Evolving Flying Creatures with Path Following Behaviors

Yoon Sik Shim[1], Sun Jeong Kim[1], and Chang Hun Kim[1]
[1]Korea University, Seoul, Republic of Korea
necromax@kebi.com

Abstract

We present a system which evolves physically simulated 3D flying creatures and their maneuvers. The creature is modelled as a number of articulated cylinders connected by triangular patagia in between. A creature's wing structure and its low-level controllers for straight flight are generated by an evolutionary algorithm. Then a feed-forward neural network is attached to the low-level controllers, and the connection weights of the network for a given trajectory are found by a genetic algorithm. We show that a control system sufficiently effective to allow aerial creatures to follow a complicated path can be achieved by two-step evolution process.

Introduction

To date, artificial evolution researchers have found only a few applications for virtual embodied agents. Sims (Sims, 1994) pioneered an innovative approach in generating both the morphology and the behavior of physics-based articulated figures by artificial evolution. His "blockies" creatures moved in a way that looked both organic and elegant. Other recent work such as Sexual Swimers (Ventrella, 1998), Framsticks (Komosinski and Ulatowski, 1999), and Virtual Pets (Ray, 2000), have produced various applications in their own way. The Golem project (Lipson and Pollack, 2000) even took this technique a step further by evolving simulations of physically realistic mobile robots and then building real-world copies.

Relatively little work has addressed the simulation of flying behavior. The interaction between a flapping wing and the air is very complicated. The forces generated by this interaction are chaotic and their simulation is often unstable because of high sensitivity. Computational fluid dynamics (CFD) methods have sufficient accuracy, but they are too time-consuming to build into an evolution framework. Reynolds (Reynolds, 1987) developed a stochastic model of the flocking behavior of birds, although he did not animate individual bird animations. Ramakrishnananda and Wong (Ramakrishnananda and Wong, 1999) presented a physically based bird model using two segmented airfoils but the entire motion of both wings is controlled by predesigned functions and only overall lift and thrust were calculated from aerodynamics. Wu and Popović (Wu and Popović, 2003) developed a realistic model of a bird with flexible feathers and successfully generated flying motions corresponding to a given trajectory by optimizing the control parameters for each wingbeat using simulated annealing.

In terms of virtual creatures, our prior work (Shim and Kim, 2003) has proposed a new developmental model for evolving wing morphology and behavior with a traditional nested graph structure, but it was difficult to generate higher-level behaviors, such as following targets, because of the instability of the air surroundings and unnecessarily complex controller networks.

We now present a two-step evolution process for evolving path-following flying creatures. We describe an improved evolutionary design for creature generation and simplified aerodynamics for use in the time-consuming evolution process and to make convenient of genetic encoding. Since the activity of flying is very sensitive to the characteristics of the environment(low density of fluid, hard to maintain balance against the gravity), we use a more robust periodic controller for motor control. Since the new evolved wing structure has many more degrees of freedom of joint than the predesigned wings used in previous work, searching through all the control parameters at each wingbeat is too computationally expensive and time-consuming process. We therefore exploit a continuous map of the neural network structure so that the optimization is only required once for an entire simulation.

Overview

The entire process is divided into two stages. In the first stage, our creature evolution system generates the creature's body and its low-level controllers that support straight forward flight. After this basic forward-flying creature has been generated, a neural network is connected to its controllers, and the connection weights of the network are optimized by a genetic algorithm to make the creature to follow given trajectory. During the optimization, a set of arrays which contains the weight vectors for the neural network is used as the population. At each time step, the network receives the position and velocity of the creature's central body, the feedback

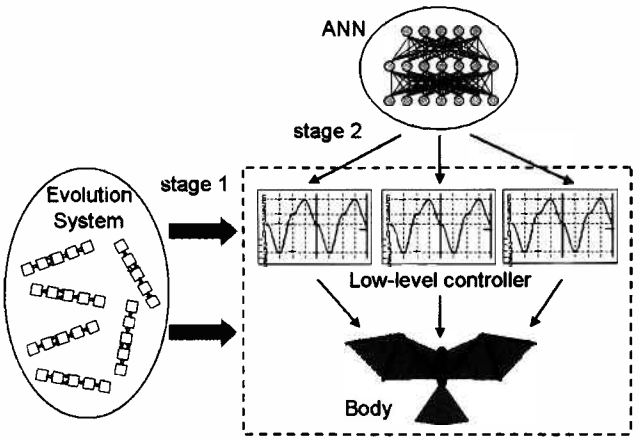

Figure 1: Overview of the entire system. The evolution system creates straight-flying creature, and a neural network controls the creature so that it will follow a given trajectory.

of the neural network output, and the target position which is calculated in realtime based on the creature's velocity and the given trajectory. The signals coming out of the network are fed to the creature's controllers, and vary their behavior so that the motion of the wings is changed.

