Rabbitstew: A Robot Simulator with Variable Morphologies

Ethan G. Jucovy

December 14, 2005

Abstract

I propose to design a realistic physical simulator for robots in Python which can easily use and manipulate automatically generated as well as designed robot morphologies. Following the completion of this simulator I propose to evaluate the relative strength of coevolutionary methods to conventional evolution in robot bodies using a genetic process to evolve creatures in simulation to perform a simple competitive task. I will allow the physical structures of one population of robots to evolve along with their control mechanisms while a second population remains in a fixed, human-designed body, and the relative successes of these two populations will be compared. The present paper describes my long-term goals and gives the broad implementation details of the proposed simulator.

1 Motivation

In most robotics research, emphasis lies on improving the "brain" or control procedures in a fixed physical form. In evolutionary robotics, control is generally "evolved" (frequently in simulation) through multiple generations by evaluating individual fitness in a large, non-uniform population and applying selection, crossover, and mutation to the population until successful brains are developed. For the most part, this process occurs in a single human-engineered robot body which has proven itself reasonably durable and successful both in and out of simulation, this often being a model of a commercially available research robot.

In the past fifteen years, however, another trend has been gaining popularity: that of coevolution, or holistic evolution[1], whereby a robot's morphology, as well as its control parameters, is allowed to vary and be subject to selection pressures.

Several theories have been advanced to explain why holistic evolution is preferable and results in more successful robots. Three major theoretical lines of argument have been advanced. The first is a simple argument from biological observation: brains and bodies evolved together in nature resulting in carefully tuned creatures with brains specifically designed to control the bodies they are in, rather than general purpose brains that can just as easily be used to control completely different bodies. The second, related argument takes a developmental perspective: holistic evolution avoids potentially constraining human bias and, proceeding by a large number of small and gradual modifications to both brain and body, allows a tighter coupling of brain and body to emerge.[7] The third is a theoretical mathematical argument which will be discussed later. However—somewhat surprisingly—no direct, side by side comparison has ever been done to confirm that holistic evolution *is*, in fact, a better approach.

For this reason, I propose to compare holis-

[1] The accepted term in the literature is the former, "coevolution"; however, following [9], I prefer the term "holistic evolution" as more descriptive and less ambiguous.

Proceedings of the Class of 2006 Senior Conference, pages 34–44,
Computer Science Department, Swarthmore College ©2005

tic evolution to the conventional evolution of control parameters in fixed morphologies[2]. I will evolve two populations of robots in simulation using a competitive selection method to maximize evolutionary pressures: one population will contain robots with varying, initially random morphologies, and will be evolved holistically; the other population will have only its control structures evolved within a fixed architecture modelled on a real, commercial research robot such as an ActivRobots Pioneer (`www.activrobots.com`) which is generally accepted to have good design and to have the ability to perform well in a wide variety of tasks.

By the nature of the competitive, variable fitness function, which depends crucially on the other members of the population, there is no obvious, objective way to compare the relative success of two populations of robots. As a substitute, at periodic intervals throughout the evolutionary process I will take the best-performing members of each population and compete them against each other to determine which population has evolved "more successfully." As my sympathies do lie with the proponents of holistic evolution, I expect that, *eventually*, the holistically evolved population will outperform the other, perhaps by quite a consistent and wide margin, though I expect that the opposite will be true during the early stages of the evolutionary process.

In order to perform this experiment I will need a simulator which allows robot morphologies to be specified easily and modified automatically without additional code compilation or intervention from the user. Unfortunately, no such simulators are currently available. Therefore my first step must be to develop a simulator which will allow me to run this experiment.

In the present paper I describe the details of both my proposed simulator, the Robotic Artificial Brain/Body-Intertwined Simulation Toolkit and Evolution Workshop, or Rabbitstew, and of the experiment that I will perform on the simulator. In the first section I discuss related work in the field of holistic evolutionary robotics, and I describe the present state of commercially and freely available robot simulators; in the second section I give the proposed implementation details of both the physical simulator itself and the separate three-dimensional graphical display program; and in the final section I describe the experiment which motivates the design of the simulator.

[2]For lack of a better term, I will refer to this as "conventional" evolution.

