International Journal of CAD/CAM Vol. 9, No. 1, pp. 85~91 (2009)

INTERNATIONAL JOURNAL OF CAD/CAM
www.ijcc.org

Autonomous Animated Robots

Masahito Yamamoto, Kenji Iwadate, Ryosuke Ooe, Ikuo Suzuki, and Masashi Furukawa
Graduate School of Information Science and Technology

Abstract : In this paper, we demonstrate an autonomous design of motion control of virtual creatures (called animated robots in this paper) and develop modeling software for animated robots. An animated robot can behave autonomously by using its own sensors and controllers on three-dimensional physically modeled environment. The developed software can enable us to execute the simulation of animated robots on physical environment at any time during the modeling process. In order to simulate more realistic world, an approximate fluid environment model with low computational costs is presented. It is shown that a combinatorial use of neural network implementation for controllers and the genetic algorithm (GA) or the particle swarm optimization (PSO) is effective for emerging more realistic autonomous behaviours of animated robots.

Keywords **: physics modeling, artificial life, computer animation, automated motion design**

1. Introduction

Since Sims's seminal work of virtual creatures, research interests in the evolution of morphologies and controllers of three-dimensional virtual creatures on physically modeled environment have been grown [7]. Due to the recent availability of high performance computers, some physics engines can be easily used. Physics engines can simulate a three-dimensional physical environment including the gravity, the friction forces and the collision detections of objects, based on the differential equations. However, it is still difficult to design morphologies and controllers of virtual creatures simultaneously. Thus, relatively simple and unrealistic creatures consist of some spheres or some boxes are focused in the previous works [1, 3, 4, 8].

In the field of computer graphics and animation design, an autonomous motion design is one of recent hot topics. The technique of the creation of CG crowd is actually used in some major films [6]. It will be more valuable for designers to create motions of more realistic virtual creatures autonomously, because it is a time consuming task to create motions manually by using motion captures.

In this paper, we present more complicated and more realistic autonomous virtual creatures (called animated robots in this paper) and develop design software for creating them. An animated robot can behave autonomously by using its own sensors and controllers on three-dimensional physically modeled environment. The design software enables us not only to design three-dimensional animated robots consists of rigid objects, elastic objects with various joints and controllers, but also to simulate the behaviour of designed animated robots on the physical environment instantly at any time during the design process. In order to simulate more realistic world, an approximate fluid environment model is presented, which enables us to implement virtual fluid environments with low computational costs. In the developed software, designed animated robots may have sensors and controllers with neural networks and these controllers can be evolved by means of the genetic algorithm (GA) or the particle swarm optimization (PSO). In this paper, we demonstrate that the salamander model can obtain an autonomous both walking behaviors and swimming behaviors on virtual land and water environments. Furthermore, in order to show the validity of the approximate fluid environment, the flight control task of the helicopter model is demonstrated.

*Corresponding author:
E-mail: masahito@complex.eng.hokudai.ac.jp

2. Animated Robot

An animated robot is a virtual object on three-dimensional physically modeled environment. The animated robot can behave autonomously by controlling its actuators based on its sensor inputs such as the distance and the angle of the light sources, other objects and so on. We can design an animated robot by conjugating some primitive objects such as spheres, cuboids, meshed spheres and meshed cuboids. Figure 1 shows an example of our designed gold beetle and dragon fly robots.

In our designed software, not only rigid objects such as shown in Fig. 1, but also elastic objects can be designed. Elastic objects are 3D physics models based on the spring and the math objects (very small rigid objects). An arbitrary shape of an animated robot can be designed by using some rigid and elastic (spring) objects.