The concept can be explained at an intuitive level by comparing it to three elements of a real life form: the body, the motor nerves of the cerebellum (evolved low-level controllers) which is in charge of motor control, and the cerebrum (a neural network) which gives high-level control to the cerebellum. In the next sections, we describe each process in more detail.

Genotype Representation and Creature Development

The creature is constructed from a basic model which has truss-shaped wings (Shim and Kim, 2003). The genotype encoding is accomplished using a multi-connection list. This allows complete symmetry of shape and wing motion. In this form, variations of the genotype, such as mutation or

Figure 2: A genotype structure. The wing-root has double connections for two symmetric wings.

mating, because of its linear structure. The Root represents the central body (fuselage) of a creature and the wing-root represents the root of wings, which lies inside the central body. These first two nodes and their connections exist in all individuals by default; but they are followed by an arbitrary number of nodes for wing segments. Only the wing-segment nodes are deleted or added during genetic variation.

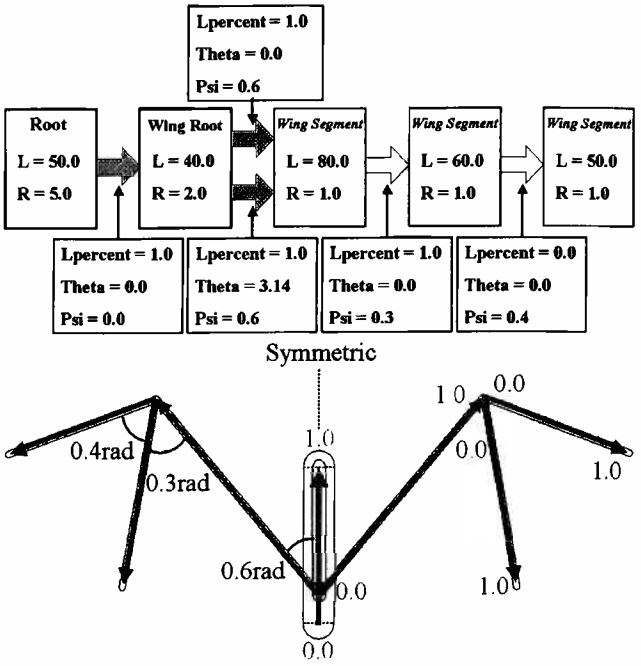

Figure 3: An example of a genotype and the resultant skeletal structure. In the lower picture, the parameters of the left wing and right wing describe **Psi** and **Lpercent** respectively.

Each wing cylinder has a joint which connects it to its parent. The wing joint can have two degrees of freedom, allowing both dihedral and sweep motions. The corresponding geometric primitive is a capped cylinder which is aligned along the local z-axis. Each cylinder's two end points (the centers of each hemi-spherical end) are known as the start and the end point. The start point is defined as position 0 on the cylinder and the end point corresponds to limit parametric distance along the cylinder's local z-axis. A node connection specifies how and where to attach each cylinder to the parent. **Lpercent** indicates the position on a parent cylinder where the child's 0 point will be placed, and the value is set to 0 or 1 to obtain a truss-like wing structure. **Theta** (θ) is an initial sweep angle rotated about the parent's local z-axis. The wing grows in the same direction with making an acute angle of **Psi** (Ψ) which is the angle between the local z-axes of parent and child. **Psi** and **Theta** are set as neutral values of each joint at the time of initialization and varies over time during the simulation. After two cylinders have been attached to each other, a zero mass rigid film is combined between the two cylinders. Since this film is attached to a child cylinder, its linear and angular velocities can be calculated from that of the child cylinder. The whole of a creature's wing is thus composed of a sequence of child-parent pairs with attached films, and it can be considered as a webbed structure like a kite or a hang-glider, and each film is subject to the appropriate aerodynamic forces. The model has a fan-shaped tail which is attached to the rear of the cen-

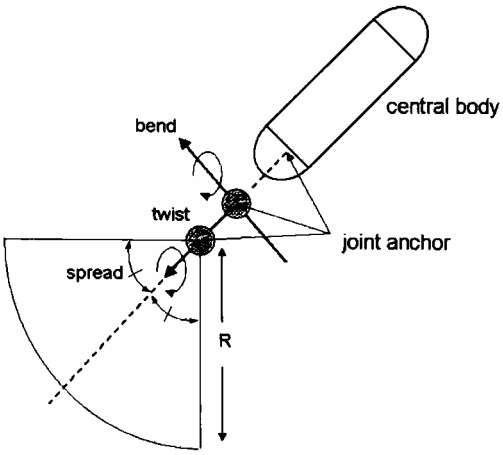

Figure 4: The tail.

