Infusing Virtual Creatures with Vision

Marcin L. Pilat and Christian Jacob

University of Calgary, Calgary, AB, Canada
(pilat, jacob)@cpsc.ucalgary.ca

Abstract

In this paper, we evolve light following behaviours in pre-evolved virtual creatures. The neural controllers of creatures evolved for movement are augmented with simple visual neurons and neural connections. The resulting neural networks are trained for light following by an evolutionary algorithm. We show that, through this process, we are able to train the neural controllers to follow a light source. Many of the evolved behaviours show stability and adaptiveness to environmental perturbations of body orientation.

Introduction

The ability to sense the environment is crucial for natural life. Living things have evolved an assortment of different sensing capabilities such as vision (light), hearing (sound), and feeling (vibrations) in order to find food or avoid predators. In many animal species, these sensory systems are also used for communication, enabling social interactions, group dynamics, and emergence of higher level biological phenomena. Sensing is made possible by the interplay of physical structures (morphologies) and sensory elements (neural networks). Thus virtual creatures are an ideal test-bed of sensing models. Wake (2001) argues that the study of morphologies and development will provide clues to answer many open questions in biology such as the existence of different body types (morphologies) and effects of perturbation on natural ecosystems.

In this work, we implement a simple sensory model for virtual creatures. The model resembles vision or hearing systems of animals by locating a source object in the environment. We use a light source to denote the object, but the same system can and has been used to identify other creatures and objects. Our virtual creatures differ from other approaches (e.g. Miconi and Channon (2006)) in that they have arbitrary axes of movement and are not only constrained to movement in a plane. This difference complicates the learning task by increasing the amount of required sensory information.

Instead of evolving creatures with sensing abilities from scratch, we employ a tiered approach. Creatures, that we have already trained to walk, are infused with random visual networks and trained to follow the source object using a genetic algorithm. This approach gives us the ability to reuse previously evolved morphologies and neural controllers, and allows us to assess how difficult this problem is compared to evolving walking and sensing creatures from scratch. This idea stems from more realistic real-world robotic applications where morphologies and some behaviours (such as walking) can be provided via an engineered approach, and additional abilities (such as sensing) can be added on at a later time. We use our Creature Academy system introduced in Pilat and Jacob (2008) to streamline the evolution using a 2-tier training ground approach where each training ground runs a specific evolutionary training algorithm.

Background

The pioneering work on evolution of morphologies and controllers of virtual creatures was presented in Sims (1994a) and Sims (1994b). Sims used a genetic algorithm to co-evolve the morphologies and controllers of articulated virtual creatures made from 3D blocks. His graph-based generative encoding produced modularity and symmetry in the morphologies. Neural network controllers were embedded into the creature modules. The results showed various behaviours for independently evaluated learning tasks, such as swimming, walking, jumping, and following a light source. This work provided key insights into the interplay between shape adaptability and physical constraints in biological life. Sims evolved light-following creatures from scratch by using the relative direction of the light source to the creature for their evaluation. His fitness function was based on the speed at which the creatures moved towards the light and was averaged over several trials using different light source locations.

Miconi and Channon (2006) have successfully evolved virtual creatures for the tasks of locomotion and the co-evolutionary task of box-grabbing. In contrast to Sims' work, their model used a tree-like genotype and standard McCulloch-Pitts neurons. External sensors provided the neural networks with information on the distance to the op-

ponent and distance to the box. The joint axes of virtual creatures were either along the x- or y-axis, which resulted in creatures with movement along the x-y plane.

Creature Evolution

Creature Academy

Creature Academy (Pilat and Jacob (2008)) is a tiered experimentation system for evolution of form and function. The system allows for quick creation of virtual worlds — represented as training grounds — that are used for experimenting with various evolutionary *approaches* (such as genetic algorithms or neural network learning algorithms), *scenarios* (such as genotype representations, fitness criteria, phenotypical models), and *environments* (such as landscapes, aquatics, gravity). The idea is analogous to athletes training for competitions by using different training tasks in specific environments in order to improve their skills. Figure 1 shows a schematic view of the Creature Academy system.

Training grounds of the Creature Academy can be structured into a tiered (hierarchical) system akin to rounds in a sport tournament (see Tiered in Fig. 1). Upon finishing an evolutionary run, training grounds can send their best performing creatures to a higher tier training ground where the creatures are further improved (via evolution, learning, etc). Once all training grounds have completed, the remaining best individuals can be thought of as having successfully passed all the previous training grounds.