2 Related Work

2.1 Holistic Evolution Research

Salmon[9] offers a more complete overview of work done on holistic evolution and summarizes some of the empirical and theoretical arguments in favor of holistic evolution.

2.1.1 Proof of Concept

Karl Sims[10, 11] successfully demonstrated holistic evolution, essentially as a proof of concept; starting with completely random morphologies, spreading control structures across the robots' bodies, and giving the robots a competitive task to complete, he produced a number of interesting robots with widely varying morphologies, many of whose strategies in the competition were essentially quite simple and depended crucially on their particular physical structures in ways which conventionally evolved robots could not have developed in a fixed form.

Sims evolved five groups of creatures, each with a different fitness metric: one group was evolved to quickly take control of a block in the center of an arena and prevent an opponent creature from doing the same; one group was evolved to locomote by swimming in an underwater environment; one group was evolved to walk on land, another to hop in a low-gravity world, and another to quickly follow a point of light. However, he did not compare this in any way with non-holistically evolved robots as his motivation was primarily to create thought-provoking computer art and to demonstrate the feasibility of holistic evolution rather than its advantages.

Thomas Ray[8], in a follow up to Sims' work, implemented a similar procedure, replacing the automated fitness selection functions of the original study with ratings provided by human observers to select for increased aesthetic and emotional appeal of the resulting reatures.

Somewhat troublingly, Ray estimates that up to 90% of over 300 "interesting genomes" that resulted, many of which were able to swim and crawl successfully, were found by the random generation which initializes a population. However, he does say that evolution improved his creatures, and it is unclear whether the 90% he cites were found *purely* by random initialization or whether some they were enhanced by at least some evolution.

Furthermore, Ray was selecting for subjective aesthetic qualities more than for stability and success at a well-defined task in a physical environment; indeed, one of the creatures he selected for reproduction had an unfortunate tendency to explode under the strain of its own internal forces. It is likely that, with more physically challenging selection pressures, the importance of evolutionary processes would increase while the likelihood of finding successful individuals purely by random search would decrease significantly.

Pablo Funes and Jordan Pollack[6] provide independent supporting evidence for the potential of physical structure evolution in addition to control evolution. To demonstrate that physical structure can effectively be generated by a genetic process, they evolved passive objects such as bridges, cantilevers, and crane arms in simulation using simulated Lego[3] blocks. Perhaps confirming the developmental argument that undirected search techniques such as simulated evolution remove restrictive human biases, their resulting structures were often unusual and counterintuitive, but perfectly functional even when reproduced with physical Lego blocks.

2.1.2 Advantages of Holistic Evolution

Providing some evidence of the value of holistic evolution, Balakrishnan and Honavar[1] performed a semiholistic evolution in a very limited, idealized environment, where fitness measures were fixed based on success at a block-pushing task. In their first trial they performed conventional evolution of neural networks in a fixed structure: a robot with eight non-intersecting short-range sensors. In subsequent trials the number, range and placement of sensors on the robot could be modified during the evolutionary process. Using this limited holistic evolution, performance as measured by the static fitness function did not drop, and, interestingly, the number of sensors was generally minimized despite no pressure for efficiency being built in. In addition, when sensor range was evolved in addition to number and position, peak performance increased while sensor number still remained relatively low. These results suggest that holistic evolution may help robots discover solutions that are both more effective and more efficient than would be discovered with conventional evolution in a fixed form.

Further evidence that holistic evolution can improve efficiency was provided by Bongard and Paul[3], who also used a limited form of holistic evolution to show that performance and efficiency can be improved by allowing evolution to modify physical structure. In this case, variable parameters in the morphology consisted of the lengths of the body segments and the dimensions and placement of weights on the legs of a biped robot whose task was

[3]Lego is a registered trademark of the Lego group.

to walk as far as possible in a given amount of time. The populations with variable morphologies both performed better on average and had higher peak performances than those with static morphologies, providing additional empirical support for holistic evolution; however, as in Balakrishnan and Honavar's work, the holistic evolution performed was fairly limited.