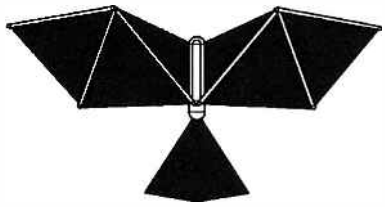

Figure 5: An example of a completed creature.

tral body cylinder. The tail has three degrees of freedom which are: bend, twist, and spread. The tail plays an important role in balancing and in turning movements. The tail radius is also included as a parameter of the genotype.

Controller parameters

The controller is defined by a combination of piecewise sinusoidal functions. The output value of the controller is a desired value of the proportional-derivative controller which is used for calculating the actual motor torque. The controller parameter includes the duration of a cycle and the number of control points. Each control point can have arbitrary values between -1 and 1, but all controllers operate with a cycle of the same duration. Hence, all joint motions can be controlled synchronously. We can write the controller function U as:

$$U(t) = \frac{p_n + p_{n+1}}{2} + \frac{p_n - p_{n+1}}{2} cos(\frac{Nt}{D} - n)\pi \qquad (1)$$

$$n = 0,1,2,...,N-1 \qquad p_N = p_0 \qquad 0 \le t \le D,$$

where N is the number of control points, and d is duration of a cycle. The control function is constructed by smooth interpolation between each pair of neighboring control points. The last control point is connected back to the first one, which makes the function to be periodic. These parameters are also coded into each node of the genotype. Using a

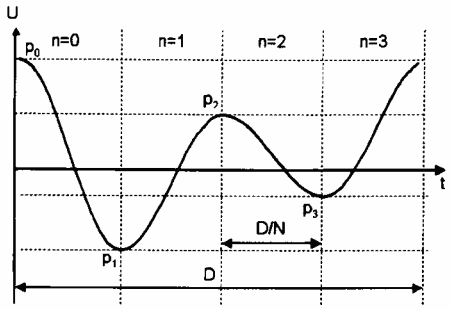

Figure 6: Low-level controller.

periodic sinusoidal function has two advantages. First, most locomotion (including flapping flight) of real animals can be expressed using a sinusoidal function. Second, the sinusoidal function has an infinite derivative continuity so that unexpected fluctuations of a joint motor can be minimized.

Dynamics Simulation

The movement of a physical creature is calculated by an articulated rigid body dynamics. Resistance forces on patagia are calculated by simplified aerodynamics, with gravity. We used the Open Dynamics Engine by Russell Smith (Smith, 1998) which is an open-source articulated body simulator. The integrator is a first order semi-implicit integrator (Stewart and Trinkle, 1996), where the constraint forces are implicit, and the external forces are explicit. A medium speed ($O(n^3)$, where n is the number of links) Lagrange multiplier method (Anitescu and Potra, 1997) is used to calculate the articulated body dynamics.

Force on the Triangular Film

The forces acting on a surface depend on its area and the angle of attack with respect to the velocity of the airstream. A film is divided into several patches, and the calculated forces are considered to be exerted on the center of mass of each patch. The stream velocity for patch i is simply a negation of the velocity of the patch's center of mass. The

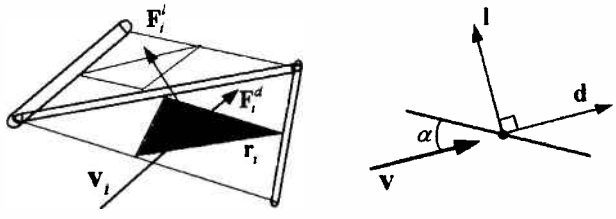

Figure 7: Force on a triangular patch.

lift and drag forces are due to the lift coefficient C_L and the drag coefficient C_D which are functions of the angle of attack α. Drag acts in the direction of the airstream velocity. The direction of lift is perpendicular to that of the drag, so

it produces not only lift but also thrust tangential to a patch during the flapping motion of the wings. Each force on the patch i can be simply written as:

$$\mathbf{F}_L^i = \frac{1}{2}\rho C_L(\alpha) A_i \|\mathbf{v}_i\|^2 \mathbf{l}_i \quad (2)$$

$$\mathbf{F}_D^i = \frac{1}{2}\rho C_D(\alpha) A_i \|\mathbf{v}_i\|^2 \mathbf{d}_i, \quad (3)$$

where A is area of the patch, and ρ is the density of the air (Fox and McDonald, 1976). We synthesized functions to produce lift and drag coefficients at any particular angle based on the typical plots for a real bird (Withers, 1981; Mueller, 2001). After the forces on each patch have been calculated, the resulting force and the torque on the skeleton are obtained by:

$$\mathbf{F}_{total} = \sum_{i \in N} \mathbf{F}_L^i + \mathbf{F}_D^i \quad (4)$$

$$\mathbf{T}_{total} = \sum_{i \in N} (\mathbf{F}_L^i + \mathbf{F}_D^i) \times \mathbf{r}_i \quad (5)$$

where N is the set of triangular segments covered by film and the $\mathbf{r}$ is the relative vector from the cylinder's center of mass to the center of the each segment. The forces on the tail can