Particularly, in the process of designing an elastic animated robot, it is very important to know the stability of the robot on the physical environment with

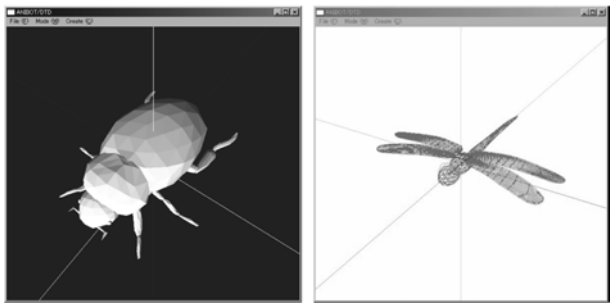

Fig. 1. A gold beetle robot and a dragon fry robot.

gravity, frictions and collisions. By simply switching the mode from "modeling" to "simulation" in the software, the behaviors of designed animated robots can be simulated. Thus we can know the behaviors of designed anibots and the stability of it immediately. This feature is novel and can support the modeling process of these virtual creatures.

Figure 2 shows some typical elastic models, but an *n*-hedron object (c) has unstable structure on physical environment. In our design software, we can know such an unstable structure during the design process.

3. Design Software

3.1 Design of Animated Robots

An animated robot consists of some rigid objects with some joints. The structure of the robot can be stored into the database in the form of graph structures. Nodes in the data structure correspond to the object components such as rigid objects and joints, and links correspond to the joint relation between these objects.

A rigid object is a main component of animated robots. Some primitives such as spheres, cuboids, meshed spheres and meshed cuboids are prepared for designing animated robots. Figure 3 shows an example of various rigid objects.

A joint object can be used to conjugate a pair of rigid objects. Each joint has six degrees of freedom. Optionally, some of six degrees of freedom may not be used, i.e., fixed joints are implemented. An attribute of the joint can be changed to be a flexible elastic spring based on

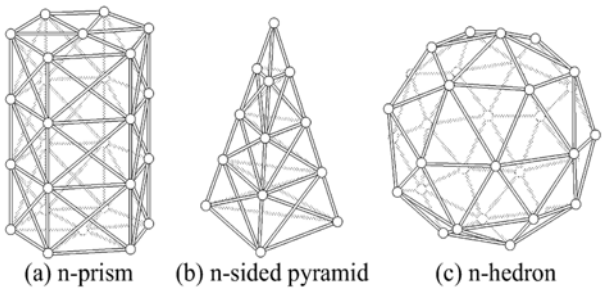

Fig. 2. Design of three types of elastic robots.

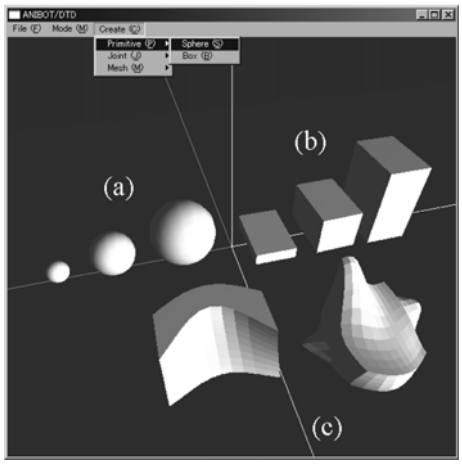

Fig. 3. Various rigid objects (a) spheres (b) boxes (c) meshed objects.

the spring-damper model. In this case, each joint has some parameters, a spring factor and a damping factor, to determine the spring strength.

Of course, designed objects can be copied and saved. The data representing an animated robot are stored in the XML format.

3.2 Design of Motions

In order to animate designed objects, some kinds of sensors are prepared. In the current version of our developing software, photo sensors and touch sensors are implemented. However, other sensors such as image sensors, or sonar sensors can be easily implemented. The photo sensors can be attached to the surface of rigid objects. Input signals from the sensors are used for motion control.

A motion of an animated robot is created by the rotations of joints, and expansions and contractions of springs. Two types of motion attributes are prepared. The first one is the attribute for a passive motion caused by surrounding forces. The second one is the attribute for an active motion that makes an effect with surrounding forces to control a model. The active motion is controlled by a neural network controller.