Conrad[4], meanwhile, provided a theoretical argument in favor of holistic evolution, called the extradimensional bypass. Any complex fitness landscape will contain peaks and valleys, and two adaptive peaks which may be separated by a valley in a low-dimensional landscape may be connected by an adaptive ridge if additional dimensions are added to the landscape. As a holistic approach to evolution includes more free parameters than a conventional approach, the fitness landscape consists of many additional dimensions; so adaptive ridges may exist between what would be peaks in a lower dimensional (fixed body) manifold of the same space.

Stober and Gold[12] performed a variety of experiments, with tasks ranging from wall following to object gathering, to evaluate the potential advantages of evolving neural network morphologies, as opposed to merely evolving the weights of the network as in conventional evolution. They found, however, that performance did not improve significantly when network morphologies were allowed to evolve, and in fact successful strategies found by the two populations were quite similar.

2.2 Existing Simulators

There exists at present no simulator up to the task of performing holistic evolutionary development. Sims' work was done on a massively parallel architecture with custom code that cannot be made publically available. Ray, Bongard and Paul used MathEngine, an apparently now defunct real-time physical simulation package produced by MathEngine PLC, Oxford; Bongard's more recent research and that of Pollack have used code which employed the Open Dynamics Engine and are not publically available.

The open source Player/Stage package does contain a three dimensional rigid body simulator, Gazebo, designed in ODE and OpenGL, as well as Player itself, a network server for robot control. However, specifying a robot morphology for use in Player and Gazebo is slow and labor intensive; a Player device must be developed to specify how the robot interacts with its sensors and actuators, while a Gazebo model must be hard-coded and compiled to describe the robot's physical body. Other simulators are equally problematic: Webots and JRoboSim require users to describe robots similarly to the Player/Stage project and are not intended for morphological evolution; the Graphical Workshop for Modelling and Simulating Robot Environments (Gwell) and Robsim only allow robot models to be hand-created by the user in a visual runtime environment; Easybot requires precompiled libraries to specify robot controllers and does not perform collision detection or other physical modelling.

The Laboratory of Intelligent Systems (LIS) at the Ecole Polytechnique Fédérale de Lausanne has been developing several evolutionary robotics tools including Enki, a fast two-dimensional simulator capable of simulating large groups of robots; Teem, a software framework for evolutionary robotics experiments; and Goevo, an application for evolving neural network controllers for real or simulated robots. Unfortunately LIS does not currently have any tools available for evolving robot morphologies, and the pieces of their evolutionary framework that do not directly rule out morphological evolution are not designed with this in mind.

Table 1 contains download and documentation locations for the software mentioned above.

3 Rabbitstew

3.1 Tools

I will write the simulator in Python. For the physics of my simulator I will use PyOde, a set of Python bindings for the Open Dynamics Engine (ODE), a native C library for rigid body dynamics with built in support for fast collision detection and joint connections between bodies. ODE is quickly becoming a standard library for simulation of evolutionary robotics and will be well suited to this task. I will implement a visualization of the simulation separately using the Visual module for Python. Neural network implementation will be carried out using the Conx module of the Pyro programming environment. See Table 1 for locations of online resources for this software.

3.2 Simulator Implementation

Rabbitstew is composed of two major parts: the data structures which fully describe robot genotypes for storage and manipulation, and the physical instantiation of the robots active in the simulation. Passive features in the simulation environment are not considered to be a separate category but are instead treated identically to robots; a passive structure such as a block or a wall can be defined simply by creating a robot with no brain.[4]

3.2.1 Robot Genotypes

The design of robot genotypes in Rabbitstew is broadly based on that described by Sims[11] with elements of Ray's[8] implementation.

An individual robot genotype will be represented by a directed graph which is composed of Nodes and Connections including an arbitrary root Node. Each Node represents one body unit and contains a Segment and a list of Connections. Each Connection represents a physical attachment between the two body units specified in the parent and child Nodes and will contain a child Node, a relative position, orientation, and scale, two bits representing joint type (hinge, ball, slider or fixed), and a direct-recursive limit parameter since circuits are permitted in the graph. To prevent infinite indirect recursion, robot phenotypes will have a universal, externally imposed, user defined size-ratio limit, a restriction on the number of body segments relative to the size of the genotype.