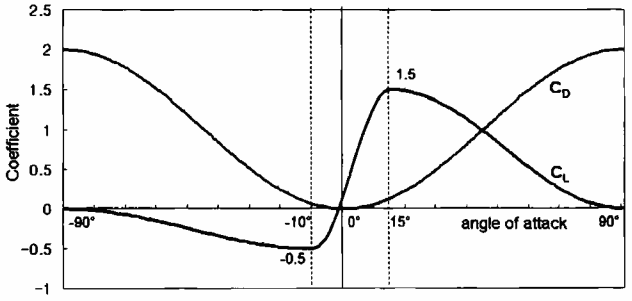

Figure 8: Lift and drag coefficients.

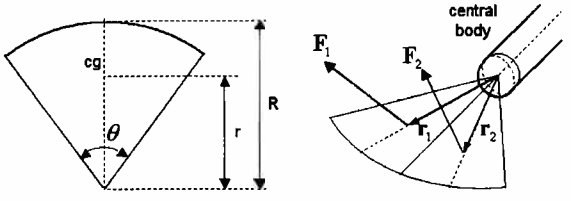

Figure 9: Forces on the tail

be calculated in the same way as those on a patch. Since the tail is fan-shaped, it is simply divided into two sub-fans to accommodate tail twist. The center of mass of the tail can be calculated using the length from a fan's vertex to the center of mass.

$$\|\mathbf{r}\| = \frac{4R\sin\frac{\theta}{2}}{3\theta} \quad (6)$$

R is the tail radius, and θ is the angle of a fan.

Force on the Cylinder

The forces on a central body cylinder are also calculated using existing method (Wejchert and Haumann, 1991). Instead of calculating the forces on every surface of the cylinder, the total resistance is divided into two terms that act at its center of mass: these are linear resistant force and resistant torque. In case of the linear resistance, the area that receives force

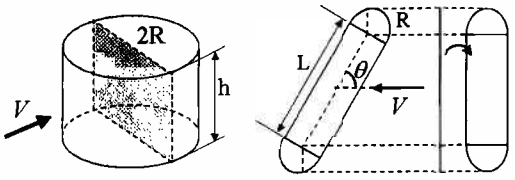

Figure 10: Idealized linear resistance of a cylinder.

is idealized as the projection of the capped cylinder on to a surface normal to the velocity vector. Therefore, the linear resistance of the cylinder can be written as:

$$\mathbf{F} = \alpha_n(2RL\sin\theta + \pi R^2)v\mathbf{v}. \quad (7)$$

The resistant torque is calculated by dividing a cylinder into slices and integrating each over all cylinder slices. For convenience, we approximated a capped-cylinder by a standard cylinder with the length of L+2R.

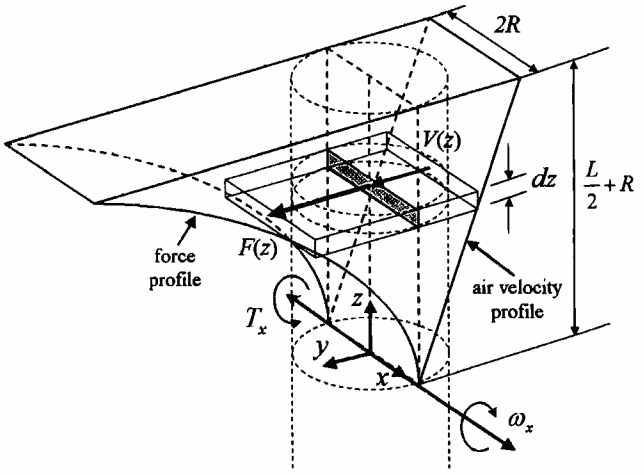

Figure 11: The angular resistance of a cylinder.

$$T_x = -4\alpha_n\omega_x^2 R\int_0^{\frac{L}{2}+R} z^3 dz \quad (8)$$

$$T_y = -4\alpha_n\omega_y^2 R\int_0^{\frac{L}{2}+R} z^3 dz \quad (9)$$

$$T_z = -2\alpha_n\omega_z^2(L+2R)\int_0^{R} r^3 dr \quad (10)$$

The angular velocities and torques described above are measured about the local axis of each cylinder.