The user of the developed design software can switch the mode between "modeling mode" and "simulation mode" at any time during the design process. In the modeling mode of our designed software, above explained three-dimensional anibots can be easily designed as well as the typical 3D modeling tools. In the simulation mode, designed objects laid to the physical environment with gravity. We adopt PhysX engines by NVIDIA Cooperation [5]. Therefore we can know the behaviors of designed animated robots immediately during the modeling process. It will be very helpful for such kind of design software for virtual creatures. Figure 4 shows a demonstration image of the mode switching. Four kind of objects are designed in the modeling mode (Fig. 4 (a)), and the simulation result is obtained immediately

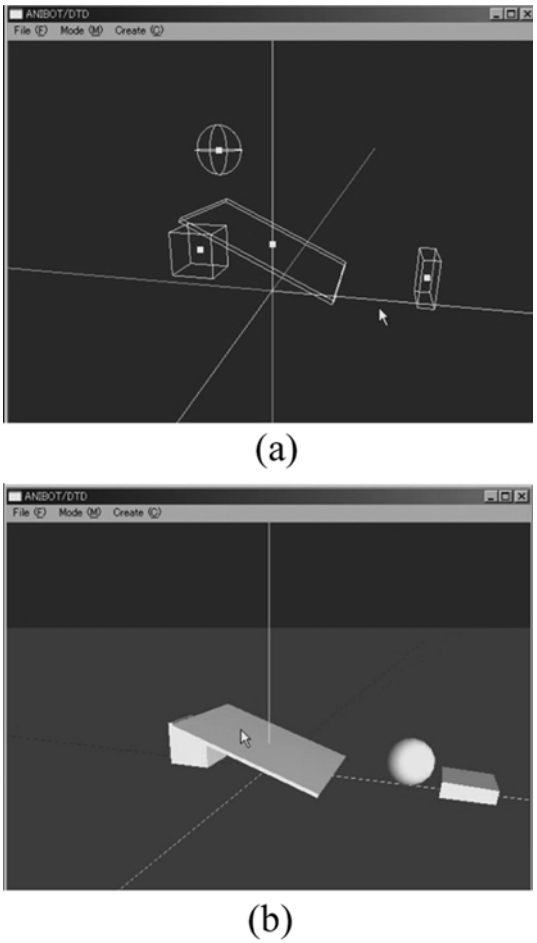

Fig. 4. Design mode (a) and simulation mode (b).

by switching to the simulation mode (Fig. 4 (b)).

3.3 Design of Physical Environment

As described in the previous subsection, PhysX engine is used for simulating the physical environment. However, it is very difficult to model the fluid effects and compute the force of drag and lift. In particular, the particle method is very time-consuming method, thus it is not appropriate for developing such kind of animated robots, because many high-speed simulations are required for optimizing their motion controllers. In this paper, an approximate fluid environment model is proposed, which can reduce the computational time drastically compared with the particle method.

Suppose that there is a flat board in the uniform fluid environment as shown in Fig. 5. If the flat board moves with the speed $\boldsymbol{v}$ in the uniform fluid with the velocity $\boldsymbol{u}$, the force $\boldsymbol{F}$ of drag to the flat board is calculated in the following equation:

$$F = -\frac{1}{2}\rho A_p C_d v^2 \boldsymbol{n} \quad (1)$$

where ρ is the density of the fluid, A_p is the reference

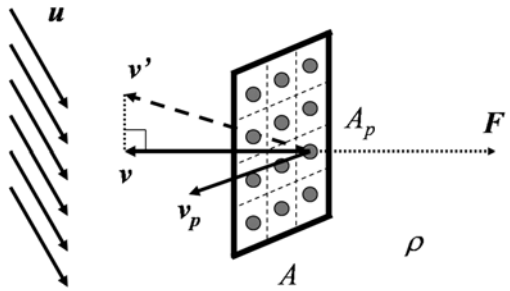

Fig. 5. Force of drag.

area, C_d is the drag coefficient, $\boldsymbol{n}$ is the unit vector indicating the direction of the velocity (the negative sign indicating the drag is opposite to that of velocity).

In order to compute the force of drag more precisely in the case of the flat board moves and rotates, the flat board is divided into some equal size areas and the equation (1) is applied to each area. More complicated meshed object can be treated in the same fashion, i.e., each mesh is divided into the small area and is led to compute the force of drag. Note that the drag coefficient is highly dependent on the size and the shape of the whole object. In general, it is very difficult to determine the precise values of it for any object, thus C_d is set to 1.5 for all objects in this paper.