A Segment will consist of two bits representing shape (box, sphere, or cylinder) and physical dimensions for the Segment, with the number and type of dimensions dependent on the particular shape of the Segment. Shape dimensions are relative and will be normalized to have identical volume; absolute dimensions of a given Segment will determined by a combination of the parent and child Segments' dimensions and the scale factor of the connection between them.

Each Segment will also contain a Brain which is represented in genotype as a directed graph of neural units with graph connections storing the network weights between units. Three types of neural units will be available: Sensors, which get input from the environment, Effectors, which send torque outputs to the parent Joint of the Segment in which they are located, and Neurons, which are purely processing units with inputs from and outputs to other neural units. Available Sensors will include binary contact Sensors which are active if and only if the associated Segment is currently in contact with any other physical body and sets of three direction Sensors which give the normalized direction from the center of the associated Segment to a particular source (available sources will be the center of the target and the center of the root Seg-

[4]This may be changed in a later release of the simulator if it is found to be unsatisfactory; its major advantage, and the motivation for this organization at least in the initial release, is its simplicity.

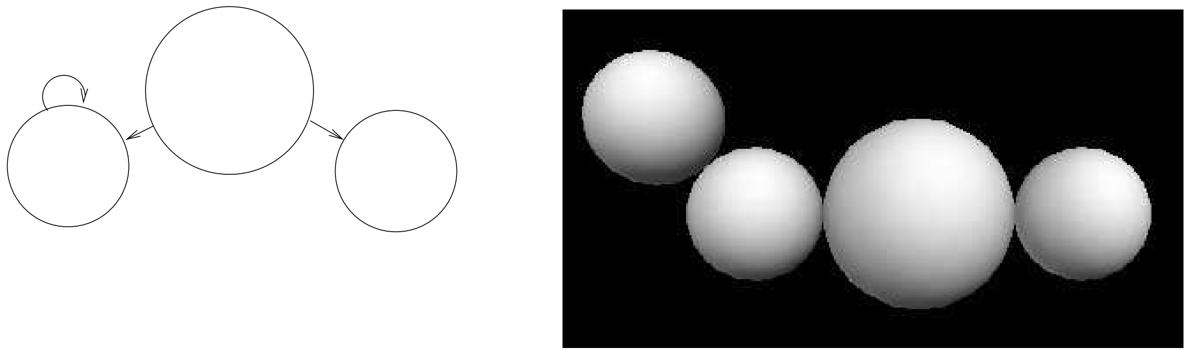

Figure 1: A sample morphological graph (attributes and parameters, including the single-level recursive limit parameter for the circuit, not shown) and the resuling structure.