Evolution Process

Each creature is initially placed at a given altitude, and then the simulation is started. The fitness f is defined by the sum of the hovering time (t_h), the flight speed (in the direction of $\mathbf{d}_f$), with the addition of a penalty for unnecessary shaking of the central body.

$$f = k_a t_h + \int_\zeta \{k_b(\mathbf{v}(t)\cdot\mathbf{d}_f) - k_c\|\omega(t)\|\}dt \qquad (11)$$

The terms $\mathbf{v}(t)$ and $\omega(t)$ are the linear and angular velocities of the central body respectively. The symbol ζ denotes the simulation period and k is the weight constants for each term. In the early stage of evolution, hovering time is crucial for survival, because most creatures fall to the ground before the end of the simulation period. The simulation speed is improved by prematurely aborting the simulation of any individual that is not performing well. A creature is also considered to blow-up if the speed of its body parts exceeds some value (typically $1000m/s$), and that creature is aborted. Additionally, interim fitness measuring is performed after a quarter of the total evaluation period has elapsed by removing individual whose fitness is less than one fifth of that of the least fit creature of the previous generation. After fitness evaluation, a typical genetic operation (mutation, crossover, grafting) (Sims, 1994) is performed to produce the next generation.

Combining Controllers with Neural Network

The creature control is done by giving variation to the evolved controller of a creature using a neural network. Since we do not have the training samples necessary for path following control, nor the gradient information of numerical physics simulator needed for the gradient-based training methods such as backpropagation, the network weight is found by a genetic algorithm.

Neural Network Parameters

Two control values are assigned to a single controller which are: scaling s and phase transformation p. Since the tail has no controller, each degree of freedom of the tail is directly controlled by the network (θ^B, θ^T, θ^S). The duration of a cycle D is the same for all the controllers. The scaling signal is simply multiplied to the controller output, and a phase transformation performs time warping (we use t' instead of t, see figure 14) in a way similar to (Wu and Popović, 2003). At the start of each cycle, new values of scaling, phase, and duration are obtained by sampling the continuous output of the network, and these values are retained until the cycle ends. The output of the previous state becomes the input of the network, together with other external states of the creature such as the target position(T), the velocity of the central body(v), and the values from a simplified gyrosensor (G) (Shim and Kim, 2003). All vectors (T,v) are transformed to the local coordinates of the central body. The bipolar sigmoid which has a range of -1 to 1 is used as the activation function of a neuron.

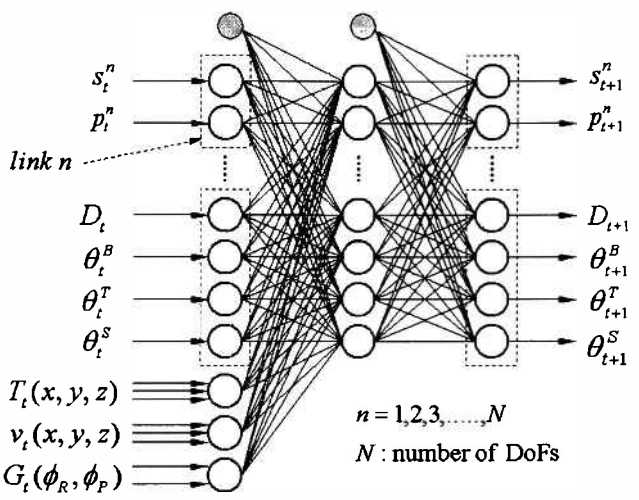

Figure 12: Input and output signals of a neural network.

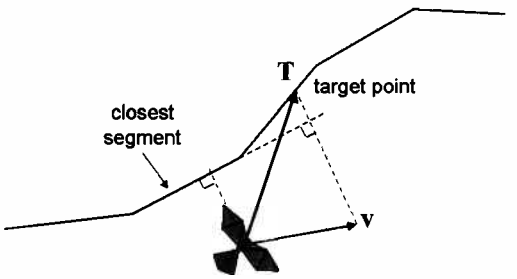

Figure 13: Obtaining the target position on the trajectory.

The path trajectory is given as a piecewise linear curve. At each simulation time, the desired target point is obtained as a function of the current velocity of the central body similar to a heading-based approach (Ribals et al., 1993).

Learning for Given Trajectory

The length of a chromosome for the genetic algorithm is H×(4N+17), including connection weights from the bias term, where H is number of hidden nodes and N is the total number of degrees of freedom in all joints. We use 1 to 2 times as many hidden nodes as the number of input nodes. The fitness is given by

$$f = \int_\zeta \{k_1 d_{min}(t) + k_2 v_t(t) + k_3 a_T(t)\}dt, \qquad (12)$$

where d_{min} is the minimum distance from the trajectory, v_t is the speed of the creature in the current path direction, and a_T is the acceleration towards the target point. Lower values represent better fitness. Each time the fitness evaluation

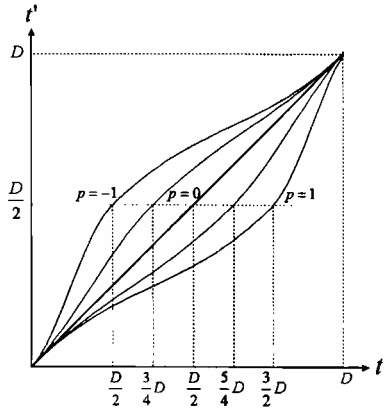

Figure 14: Time warping for phase transformation.