Virtual Creature Model

We model the phenotype of a virtual creature as a rooted tree, so that tree-based genetic programming can be applied. Nodes of the tree describe body parts of the creature, whereas the links represent connections among the body parts. The morphological genotype, however, has the form of a directed graph with possible cycles and self loops. A genotype body node stores information used to create the corresponding body part. The body type (a cuboid/block shape) has specific dimensions (width, height, depth), a recursion limit (how many times the node will be used to create phenotypic body parts while in a recursive cycle), and a local neural network (part of the neural controller). Body parts are linked using hinge joints which provide each part with one degree of freedom with respect to its parent. The joint axes are formed from two arbitrary orthogonal vectors (initially random but modified through evolution) so that the movement of each body part is not constrained to the x-y plane. Each phenotypical joint is stored in the child node and denotes the connection of the child to its parent.

Each genotype graph link specifies the position of the contact point on the parent body, orientation of the child node relative to the parent, and a scale parameter specifying how the size of the child is scaled relative to the parent node. This can be used to generate increasingly smaller/larger body parts. Negative scale denotes a reflection about the main axis of the parent node and allows for the generation of symmetric arrangements.

The control of a virtual creature is accomplished via a recurrent neural network (NN) made up of neurons and neural links specifying the directed graph NN topology. Each body part contains a list of sensor, effector, and hidden neurons. The network is encoded into the genotype of the creature and is copied into the phenotypical body parts during development. Connections between neurons in different local networks are only allowed between neighbouring body parts. A global neural network node (not associated with a physical body) is used to facilitate communication between the local neural networks and provide a form of centralized control.

As in Pilat and Jacob (2008), we use simple McCulloch-Pitts hidden layer neurons with a hyperbolic tangent transfer function. The input degrees of hidden neurons are controlled by a user parameter. The return value of every neuron in the NN is in the range [-1,1] to standardize the output/input values of all neurons in the system. Each body part contains a joint angle sensor neuron and an effector neuron. The sensor neuron stores the values of the body's joint angle with respect to its axis. The effector neuron (which uses the hyperbolic tangent transfer function on its input) represents the output of the local neural network and is used to calculate the desired angular velocity along the hinge joint axis. Special sinusoidal periodic source neurons are used as wave generators and feed the network with a periodic signal.

Algorithm

We use a standard genetic algorithm on a population of creature genotypes with two selection procedures: tournament selection and ranked selection. The algorithm evaluates a specified number of creatures at a time (9 for this work). This form of a parallel GA evaluates several creatures at roughly the same time as evaluating one creature (due to the nature of the ODE (2007) physics engine). The tournament selection algorithm concurrently runs 3 tournaments of 3 creatures, whereas the ranking selection algorithm evaluates 9 creatures from the population at a time.

Our evolutionary algorithm was implemented as a state machine with most steps equating to an interaction step of the physics engine. Evaluation of each creature was performed in steps, with each step consisting of: a pass through the neural network, actuation of the joints using the values computed by the neural network, and stepping of the physics engine. The integrity of each simulated creature was constantly checked using joint separation errors to prevent "joint explosions" by disabling invalid creatures. The system supports physical interaction. However, for the presented work, we have disabled creature interactions and only allowed collisions with the ground.

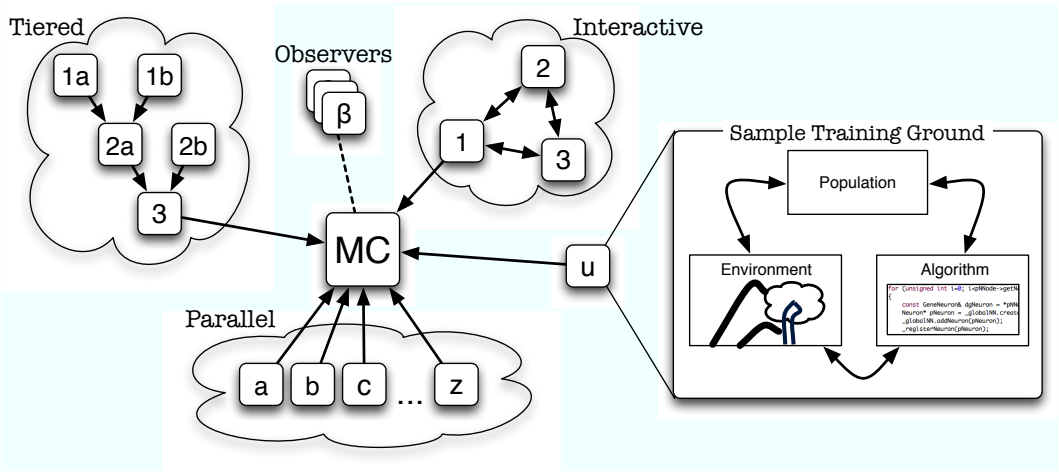

Figure 1: Schematic view of the Creature Academy system. Training grounds are shown as boxes with arrows showing the flow of creatures between training grounds. The Morphocosm (MC) creature ecology is fed with creatures from different evolutionary training scenarios: tiered, interactive, and parallel. A set of observers (β) visualizes the MC via different methods. Internal details of a sample training ground are shown on the right.