3.4 Optimization of Motion Control

An animated robot takes the input signals from the environment by using its sensors and outputs the torques to all controllers (joints) and the behavior of the robot on the physical environment is determined as a result. Of course, the animated robot is affected by other objects, for example, the collisions against other objects. In order to compute these effects, physics engines are very useful.

According to the previous studies, an Artificial Neural Network (ANN) is used for controlling the actuators (joints). However the connection structure of ANN is not limited to the feed forward network in this paper. If random values are assigned to the weight of the neural network, random behaviors are obtained regardless of the input signals.

In order to acquire reasonable behaviors of animated robots, we adopt a genetic algorithm (GA) and a particle swarm optimization (PSO) for optimizing the weight values of each neural controller. Combining ANN and GA (or PSO) is a most promising approach for optimizing this kind of motion control. The obtained behaviors are strongly depending on the predetermined fitness function. The optimization of behaviors will be described in the next section.

4. Design of Autonomous Behaviors

In order to make more realistic autonomous behaviors of designed objects, the neural controllers are optimized

so as to maximize the fitness function. We present two demonstrations for acquiring the autonomous behaviors of animated robots.

4.1 Salamander model

First, we present a salamander model simulation. A salamander model is designed by using our developed modeling software by one of the authors (see Fig. 6). It consists of 20 meshed primitives, two sensors and 13 joint controllers. The density of the model is 300 [kg/m^3]. The coefficients of static and dynamic frictions are 0.9 and 0.8, respectively. The restitution coefficient is set to 0.3.

The objective of the salamander model is to reach the light source put on the physical environment. The sensor inputs are the light strength and the difference between the sensor position and the light position. The output torque for each controller is computed in the following equation:

$$Torque_{\,i} = R_i \cdot \sin(\omega_i t + \varphi_i) + B_i \tag{2}$$

where R_i is an amplitude, ω_i is an angular velocity, φ_i is a phase and B_i is a bias. Although the sine function is used for creating the cyclic forces, it may not be essential in case that the simulation step (time interval of sensor inputs) is relatively small.

In the GA optimization process, some parameters are set to the values as shown in Table 1. A mutation operation is implemented by replacing the values of randomly selected weights to random values. A crossover operation is executed by replacing the output neuron and its all connecting weights with that of randomly selected controllers of randomly selected individuals.

Fitness functions are crucial for the evolution process. In this paper, we adopt the weighted sum function

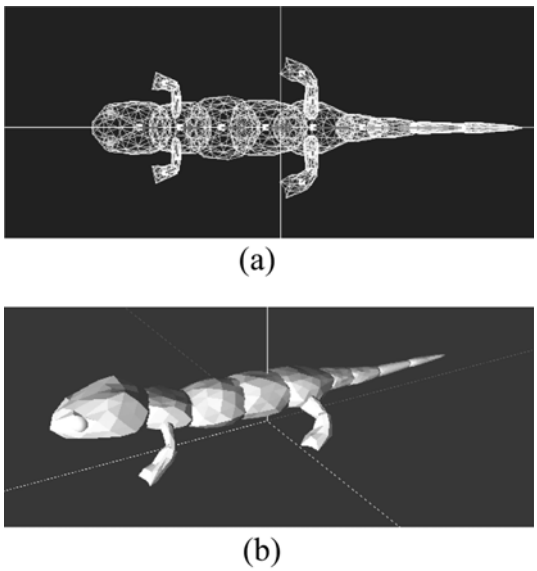

Fig. 6. Salamander model.

Table 1. GA Parameters for the salamander model

Population size	20
Crossover provability	0.4
Mutation provability	0.05

consists of distance, energy and direction factors. If the distance (D) from the light source to the robot is small during one simulation, the robot obtains better fitness values. If the energy consumption (E) is small and if the direction difference (L) between sensors and light source is small, the animated robot gets better fitness values. The fitness function is as follows.

$$f = -aD - bE + cL \ (a, b, c : constant) \tag{3}$$

$$aD > cL > bE \tag{4}$$

The direction factor (cL) is not always required, but it may be necessary for obtaining an animal-like behavior, because the animal tends to direct to the target.

