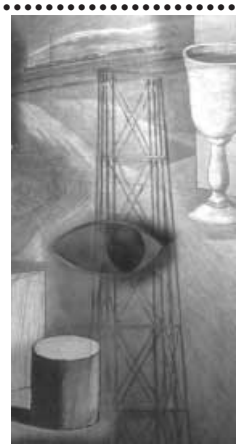
THE JOURNAL OF VISUALIZATION AND COMPUTER ANIMATION
J. Visual. Comput. Animat. **10**, 215–224 (1999)

An Efficient Control over Human Running Animation with Extension of Planar Hopper Model



By Young-Min Kang, Hwan-Gue Cho and Ee-Taek Lee*

The most important goal of character animation is to efficiently control the motions of a character. Until now, many techniques have been proposed for human gait animation. Some techniques have been created to control the emotions in gaits such as 'tired walking' and 'brisk walking' by using parameter interpolation or motion data mapping. Since it is very difficult to automate the control over the emotion of a motion, the emotions of a character model have been generated by creative animators. This paper proposes a human running model based on a one-legged planar hopper with a self-balancing mechanism. The proposed technique exploits genetic programming to optimize movement and can be easily adapted to various character models. We extend the energy minimization technique to generate various motions in accordance with emotional specifications. Copyright © 1999 John Wiley & Sons, Ltd.

Received July 1999

KEY WORDS: animation; human gait; genetic programming; energy control

Introduction

The most important goal of character animation is to efficiently control the characters. Many techniques for automatic motion generation have been proposed, but it is very difficult to control the motions of characters with those techniques. On the other hand, techniques which provide powerful control ability, such as the keyframe method, lead the motion creation to inefficient work. For a long time it has been the target of many researchers to find efficient control techniques which control the motions of characters with an automated process.[1]

A common approach to character animation is to simulate the dynamics of an articulated figure.[1–3] This approach can generate plausible animation, since the motion is forced to fit the physical laws. However, dynamic simulation requires heavy computation and it is difficult to represent autonomous behaviours. Another technique, SAN (sensor–actuator network), exploits the neural network model.[4] Although this approach has the advantage that it requires no *a priori* knowledge of complicated motion equations, animators cannot predict the results. One disadvantage of this kind of automation technique is that it suffers from a lack of control over the detailed behaviour of characters.

In this work we consider the animation of running humans. Many techniques have been proposed to simulate the motions of the human skeletal structure.[5–7] The straightforward approach of these techniques uses rotoscopy, which maps real motion data obtained by experiments to a skeletal model. This approach is very simple but it requires expensive hardware devices to obtain motion data of a real model.

Raibert and Hodgins implemented 'animation of dynamic legged locomotion' with numerical integrating equations of motion derived from the physical models.[8] They exploited the hopper model developed in biomechanics to implement their running models such as biped and quadruped models.

Another approach is to generate new motions with parameter interpolation or extrapolation of the given motion data.[9] For example, given two walking motions, tired walking and brisk walking, this technique can synthesize a more tired walking or a more brisk walking by

*Correspondence to: Y.-M. Kang, Department of Computer Science, Pusan National University, Pusan 609-735, South Korea. E-mail: ymkang@pearl.cs.pusan.ac.kr

Contract/grant sponsor: Electronics and Telecommunications Research Institute (ETRI).

 CCC 1049–8907/99/040215–10$17.50

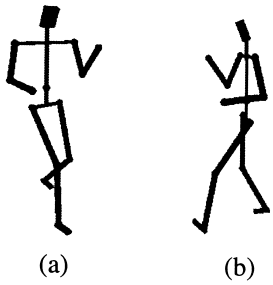


Figure 1. Character model: (a) front view; (b) side view

extrapolating the given motion data. One disadvantage is that it cannot create any motions if there are no motion data. Thus we cannot apply this animation technique to the virtual or imaginary creatures that often appear in cartoon-like situations. This paper describes a new automatic generation method and control of human running animation without any motion data. The most important advantage of our control technique is that it can be easily adapted to various characters such as animals and even imaginery characters.

When animating characters, balance should be taken into account. The implementation of a balancing motion is essential for realistic animation. Balancing has been studied extensively in robotics.[10,11] In the field of computer graphics, Boulic and Mas implemented the balancing motion using inverse kinematics.[12] However, such methods require heavy computation.

The running model presented in this paper is based on the planar hopper model, and a simple balancing technique is introduced. The basic idea in our technique is that the motion characteristics of a character can be dominated by energy consumption, which can be a good parameter for animation. In order to find the optimized running dynamics, we use the genetic programming approach, where each character model has its own 'gene' sequences. Many works have shown good performance of the genetic programming approach to the animation of virtual characters.[13–15]

System Overview

Our character model is shown in Figure 1. This model has 11 joints with various degrees of freedom. The user interface of our system is shown in Figure 2. For the modelling of the running motion, we first developed a one-legged planar hopper, which has been studied extensively in robotics;[16] we then extended this one-legged hopper to a two-legged planar hopper by adding more parameters. The realistic human running motion requires not only realistic movements of legs but also a balancing mechanism to support the upper body.