Vision Training

Vision Infusion

In Pilat and Jacob (2008), we have used a 2-tier Creature Academy to successfully breed walking and jumping behaviours. The tier-1 training ground experiments (using a walking or jumping fitness function) resulted in an assortment of successful walking or jumping creatures. In this work, we also use a 2-tier Creature Academy with the first tier consisting of the experiments performed for Pilat and Jacob (2008). For each second tier experiment, we choose a random tier-1 creature, generate a whole population of identical copies of that creature, infuse each copy with a different sensory neural network, and train the resulting network via a genetic algorithm that modifies the neural network weights. A schematic diagram of the process is shown in Fig. 2.

The vision infusion is performed by the addition of one or more sensory neurons to the main body part of the creature. Randomly weighted neural links are then created between the new sensory neurons and the rest of the neural network (ensuring the validity of each neural link connection). This results in each creature of the population having the same morphology and neural controller topology but with different weights on the randomly generated neural links. Evolution is performed on this population of creatures by using a genetic algorithm with neural link mutation and/or topology preserving crossover. Since the evolutionary algorithm only modifies neural network link weights, it can be thought of as a form of NN training.

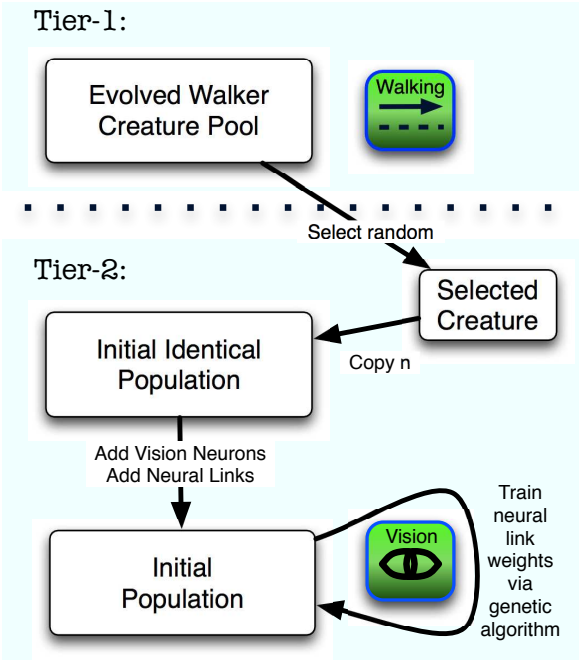

Figure 2: Schematic view of the Creature Academy setup used. Previously evolved walking creatures were randomly selected, multiplied, and infused with vision networks to fill an initial population for tier-2 experiments. The creatures were then trained on the light-following task using a genetic algorithm.

Vision Models

Several different sensing models can be implemented for a virtual creature. In this work, we implement a simple radius-

based source point detection (Fig. 3a). A sensing sphere of a fixed radius is created around the body part that holds the vision sensor. For each sensor, the simulation system registers all objects within its sensing radius. We have the ability to choose the type of object to register. For this work, we only consider a single light source. This sensing ability can be viewed as a simplified model of a 360 degree vision or single sensor hearing system in animals.

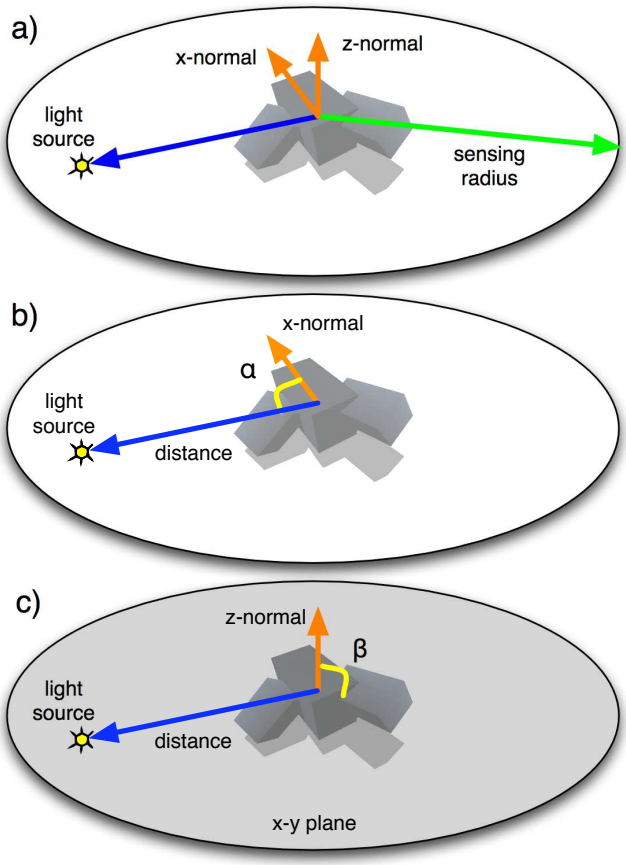

Figure 3: Vision model used in this work: a) shows all the components of the model, b) shows α angle calculation, and c) shows β angle calculation.