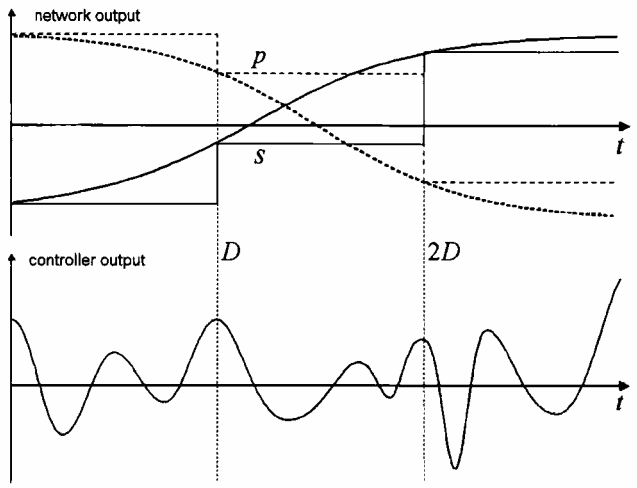

Figure 15: Neural network output(upper) and resultant controller signal(lower).

is complete, the population is divided into three groups according to fitness, giving a strong penalty to last two groups which show unexpected behaviors. The first such behavior is a blow-up of the simulation, and the other occurs when the heading direction of central body deviates seriously from the current path direction ($h \cdot d_t \leq 0$, where h is the heading direction and d_t is desired direction). A creature which does not commit any of these misdemeanor (called a survivor), is placed in the first group of the population. Heading failures go into the second group, and blow-ups go into the worst group. In those two worse cases, the simulation is aborted and the fitness is assigned by following metric:

$$f = P, Q + \frac{1}{t_a}\int_0^{t_a} \{f(t)\}dt. \qquad (13)$$

P and Q are sufficiently huge numbers to distinguish each group, and t_a is the time when the simulation is aborted. The second term indicates the mean fitness over the simulated period. We check each criterion only in some time interval ($0 \leq t \leq t_c$), which we call the "checktime". That is because, if we check these criteria in all simulation periods, few creatures would be survived in the first group(especially most of the creatures will commit the heading failure). The checktime is initially set to some fraction (α) of the full duration, and increases gradually as a function of the number of survivors in the first group

$$t_c^{g+1} = \zeta\{(1-\alpha)\frac{S^g}{P} + \alpha\}, \qquad (14)$$

where g is the generation, S is the number of survivors, and P is the population size.

Results

Each creature was placed in the air at a height of $1000m$ from the ground, and the simulation was started at an initial push of $50m/s$. Typical settings for our runs were as follows. In the first stage (evolving simple forward flight creatures), the population size was generally 200, and runs lasted for 100 to 200 generations. The top 20% of genotypes were used as elites, and the remaining 80% of the new generation were created by selecting a couple of parents by roulette wheel selection, and copying one of them (choosen by tournament selection with a 90% probability of selecting the fitter) with probability 30%. The rest of the offsprings are made up of 30% crossovers, 30% grafting, and 10% random individuals. Mutations were then applied stochastically to the newly generated genotypes. The integration step was generally 0.05 seconds, and the evaluation period was around 500 simulation steps.

In the network evolution stage, the population size was 200-300, and the number of generations was 300-500. The simulation period was set to 1000-2000 time steps, depending on the speed of a creature in straight flight. The setting for the genetic algorithm were the same as those for creature evolution, but the number of crossover points was in the range of 1 to 4. The running time was generally 4-6 hours for creature evolution and 6-10 hours for path-following, running on a single Pentium-IV 2.53Ghz PC.

As expected, most of the creatures fell to the ground during the simulation in the early generations. A creature capable of staying airborne for longer than 500 time steps did not appear until the first quarter of the total running period had almost elapsed. This suggests that air is a harsh environment for evolved creatures, compared with water or land, especially in terms of maintaining balance. Since the creatures have to retain balance against the disturbance of gravity through their life time, this seems to limit the range of motion patterns that evolve. In terms of path-following behaviors, there were some local optima such as hovering near the trajectory with low velocity, or continuing to move straight ahead when the variation of path direction is small.