After the evolution with 100 generations, reasonable walking behaviors of salamander model by twisting the body are obtained. Figure 7 shows the snapshot of the evolved walking behaviors of the salamander model. A swimming behavior of the salamander model is also evolved in the water environment (data not shown).

4.2 Helicopter Flight Control

We present another demonstration of the flight control of helicopter model in order to show the validity of the approximate fluid environment presented in section 3.3. The helicopter model as shown in Fig. 8 is inspired by the sketch of Leonard da Vinci. However, it is well known that da Vinci helicopter cannot flight stably.

First, we verify that the flight of da Vinci helicopter is unstable in our approximate fluid environment, and then the revised helicopter model is developed and the flight controller is optimized by using the particle swarm optimization (PSO) [2].

The da Vinci helicopter has the axis with the length 1.2[m] and width 0.1[m]. The wing with 0.6[m] radius winds 1.5 rounds. The density of the axis and the wing is 100.0 [kg/m^3] and 85.5 [kg/m^3], respectively.

By adding the torque to the axis of the helicopter, the flight simulation is executed. In our simulation result, unstable flight of the da Vinci helicopter is observed (data not shown). Although our approximate fluid environment model does not consider the force of lift, the affect of wind and the noise, it is shown that our proposed environment model is valid from the above simulation result.

Next, we present the revised simple helicopter model in order to enhance the flight ability of da Vinci helicopter as shown in Fig. 9. The revised model has two propellers that rotate to the opposite directions by

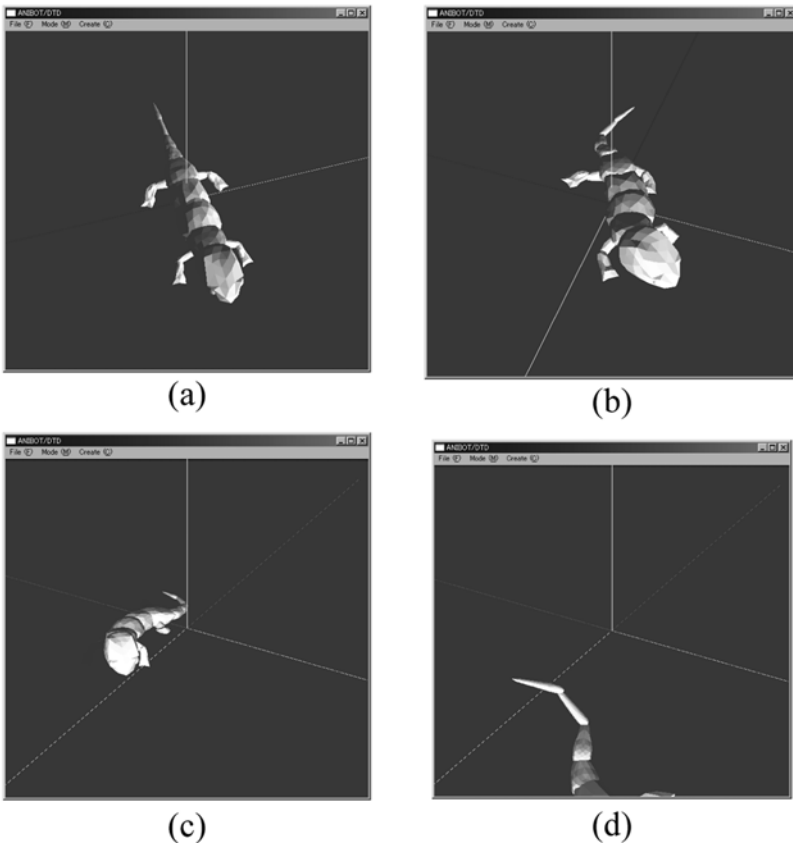

Fig. 7. Evolved walking behaviors.