We have identified various methods of feeding the source object location into the neural network and have done experiments evaluating some of the different methods. A summary of our approaches is presented in Table 1. Source object locations can be represented via absolute or relative vectors, angles, signs, and distances. Multiple sensors can also be used for some calculations.

The most direct way of specifying positions to the neural network is to use raw vectors or vector differences. This method allows the neural network to process the vectors directly. However, this computation requires neurons that can calculate differences and angles. Our neural networks containing McCulloch-Pitts neurons are not powerful enough

Table 1: Various methods to feed the neural network with the location of a sensing source: $\vec{s}$ = source location, $\vec{c}$ = center of block, $\vec{n}$ = x-normal of block, $\vec{a}, \vec{b}$ = vectors from sensors to source location, $\vec{z}$ = z-normal of block, $\vec{zproj}$ = x-y plane z-normal projection.

Method	Pros/Cons
Direct Vector ($\vec{v} = \vec{s} - \vec{c}$) & Distance ($d = \lvert\vec{v}\rvert$)	Pros: lowest computing cost Cons: orientation of body not considered
x-normal Vector ($\vec{v} = \vec{s} - \vec{n}$) & Distance ($d = \lvert\vec{v}\rvert$)	Pros: lowest computing cost Cons: requires heavy neuron computation
Two Sensors ($d = \lvert\vec{a} + \vec{b}\rvert/2$) ($sign = \lvert\vec{a}\rvert - \lvert\vec{b}\rvert$)	Pros: easy to compute Cons: requires two sensors, 0° equivalent to 360°
x-normal Signed Angle ($\alpha = \vec{s} \cdot \vec{n}$) ($s = (\vec{s} \times \vec{n}).z$) & Distance ($d = \lvert\vec{s} - \vec{n}\rvert$)	Pros: provides a lot of information Cons: more difficult to compute
x-normal Signed Distance ($s = (\vec{s} \times \vec{n}).z$) ($d = s * \lvert\vec{s} - \vec{c}\rvert$)	Pros: uses only one scalar, works well for plane motion Cons: trouble with rotating morphologies
x-normal Signed Distance ($s = (\vec{s} \times \vec{n}).z$) ($d = s * \lvert\vec{s} - \vec{c}\rvert$) & z-normal Sign ($\hat{z}$)	Pros: uses only two scalars, works well for any morphology Cons: more computation required
x-normal Signed Distance ($s = (\vec{s} \times \vec{n}).z$) ($d = s * \lvert\vec{s} - \vec{c}\rvert$) & z-normal Angle ($\beta = \hat{z}(\vec{z} \cdot \vec{zproj})$)	Pros: works very well for any morphology, used for most experiments Cons: most computation required

to compute such values easily. With the addition of more neuron types (such as in Sims (1994a)), this method should provide the most information to the NN.

In our work, we opted for a simpler approach of supplying the neural network with angles instead of vectors. An angle tells the NN how far the orientation of the body part is from the desired orientation. The sign of the angle also gives extra information about the location of the source object relative to the body frame of the part (e.g. left/right or up/down). A distance measure provides the neural network with information on how close the source object is from the current location of the body part. A combination of these values provides the neural network with enough information to help the virtual creature to establish how it needs to move and in which direction.

Out of the methods outlined in Table 1, we have chosen to use the x-normal angle α (or its sign), distance, and z-normal angle β (or its sign). A schematic diagram of these

measurements is provided in Fig. 3. The α angle is the angle between the normal along the local x-axis of the body part and the vector from the center of the body to the source location (Fig. 3b). The angle is calculated using projections of both vectors onto the x-y plane. The magnitude of this angle informs the NN how far the body frame is from pointing its x-axis directly to the light source. The sign of the angle denotes that the light source is either at the left or the right of the local x-axis.

The β angle provides the NN information about the body part orientation relative to the x-y plane. It is the angle between the normal along the local z-axis of the body part and the x-y plane (Fig. 3c). The sign of the angle denotes whether the z-normal points away or towards the x-y plane. Since our joint axes are not necessarily along the x-y plane, our virtual creatures sometimes move by rotating their body axes, thus they need this extra information to know their orientation with respect to the x-y plane. For creatures with movement along the x-y plane only, this angle is not necessary.