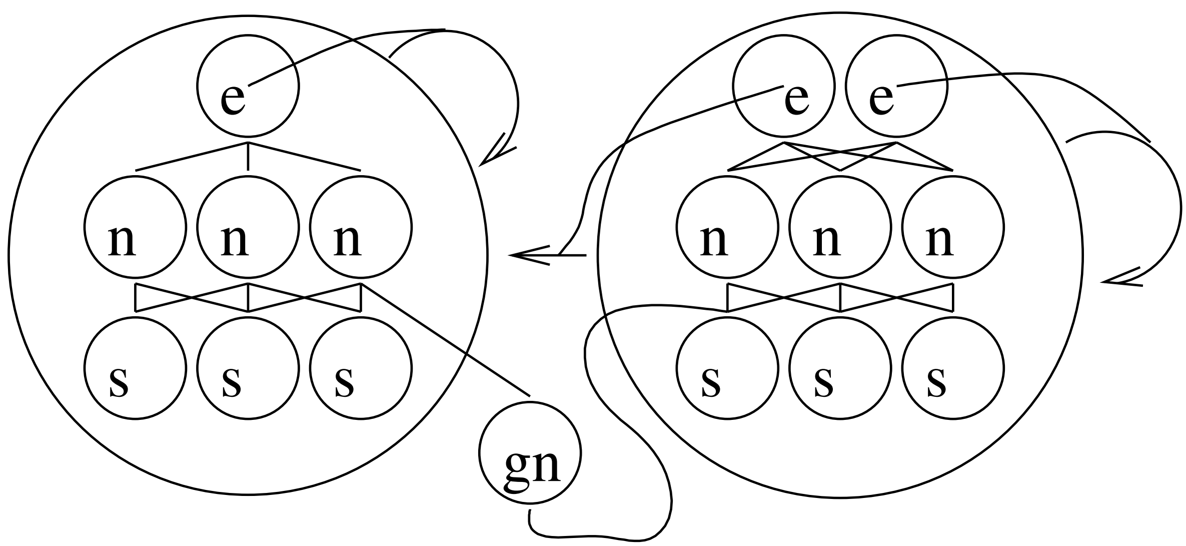

Figure 2: A graph with embedded and global computational units (Neurons, Sensors, Effectors) shown. Effector outputs to local joint connections are represented by lines to the relevant connection.

ment of the opponent in the competition).[5]

A global Brain, not associated with any particular Segment, will also available; this Brain will consist entirely of Neurons but can be connected to any localized neural units to permit centralized control. With the exception of this central Brain, all individual neural units in a Brain can only be connected to other units in the same Brain.

Note that there are no constraints on the number or types of neural units in a Brain aside from the limitation that the central Brain, if it exists at all, be composed only of Neurons. Therefore it is possible to create a fully distributed robot controller, by using no centralized Neurons at all, or a fully centralized one, by limiting the embedded neural units only to Sensors and Effectors, in addition to hybrid controllers with some localized and some centralized processing.

3.2.2 Physical Structures

When the simulation starts, each robot which will be active in the competition is synthesized from its genotype; when the simulation is terminated the synthesized robots are destroyed. While this can result in a considerable number of redundant graph traversals if a robot is active in multiple simulations, the redundant computation is only done at the initialization of the simulation and therefore should have no effect on individual simulation performance, and the the memory saved by "cleaning up" robots between simulations, which would affect simulation performance if too much memory was used, more than offsets the additional time to resynthesize robots.

[5]Additional Sensors and Effectors could be made available later and might result in more interesting and varied behavior, but for simplicity the initial sets of Sensors and Effectors will be limited to these few. Additionally, the direction Sensors should and will ultimately be generalized to give the direction to any arbitrary (fixed or moving) target for a more general-purpose simulator that can be used for other experiments.

Synthesis of a robot proceeds from the designated root Node. In an attempt to increase the number of graph nodes that are actually synthesized into at least one body part before the size-ratio limit halts the process, a breadth-first search will be used to traverse the graph. With this method, however, a low size-ratio limit combined with high recursive limit parameters on connections could still result in creatures with some graph nodes that are not synthesized into any body parts. This is intended to make the evolutionary process more complex and realistic, since many genetic features in nature are passed down without manifesting in every generation.

It should be noted that this approach still results in a direct, one to one mapping between genotype and phenotype; such a direct mapping has been argued[5] to be problematic in evolutionary processes as it ignores the many stages of growth, does not scale well to large organisms, and fails to impose constraints of symmetry, which has been demonstrated to increase organisms' efficiency[2]. Regardless, I feel that the current model is a reasonable compromise between genotype-phenotype complexity and simulator simplicity.

At each Node up to three ODE objects are created and initialized: a Body, which primarily contains an object's mass properties; a corresponding Geom object, which describes the object's spatial extent for collision detection; and, for all Nodes but the root, a Joint between the object's Body and the Body of the parent Node, which physically links two objects and provides motion constraints between them. Each robot will be stored in a list of Geoms and an ODE JointGroup. The Geom class has a function which returns the Body to which it is linked so no separate list of Bodies needs be maintained.

At each Node the corresponding neural network will also be synthesized from the local graph describing a Brain; following the synthesis of individual units the central Brain will be constructed as well. Each Brain will be implemented in the physical simulation as a standard neural network.

3.3 Visualizer Implementation

At the start of a physical simulation the specifications of each robot are written out to a data file for subsequent use by the visualizer. Each body unit of each robot is described, in turn, first by a number specifying the shape of the unit and then by that unit's *absolute* dimensions, which can vary from one (for a sphere) to three (for a box) floating-point numbers.