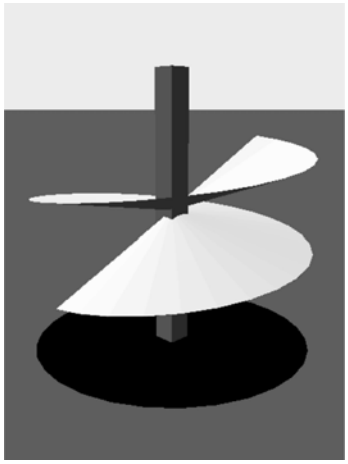

Fig. 8. da Vinci helicopter model.

giving the torque to the axis. The revised model has the axis with the length 0.6[m] and width 0.1[m]. The wing is the box with the size of 1.8[m] × 0.2[m] × (3.0×10^{-3})[m], and fixed to the axis with 35°. The density of the axis and the wing is all 100.0 [kg/m^3].

The objective of the flight control is to keep the target altitude. This task is not so difficult, but it is appropriate for verifying that our approximate fluid environment model is valid and the optimization method works well.

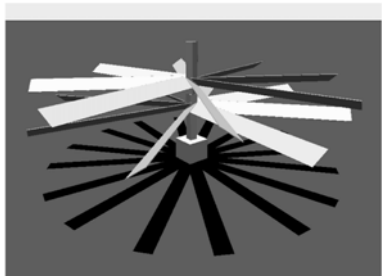

Fig. 9. Revised simple helicopter model.

Artificial neural network (ANN) controllers are used for determining the torque of the axis at time *t*. The sensor inputs of the model are the altitude and the speed to the vertical axis. The number of neurons in hidden layer is 4 and the output of ANN is the torque.

The sigmoid functions $f(u)$, $g(u)$ for hidden layer and output layer are calculated in the followings.

$$f(u) = \frac{2\alpha_f}{1+e^{-\beta_f u}} - \alpha_f \tag{5}$$

$$g(u) = \frac{2\alpha_g}{1+e^{-\beta_g u}} - \alpha_g \tag{6}$$

Particle Swarm Optimization (PSO) is a relatively new collective approach to optimization problems [7]. In order to optimize the torque control for keeping the flight altitude, PSO is applied to the weights of ANN and α_f, β_f, α_g, β_g in equation (5) and (6).

The population size of PSO is 20 and the number of search (update) iterations is 1000. The value of inertial constant is decreasing from 0.9 to 0.4 linearly and other coefficients are set to 2.0.

The fitness function f is given by the following equations:

$$f=\sum_{t=1}^{T} V_t \tag{7}$$

$$V_t=\begin{cases} \dfrac{1}{(1+|20-h|)^2} & (h\geq 2) \\ 0 & (\text{otherwise}) \end{cases} \tag{8}$$

where h is the altitude of the helicopter, and T is the simulation step (T = 1800).

Figure 10 shows the transition of fitness values in the search iterations. From this figure, we can verify that PSO works well, because the maximum fitness value is increasing as the generation proceeds.

Figure 11 shows the transition of altitude of the helicopter in some typical generations. In generation 15, the descending behavior is observed for the first time. As the generation is proceeding to 1000, an optimal behavior is obtained.

Therefore, it is shown from these experiments that our approximate fluid environment model works well although its proof is not given.

5. Concluding Remarks

In this paper, we present modeling software for designing animated robots that can behave autonomously on 3D physical environments. The main feature of our developing software is to support users to design animated robots by switching between the design mode and simulation mode at any time. Also an approximate fluid environment model is presented in order to simulate the water or the air environments with low computational costs. The neural network of a designed animated robot is optimized and reasonable behaviors of the robot are obtained.

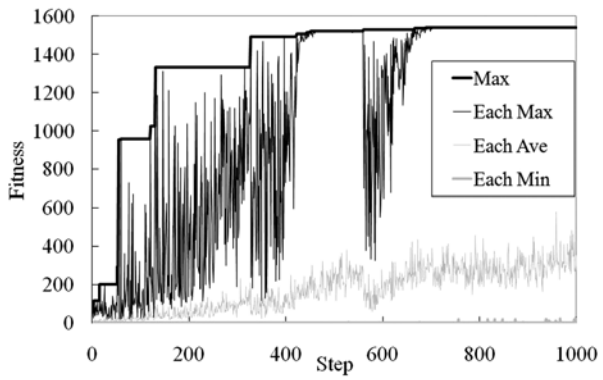

Fig. 10. Fitness values in the search process.