Results

In this section, we present results of our work in training walking virtual creatures in the light-following task. First, we take a look at how the position of the light in the evaluation environment affects the training. Second, we discuss experiments with different genetic algorithm selection methods and fitness functions. Following that, we present results with using different vision sensing neurons and how the choice depends on the original walking behaviour. Finally, we talk about the observed light-following behaviours and creature morphologies.

Light Placement

The placement of the light in the evaluation environment plays a role in how well the virtual creatures evolve light-finding and light-following controllers that work well with any light position. Placing the light at the same location for each evaluation adds an evolutionary bias, where creatures that happen to travel in that direction are considered fitter than those that do not - the evolved behaviour might not even have anything to do with the actual light. Thus, it is important to randomize the light location for each creature evaluation.

To give each creature a far enough light source to strive for, we place the light on the circumference of a circle of specified radius centered at the start location of creature evaluation. This radius is smaller than the range of the sensors so that the light source is seen during evaluation. For our experiments, we have used a light placing radius of 500 with a sensing radius of 1000 (see Fig. 3). This placement of light sources on the circumference of a circle can also add bias to the evaluation, since the light source is always the same distance from the start. We have not yet evaluated the impact of this bias.

We have outlined two ways of light placement along the circle circumference as outlined above: random and sequential. The random placement randomizes the position of the light source for every creature evaluation. This is accomplished by choosing a random angle and rotating a fixed radius vector by that angle. This method provides each evaluation with a different light position and is responsible for jumps in fitness as seen in our fitness graphs (due to non-uniform fitness criteria). The sequential method increments the placement angle by a small amount (in this case $\pi/6$) and yields a sequential placement of the light source around the circle. We have not observed a significant difference between the two methods.

A changing light source position makes it more difficult for the evolutionary algorithm to properly award individuals with partial solutions. Even if a neural network controller learns to partially steer towards the light, it can still be beaten by a random individual that happens to travel in the direction of the light source. To remediate this problem, we implemented multiple evaluations of individuals using a different light source placement for each evaluation. The fitness values out of each of the evaluations were then averaged to create the final fitness value of the individual. In some experimental runs, we noticed an improvement in fitness by using 3 and 5 averaged evaluations. However, the impact of multiple evaluations needs to be tested further.

GA Methods

In our experiments, we have used two different selection methods for our genetic algorithm: tournament and ranking. Most of our experiments were performed with tournament selection. We have not observed any significant difference in results when comparing the two selection methods. The fitness graphs of the two methods differ due to the full evaluation of all the individuals in the population in ranking selection versus the random evaluation of individuals in tournament selection. With tournament selection, the best and worst individuals are not normally evaluated at every tournament, so their fitness values stay constant until they are evaluated again or a better individual is found. The best and worst fitness plots of ranking selection experiments have varying values for each generation.

We have used two fitness criteria to evaluate the creatures in our experiments: distance and speed. The distance to light source was computed as the difference between the distance from the start position to the light source and the distance from the end position to the light source. Due to our light source placement, the maximum attainable distance fitness was 500 and there was no bound on the lowest fitness. Distance fitness around 0 signified that the creature has not moved far from its initial position during the evaluation.

The speed was calculated as the difference between the

current and last distance to the light source at each time step (i.e. the speed of the creature along a line to the light source). The maximum and minimum distances depend on how quickly the creature moves to the light source. A positive fitness means moving closer to the light source and negative fitness means moving farther away. This form of fitness calculation awards the creatures that don't move directly towards the light source more than the final distance method discussed above.

We have run identical experiments using the two different fitness criteria so that we can compare how they impact on the evolution of light following behaviours. However, apart from the different scales of the fitness graphs, we did not see a noticeable difference in either the fitness graph structure or the evolved behaviours.

Most of our experiments were run using only a mutation genetic operator which modified the neural network weights. This method can be thought of as NN training using a genetic algorithm. Mutation was always applied to the child of the selected individual to produce a mutated offspring for the next generation. Only mutation of neural weights was performed at a rate of 10% per neural link by adding a Gaussian distributed random value. We have also run experiments with a topology preserving crossover; however, we did not notice improved performance using crossover and mutation compared to only mutation. We feel that changes induced by the crossover operator were too large to help the evolution. More experiments with optimized genetic operator values are needed to validate this hypothesis.