At each n timesteps during the simulation (by default $n = 1$ but this can be overwritten by the user) the state of the simulation is written to the data file. This consists of a three dimensional position vector and an orientation quaternion (both provided by ODE accessor functions) for each ODE Body, where the states of Bodies are given in the same order as the initial data. A single small robot with three body units, therefore, would yield a state output of 21 single-precision floating-point numbers per output cycle.

When the visualizer starts it will first read in unit data, one unit at a time, creating a three dimensional model of each unit using the Python Visual module and storing the units in a single list. Models will then be initialized to the starting position and orientation given by the first set of state data and all models will subsequently be set visible. Visualization proceeds in a loop by reading in each set of time-dependent state data and updating all models accordingly until all data has been processed or the visualization is terminated by the user.

For spheres and boxes, position in three dimensions is given directly by the first three values of the 7-tuple representing a unit's state. Unfortunately slightly more computation is necessary for cylinders, as ODE represents a cylinder's position by its center of mass and the Visual module represents a cylinder's position by the central point at one end of the cylinder. Orientation is determined from the last four values which give the unit's orientation quaternion q by setting the object's axis vector to the last three terms of the result of the operation $q' \times (0, 1, 0, 0) \times q$ and its up-directional vector to the final three terms of the result of the operation $q' \times (0, 0, 1, 0) \times q$, where $q' = (q_0, -q_1, -q_2, -q_3)$.

4 Long-Term Experiment

After the simulator has been developed, I will run two evolutionary algorithms, one on a population of randomly-generated morphologies and employing holistic evolution; the other on a population with uniform, engineered morphologies and evolving only the weights of the neural network control structures.

I intend to use a all-versus-best two-at-a-time competition[6] similar to that of Sims[11], where pairs of robots from the population will be physically simulated as they compete in a time-limited zero-sum game. The evaluation of fitness I will use will be a ratio of the two robots' center-of-mass distances from the center of the "world" after a short period of simulated time; so robots would have to find strategies both for quickly reaching the goal point and for preventing their opponent from doing so. This competitive, zero-sum fitness function should result in fairly fast and dynamic evolution with a wide range in populations.

At periodic intervals throughout the evolutionary process I will select the two or three highest-performing members of each population in the current generation and I will have these "champions" compete, in the same task, with one another. These interpopulation competitions will have no effect on the evolution of

[6] Variations on competition size and structure are of course possible and could be employed for follow-up experiments.

Software	URL
Easybot	http://iwaps1.informatik.htw-dresden.de/Robotics/Easybot/
Enki	http://lis.epfl.ch/resources/enki/
Goevo	http://lis.epfl.ch/resources/evo/
Gwell	http://diablo.ict.pwr.wroc.pl/\%7epjakwert/
JRoboSim	iwaps1.informatik.htw-dresden.de/Robotics/JRoboSim/
MathEngine	www.mathengine.com (website down)
ODE	www.ode.org
Player/Stage	playerstage.sourceforge.net
PyOde	pyode.sourceforge.net
Pyro	www.pyrorobotics.org
Robsim	http://www10.brinkster.com/geniusportal/robsim.html/
Teem	http://lis.epfl.ch/resources/teem/
Visual	www.vpython.org
Webots	www.cyberbotics.com

Table 1: Software locations.

each group; they will be used simply to measure and compare the progress of each population.[7] In this manner I will evaluate the relative success of each population as compared to the other at different stages of evolution.

I predict that in the early stages of evolution the holistically evolved population will perform far worse than the stable-body population, by virtue of the latter's having an effective and functional body to work with; the latter group essentially has a significant head start by having a pre-engineered body. However, I predict that this will eventually reverse, and that the holistically evolved population will ultimately outperform the stable-body population, for several reasons.

First, by having their brains and bodies develop simultaneously, the holistically evolved robots will be able to take full advantage of their physical structure and will be able to evolve relatively uncomplicated strategies that hinge on the particular constraints and idiosyncrasies of their bodies, and they will simultaneously be able to modify their physical structures to complement and build upon the strategies that they have already developed.

Second, their population will be considerably more variable and therefore selection pressure to find general-purpose and adaptable strategies may be stronger than in the stable-body population. And third, as posited by Conrad[4], the extra dimensions of the evolutionary search space may allow for additional "ridges" in the solution landscape connecting what in a lower-dimensional space would be two separated peaks.