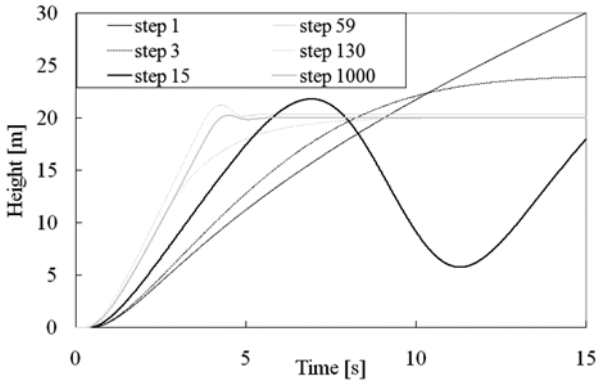

Fig. 11. Obtained behavior of the helicopter model in typical generations.

References

[1] Chaumont, N., Egli, R. and Adami, C. (2007), Evolving Virtual Creatures and Catapults, *Artificial Life*, **13**(2), 139-157

[2] Kennedy, J. and Eberhart, R. (1995), Particle Swarm Optimization, *IEEE International Conference on Neural Networks*, 1942-1948

[3] Kim, K. J. and Cho, S. B. (2006), A Comprehensive Overview of the Applications of Artificial Life, *Artificial Life*, Vol. **12**(1), 153-182

[4] Masry, M. and Lipson, H. (2005), A Sketch-Based Interface for Iterative Design and Analysis of 3D Objects, *Proceedings of Eurographics workshop on Sketch-Based Interfaces*, 109-118.

[5] NVIDIA Cooperation (November 30, 2009), http://www.nvidia.com/

[6] Reynolds, C. W. (1987), Flocks, Herds, and Schools: A Distributed Behavioral Model, *Computer Graphics*, Vol. **21**(4), 22-34.

[7] Sims, K. (1994), Evolving Virtual Creatures, *Proceedings of the 21st Annual Conference on Computer Graphics and Interactive Techniques*, 15-22.

[8] Sugiyama, Y. and Hirai, S. (2006), Crawling and Jumping by a Deformable Robot, *International Journal of Robotics Research*, Vol. **25**, 603-620.

Masahito Yamamoto He is an Associate Professor in Graduate School of Science and Technology, Hokkaido University, Japan. He received a Ph. D. from Hokkaido University in 1996. His current research interests include evolutionary creations of autonomous intelligent virtual creatures, an analysis of complex networks and large-scale optimizations.

Ryosuke Ooe He is a student in the master's program of Graduate School of Information Science and Technology, Hokkaido University, Japan. He received his BSc at Hokkaido University in 2009. His research interests include a computer simulation of flying creatures.

Masashi Furukawa He received the B.S and M.S degrees from Hokkaido University in 1971 and 1973 respectively, and received Dr. of Engineering from Hokkaido University in 1981. He was the research associate, associate professor, and professor of Asahikawa National College of Technology from 1973 to 2006. He has been a professor of Hokkaido University since 2006. He was employed as the NSF research associate at Cornell University in US in 1976 - 1977. Also, He was the visiting professor of University of East Anglia in UK in 1981-1982. He was engaged in the development of 3D CAD system in 1971 - 1986. Now, his research interest is the complex network, A-life, and meta-heuristics.

Kenji Iwadate He is a Ph. D. candidate of Graduate School of Information Science and Technology, Hokkaido University, Japan. He is also an affiliate of the Japan Society for Precision Engineering.

Ikuo Suzuki He is an Assistant Professor in Graduate School of Information Science and Technology, Hokkaido University, Japan. He received a Ph.D. in engineering from Hokkaido University in 2004. His current research interests include Complex engineering, Computer graphics and Computer animation.

Masahito Yamamoto

Kenji Iwadate

Ryosuke Ooe

Ikuo Suzuki

Masashi Furukawa