Sensing Neurons

Table 1 compares the different ways of feeding single-source sensing information into a creature neural network. In our experiments, we have used three vision sensory neurons, the input of which are: the x-normal angle α, the distance to light source, and the z-normal angle β, respectively. The output of these neurons was scaled to the $[-1, 1]$ range. Distance was represented as a $\log$ scaled distance from the body part to the light source using the function $\log(d+1)/\log(sensingrange)$ which produced a value in the range $[0, 1]$. The distance value was then combined with the sign of the α angle to form the α signed distance measure in the range $[-1, 1]$ such that left/right orientation and angle can be stored as one scalar value.

Original experiments were run without the β angle (only using the α angle and distance, or using the α signed distance alone). We have noticed better results with α signed distance value compared to using both the α angle and distance. This might have had to do with the fact that this method used a smaller neural network with no α angle neuron and its links and thus was easier to be evolved using the genetic algorithm. We feel that the sign of the α angle is enough for light source detection and that the angle itself is more of an optimization feature that can be used to move or rotate faster when the angle is larger.

The experiments without the β angle produced very few good light following behaviours (about 10% of the runs). The evolved behaviours were either on creature morphologies that kept fixed axes with respect to the x-y plane (i.e. no rotational motion) or on spinning morphologies where the neural network evolved to fix the motion along the x-y plane. This led us to believe that not enough information is provided to the neural network for spinning morphologies. To alleviate this problem, we introduced the β angle which tells the neural network the orientation of the body part relative to the x-y plane. Successive experiments showed a large increase in proper evolved light-following behaviours (Fig. 4a versus 4b, c, d).

We experimented with different sets of vision sensing neurons to identify the neural setup that produces the best behaviours. The β angle was shown to be necessary in all setups. The α signed distance proved once again to provide better results than using both the distance measure and α angle (Fig. 4b versus 4c). This is probably due to the additional complexity of the neural network using the x-normal angle neuron. Distance measure seemed to be an important part of the neural network and experimental runs without the distance neurons did not perform as well (Fig. 4c versus 4d). This might be because the distance neuron feeds the neural network with a slowly changing value that can promote more movement from the effectors thus creating more exploratory behaviours.

Morphologies and Behaviours

Three classes of interesting light following behaviours were observed. The first class consisted of the creatures with non-spinning movement constrained to the x-y plane (Fig. 5e-h). These creatures very quickly evolved light following behaviours even without the need for the β angle neurons. Some of the fixed axes morphologies first exhibited an in-place turning behaviour until they were pointing at the light and then moved towards the light. Others, moved towards the light at an angle, at times fixing their orientation before they would continue towards the light.

The second class included creatures that initially had rotating body axes usually with a spinning motion. Through the evolutionary algorithm they have adapted their neural network to movement that is mostly constrained to the x-y plane (Fig. 5a-c). This allowed them to use the α angle and distance measures to locate the light source. The creatures in this class evolved mostly in experiments with no β angle neurons. The third class consisted of creatures that had kept their rotational (spinning) motion but still managed to evolve some kind of light-following behaviours (Fig. 5d). These creatures were often quite slow at reaching the light and would at times back away from it.

Figure 5 shows some morphologies that evolved light-following behaviours. Most runs evolved some form of

