) Hierarchical Genetic Algorithms Operating on Populations of Computer Programs, in *Proceedings of the 11th International Joint Conference on Artificial Intelligence*, Morgan Kaufmann, San Mateo, California.

John R. Koza (1992) *Genetic Programming*, MIT Press, Cambridge, Massachusetts.

Helmut Lorek (1992) personal communication, publication anticipated, contact lorek@informatik.uni-oldenburg.de for details.

David McFarland (1981) *The Oxford Companion to Animal Behavior*, page 102, Oxford University Press, Oxford.

Maja J. Mataric (1992) Minimizing Complexity in Controlling a Mobile Robot Population, *Proceedings, IEEE International Conference on Robotics and Automation*, pages 830-835.

Rob Meyers, Peter Broadwell, and R. Schaufler (1985) PLASM: Fish Sample, an installation piece at the acm SIGGRAPH 1985 Art Show.

Brian L. Partridge (1982) The Structure and Function of Fish Schools, in *Scientific American*, June 1982, pages 114-123.

Wayne K. Potts (1984) The Chorus-Line Hypothesis of Maneuver Coordination in Avian Flocks, letter in *Nature*, volume 30, pages 344-345.

Olivier Renault, Nadia Magnenat Thalmann, and Daniel Thalmann (1990) A Vision-Based Approach to Behavioral Animation, *The Journal of Visualization and Computer Animation*, 1:18-21.

Craig W. Reynolds (1987) Flocks, Herds, and Schools: A Distributed Behavioral Model, in *Computer Graphics*, 21(4) (SIGGRAPH '87 Conference Proceedings) pages 25-34.

Craig W. Reynolds (1993?) An Evolved, Vision-Based Model of Obstacle Avoidance Behavior, submitted to *Artificial Life III*, forthcoming.

Karl Sims (1991) Artificial Evolution for Computer Graphics, in *Computer Graphics*, 25(4) (SIGGRAPH '91 Conference Proceedings) pages 319-328.

Guy L. Steele Jr. (1990) *Common Lisp the Language*, second edition, ISBN 1-55558-042-4, Digital press.

Gilbert Syswerda (1989) Uniform Crossover in Genetic Algorithms, in *Proceedings of the Third International Conference on Genetic Algorithms*, pages 2-9, Morgan Kaufmann Publishers.

Gilbert Syswerda (1991) A Study of Reproduction in Generational and Steady-State Genetic Algorithms, in *Foundations of Genetic Algorithms*, pages 94-101, Morgan Kaufmann Publishers.

Darrell Whitley (1989) The GENITOR Algorithm and Selection Pressure: Why Rank Based Allocation of Reproductive Trials is Best, in *Proceedings of the Third International Conference on Genetic Algorithms*, pages 116-121, Morgan Kaufmann Publishers.

Larry Yaeger (1993?) Computational Genetics, Physiology, Metabolism, Neural Systems, Learning, Vision, and Behavior or PolyWorld: Life in a New Context, submitted to *Artificial Life III*, forthcoming.

[revised January 10, 1993]