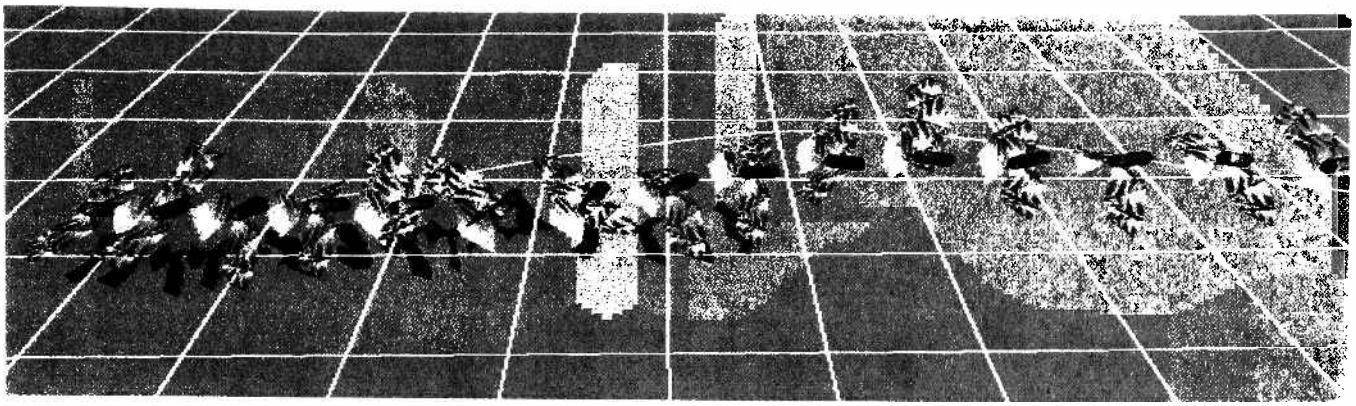

Figure 18: Sequential image of flying creature.(sampled at each 30 time steps)

Parameter	Value or Range
number of wing segment	4∼10
cylinder length	0.2∼0.8m
cylinder Radius	wings : 0.05m central body : 0.1∼0.4m
ψ	initially : 10° ∼ 70° during simulation : ± initial ψ
θ	initially : ± 0° during simulation : ± 45°
controller duration	0.2∼0.6 sec
density of cylinder	1000kg/m^3
ρ	1.21kg/m^3
k_a, k_b, k_c	20.0, 1.0, 2.0
k_1, k_2, k_3	1.0, 3.0, 2.0

Table 1: Some values used in the simulations.

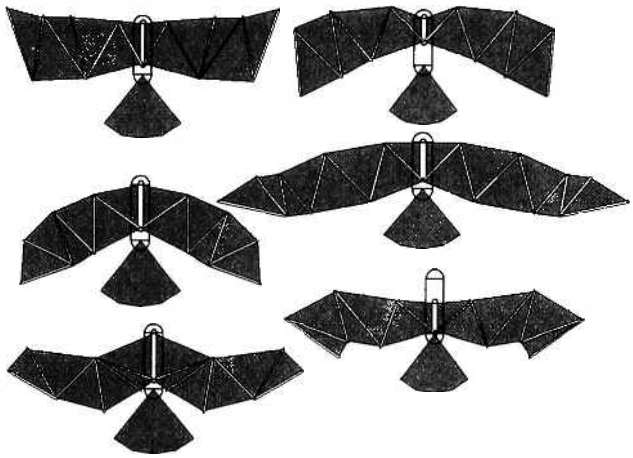

Figure 16: Various evolved wings.

Discussion and Future Work

We have described a system which generates virtual flying creatures and their path-following behaviors by artificial evolution. Several interesting creatures have evolved that exhibit organic and physically plausible flapping flight motions with various wing shapes. Additionally, we showed that the extremely difficult problem of evolving flying creatures capable of maneuvering through the air can be successfully addressed by a two-stage evolution process with robust sinusoidal controllers and a neural network. A new subspace of virtual flying creatures has been discovered in the hyperspace of possible creatures.

However, in our design, the wing structure is closer to that of a hang glider than that of a living creature, because of its non-flexible patagia. The flexibility of feathers or patagia is an important factors in the flight of real winged animals. The evolution of highly complicated shapes such as feathered wings is an extremely challenging problem. Flexible patagia would be a good start to enhancing this work but the computational cost of simulating the aerodynamics of deformable objects would need to be addressed for the time consuming evolution process. Since a creature's neural network is optimized only for a given trajectory, the sufficient training strategy or other more effective training algorithms need to be developed to produce creatures capable of following an arbitrary path. Going a step further, there is a need for research which investigates the way to evolve higher-level behaviors, such as diving toward a prey on the ground or soaring by sensing an ascending current of atmosphere. We expect that this method could also be applied to creatures in other environments (land, water) more easily with less running time, because of their comparatively stable conditions compared to creatures in the air. Another direction for future work might be to make an entire digital nature with a complete environment (land, water, sky) including evolved plants. As computers become more powerful, we believe that this sort of research will be a new frontier, which will give infinite diversity to the virtual world in the near future. Some movie clips of the work reported in this paper