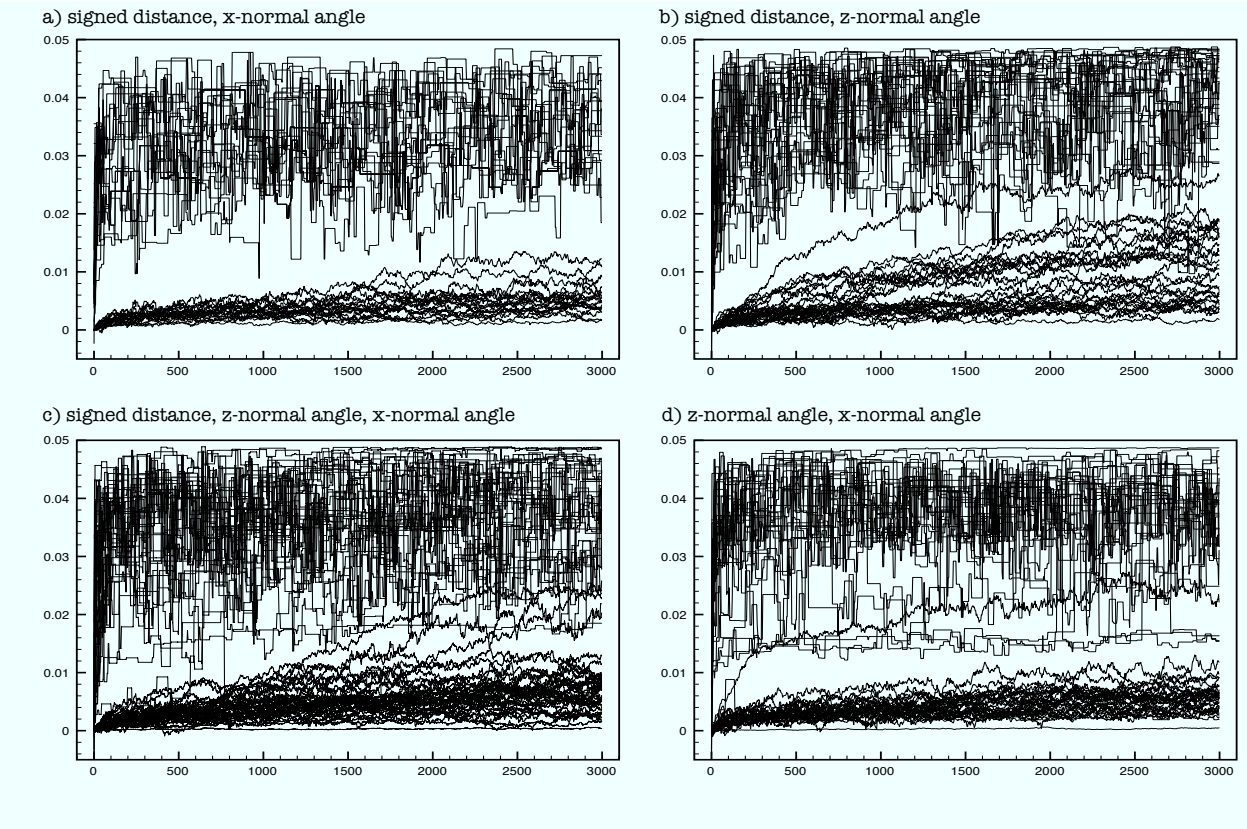

Figure 4: Fitness plots showing current maximum population fitness (top curves) and population average fitness (bottom curves) using the light following speed fitness function. The plots demonstrate different performance using different sets of vision sensory neurons.

light-detection behaviour, usually linked with turning towards the light. Some of the evolved motions were perfect until the creatures changed their orientation through either falling due to the physics or their own actuation. Quite often, the creatures were then able to either fix their orientation or continue to move with a different orientation. At other times, the fall rendered them unmovable. Many of the behaviours observed were quite stable and not prone to sudden changes of orientation.

Conclusion

In this paper, we have presented a method of adding perception abilities to previously evolved virtual creatures that had previously learnt to move. A "sensory brain" was added to these creatures to evolve the additional capability to move towards a light source. We have used a mutation operated genetic algorithm to train the augmented neural network in the light-following task. We have also discussed different ways of feeding vision information into the neural network and how this affects the creatures' ability to detect the light and move in its direction.

We were able to evolve several interesting light-following behaviours. Creatures with fixed motion along the x-y plane quickly evolved good light-following with minimum sensory input requirements. Other creatures with spinning orientations required extra sensory information — such as an orientation angle relative to the x-y plane — to come up with good behaviours, or adapted their neural networks to contrain movement along the x-y plane only. Many of the resulting behaviours were quite stable and self-repairing upon changes in creature orientation.

This work demonstrates that it is possible and feasible to use a simple evolutionary algorithm to evolve additional behaviours for previously evolved creature morphologies and neural network controllers. It is also possible to evolve orientation-repairing behaviours. These observations are important for training engineered robotic bodies, especially in environments that can unexpectedly change the orientations of the robotic forms or even change the morphologies themselves.

As future work, we will take a closer look at the evolved neural controllers to find out what neural circuitry is responsible for orientation-repairing capabilities. Experimentation with varied physical landscapes can produce more adaptable