Following this experiment a number of avenues for further study will be available, involving modifications of a number of parameters in the experiment. In addition to varying the competition structure and the sensors and body parts available to the evolving robots, a variety of tasks could be employed besides the proposed "control the center" goal. Unfortunately it is quite difficult to predict the effects of any of these modifications, so I intend to consider them only after completing

[7]Some obvious alternatives include periodically pitting the best performer of each population against each of the members of the opposite population at the same generation, or competing each member of the two populations; best vs. best has the (dubious) advantages of simplicity and quicker processing time, but other methods could be used for additional follow-up analysis or for further experiments.

the present experiment.

5 Acknowledgements

Many thanks to Josh Bongard, Douglas Blank, Claudio Mattiussi, Jordan Pollack, John Rieffel, Branen Salmon and Karl Sims for their support and their advice regarding presently available simulators. Jordan Pollack, John Rieffel and Josh Bongard also referred me to the Open Dynamics Engine. As noted above, many of the implementation details of my simulator are due to Karl Sims; much of the organization of the project resulted from recommendations by Branen Salmon. Thanks also to Cortland M. Setlow who explained the method described above for determining visual orientation by quaternion multiplication, and finally to Ben Kuperman for his frequent encouragement and review of my project.

References

[1] Karthik Balakrishnan and Vasant Honavar. On sensor evolution in robotics. In John R. Koza, David E. Goldberg, David B. Fogel, and Rick L. Riolo, editors, *Genetic Programming 1996: Proceedings of the First Annual Conference*, Stanford University, CA, USA, 28–31 1996. MIT Press.

[2] Josh Bongard and Chandana Paul. Investigating morphological symmetry and locomotive efficiency using virtual embodied evolution. In J.-A. Meyer et al., editor, *From Animals to Animats: The Sixth International Conference on the Simulation of Adaptive Behaviour*, 2000.

[3] Josh C. Bongard and Chandana Paul. Making evolution an offer it can't refuse: Morphology and the extradimensional bypass. *Lecture Notes in Computer Science*, 2159, 2001.

[4] Michael Conrad. The geometry of evolution. *Biosystems*, 24:61–81, 1990.

[5] Frank Dellaert and Randall D. Beer. Toward an evolvable model of development for autonomous agent synthesis. In R. Brooks and P. Maes, editors, *Artificial Life IV Proceedings, Proceedings of the Fourth International Workshop on the Synthesis and Simulation of Living Systems*, Cambridge, MA, 1994. MIT Press.

[6] Pablo Funes and Jordan Pollack. Computer evolution of buildable objects. In P. Husbands and I. Harvey, editors, *Fourth European Conference on Artifial Life*, pages 358–67, Cambridge, MA, 1997. MIT Press.

[7] Jordan B. Pollack, Hod Lipson, Gregory Hornby, and Pablo Funes. Three generations of automatically designed robots. *Artificial Life*, 7:215–23, 2001.

[8] Thomas S. Ray. Aesthetically evolved virtual pets. In Carlo C. Maley and Eilis Boudreau, editors, *Artificial Life VII: Workshop Proceedings*, Portland, OR, 2000. Reed College.

[9] Branen Salmon. Embodied evolution in a morphologically heterogeneous population of robots. `http://web.cs.swarthmore.edu/~meeden/cs81/projects/salmon.pdf`, 2003. Swarthmore College Senior Seminar Project.

[10] Karl Sims. Evolved virtual creatures. In *Computer Graphics Annual Conference Series*, pages 15–22, July 1994.

[11] Karl Sims. Evolving 3d morphology and behavior by competition. In R. Brooks and P. Maes, editors, *Artificial Life IV Proceedings, Proceedings of the Fourth International Workshop on the Synthesis and Simulation of Living Systems*, Cambridge, MA, 1994. MIT Press.

[12] Jeremy Stober and Jonah Gold. Evolvable morphologies for robot controllers. `http://web.cs.swarthmore.edu/~meeden/cs81/projects/stober-gold.pdf`, 2003. Swarthmore College Senior Seminar Project.