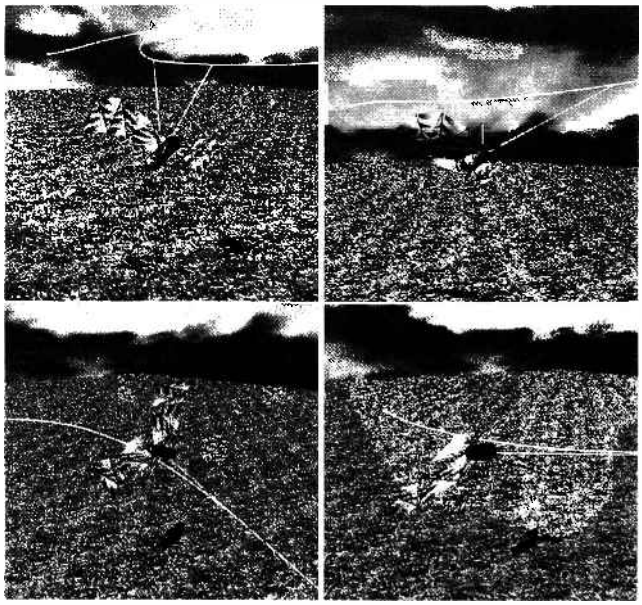

Figure 17: Creatures in motion.

are available at http://www.melkzedek.com/efc.htm.

References

Sims, K. 1994. Evolving Virtual Creatures, *ACM Computer Graphics (SIGGRAPH '94)*, pp. 15–22.

Sims, K. 1994. Evolving 3D Morphology and Behavior by Competition. R. Brooks, & P. Maes, (eds.) *Artificial Life IV: Proceedings of the Fourth International Workshop on the Synthesis and Simulation of Living Systems*, pp. 28-39, MIT Press.

Ventrella, J. 1998. Attractiveness vs. Efficiency: How Mate Preference Affects Locomotion in the Evolution of Artificial Swimming Organisms. Adami, C. et al. (eds.) *Artificial Life VI: Proceedings of the Sixth International Conference on Artificial Life*. pp. 178-186: MIT Press.

Komosinski, M and Ulatowski, S. 1999. Framsticks: Towards a Simulation of a Nature-Like World, Creatures and Evolution. Floreano, D. et al. (eds.) *Advances in Artificial Life: Proceedings of the Fifth European Conference on Artificial Life*, pp. 261-265: Springer Verlag.

Ray, T. S. 2000. Aesthetically Evolved Virtual Pets. Maley, C.C. & Boudreau, E. (eds.) *Artificial Life VII Workshop Proceedings*. pp. 158-161.

Lipson, H. and Pollack, J. B. 2000. Automatic Design and Manufacture of Robotic Lifeforms. *Nature 406,* pp.974-978.

Shim, Y. S. and Kim, C. H. 2003. Generating Flying Creature using Body-Brain Co-Evolution. in *ACM SIGGRAPH / Eurographics Symposium on Computer Animation,* pp.276-285.

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

Ramakrishnananda, B. and Wong, K. C. 1999. Animating bird flight using aerodynamics. *The Visual Computer.* 15(10) pp. 494-508.

Wu, X. C. and Popović, Z. 2003. Realistic Modeling of Bird Flight Animations. in *Proceeding of SIGGRAPH '87* , pages 888-895.

Smith, R. 1998. Intelligent Motion Control with an Artificial Cerebellum. *PhD Thesis, Dept of Electrical and Electronic Engineering, University of Auckland,* New Zealand. (Available online including ODE engine at http://opende.sourceforge.net)

Anitescu, M. and Potra, F. A. 1997. Formulating rigid multibodydynamics with contact and friction as solvable linear complementarity problems, *Nonlinear Dynamics 14,* 231.247.

Stewart, D. E. and Trinkle, J. C. 1996. An implicit time-stepping scheme for rigid-body dynamics with inelastic collisions and Coulomb friction, *International J. Numerical Methods in Engineering 39,* 2673-2691.

Withers, P. C. 1981. An aerodynamic analysis of bird wings as fixed aerofoils. *Journal of Experimental Biology 90,* 143.162.

Wejchert, J. and Haumann, D. 1991. Animation aerodynamics. *SIGGRAPH '91 Proceedings.* volume 25, number 4.

Mueller, T. J. 2001. Fixed and flapping wing aerodynamics for micro air vehicle applications. *Progress in Astronautics,* volume 195.

Fox, R. W. and McDonald, A. T. 1976. Introduction to fluid mechanics. fifth edition, Wiley.

Rivals, I., Personnaz, L., Dreyfus, G., and Canas, D. 1993. Real-time Control of an Autonomous Vehicle : A Neural Network Approach to the Path Following Problem. in *5th International Conference on Neural Networks and their Applications (NeuroNimes93),*