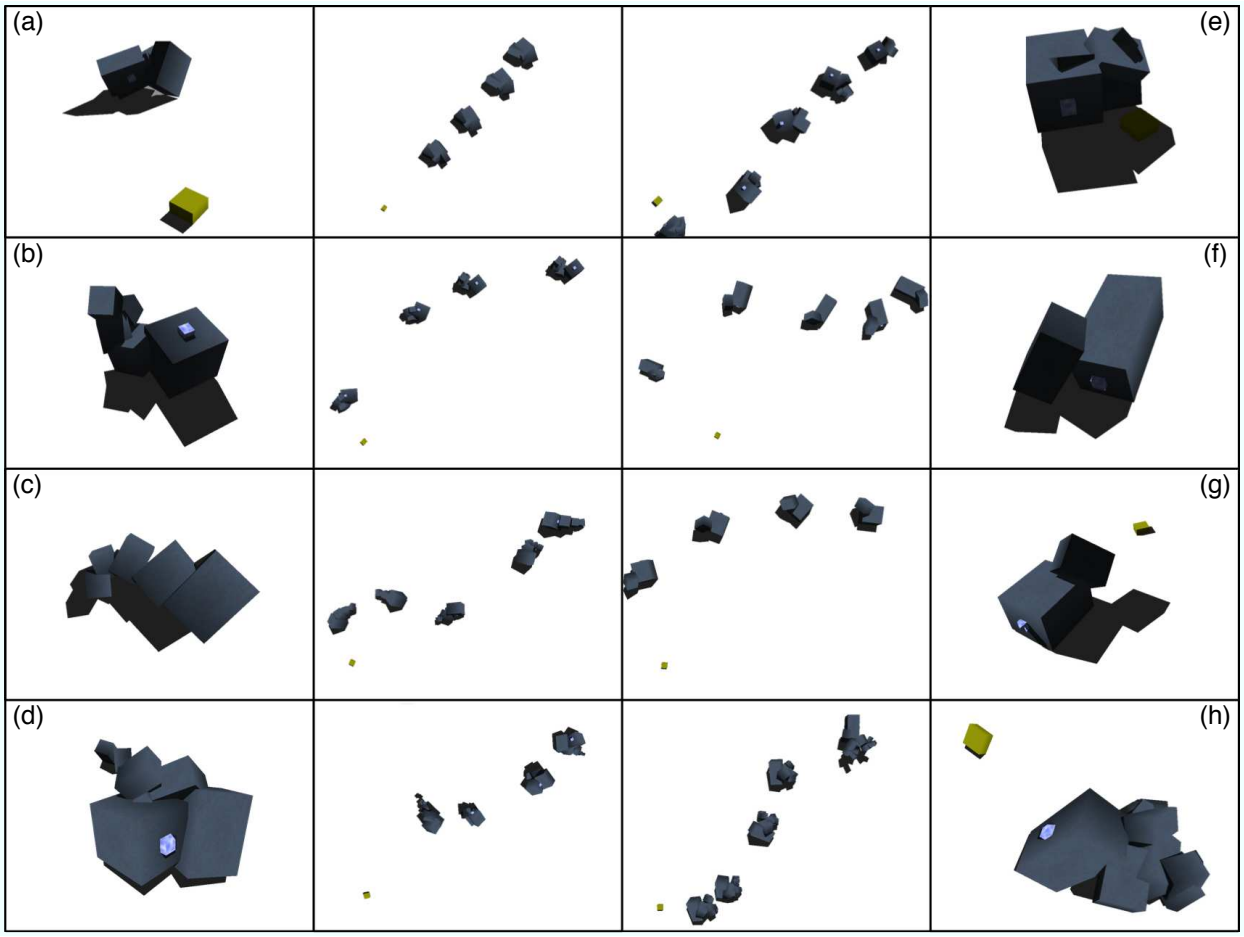

Figure 5: Sample creature morphologies that evolved light-following behaviours. A sequence of overlaid images show the light-following behaviour. The larger image shows a more detailed view of the morphology. Creatures a-c have evolved to constrain their movement to the x-y plane. Creature d has evolved a spinning strategy. Finally, creatures e-h have started off with already plane constrained movement.

neural networks. Finally, we would like to use the resulting creatures in co-operative and ecosystem scenarios, where the vision functionality is necessary to avoid environmental obstacles, to find resources, and to find other creatures.

The open-source Creature Academy system is written in platform-independent C++. It makes use of other open source projects such as OGRE (2007) for graphical output and ODE (2007) for physical simulation and realism. All the experiments in this study have been performed on a cluster of 50 Apple Mac minis, each with a 1.66Ghz Intel Core Duo processor and 1GB of RAM. More information about the Creature Academy and Morphocosm project, including images, videos, and experimental data, can be found at our project website: http://www.morphids.com.

References

Miconi, T. and Channon, A. (2006). An improved system for artificial creatures evolution. In L.M. Rocha, e. a., editor, *ALIFE X: Proceedings of the 10th Conference on the Simulation and Synthesis of Living Systems*. MIT Press.

ODE (2007). Open Dynamics Engine.

OGRE (2007). Object-Oriented Graphics Rendering Engine.

Pilat, M. L. and Jacob, C. (2008). Creature academy: A system for virtual creature evolution. In *CEC2008: Proceedings of the IEEE Congress on Evolutionary Computation*. in press.

Sims, K. (1994a). Evolving 3d morphology and behavior by competition. *Artificial Life IV: Proceedings of the Fourth International Workshop on the Synthesis and Simulation of Living Systems*, 1(4):353–372.

Sims, K. (1994b). Evolving virtual creatures. In *SIGGRAPH 94: Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, pages 15–22.

Wake, M. H. (2001). Bodies and body plans, and how they came to be. In Kress, W. J. and Barrett, G. W., editors, *A New Century of Biology*, pages 29–52. Smithsonian Books.