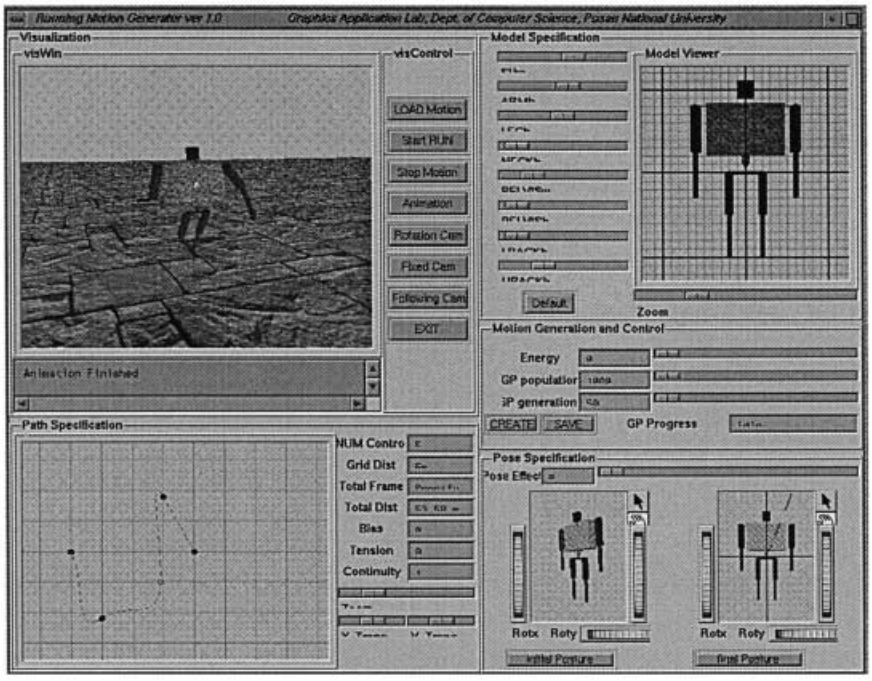


Figure 2. System interface

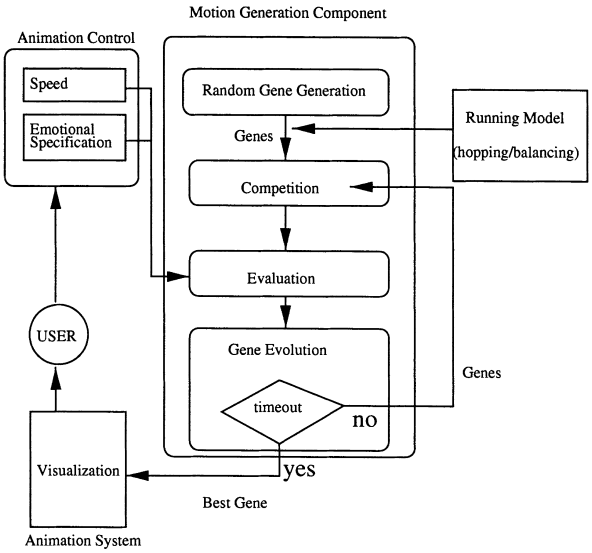


Figure 3. System flow

To provide conceptual control over the character model, we regard a set of characteristic parameters as a 'gene sequence' and apply the genetic programming technique, which finds the optimal gene to animate a natural-looking running motion. To generate motion, the system creates a population of character models and forces the characters to compete with each other. Each character model has its own gene (parameters for a running model) and runs forward according to its gene. After each character finishes its motion, the system evaluates the motion of each character. Since the basic idea of our control technique is that we can control the motions of characters with energy consumption, the system evaluates the performance of each character based on its energy consumption. According to the evaluation, the system selects the characters which have good genes and makes them evolve by applying crossover between good genes or mutation. Figure 3 shows the flow of motion generation with conceptual parameters, speed and briskness (i.e. the amount of energy that a user requires the character to consume).

Running Model

One-legged Hopper

Let us consider the one-legged hopper shown in Figure 4 as a basic model for human running. The initial state of the one-legged hopper is shown in Figure 4. This hopper is

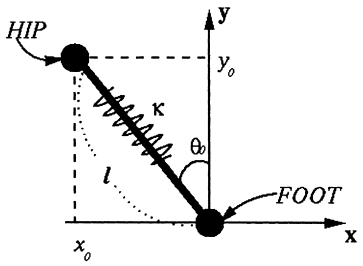


Figure 4. Initial state of one-legged hopper

assumed to move in the positive direction of the x axis. The length of the leg can vary in running, so the leg acts like a spring if its length is shorter than the initial length. *HIP* denotes the hip location of this hopper and *FOOT* denotes the foot location. The positions of the hip and foot are described as (x, y) and (f_x, f_y) respectively. The parameters determining the motion of the hopper include spring constant (κ), mass of hopper (m), length of leg (l), landing angle (θ_0) and horizontal velocity ($\dot{x}$). The locations of the hip and foot at the initial state (HIP_0 and $FOOT_0$) are given as

$$HIP_0 = (-l\sin\theta_0, l\cos\theta_0)$$
$$FOOT_0 = (0, 0) \quad (1)$$

This running model consists of two alternating phases: a standing phase and a jumping phase. The standing phase denotes the time period when the foot is placed on the ground ($f_y = 0$) and the jumping phase denotes the time period when the foot is lifted above the ground ($f_y > 0$). The running of the one-legged hopper can be described by alternately repeating these two phases.

Standing phase of one-legged hopper. Figure 5 shows the four key elements that affect the trajectory of the hip during the standing phase (gravity g, horizontal velocity $\dot{x}$, spring constant κ and angle θ between the leg and the y axis). We assumed that the horizontal velocity $\dot{x}$ is constant in running and that the vertical velocity $\dot{y}$ is determined by g and the spring force of the leg. Let h denote the time interval between two consecutive animation frames. The hip location (x_t, y_t) at time t is described as

$$x_t = x_{t-h} + \dot{x}h \quad (2)$$
$$y_t = \frac{1}{2}\left(gh^2 + \frac{\kappa(l-d)h^2\cos\theta_t}{m}\right) + 2y_{t-h} - y_{t-2h}$$

where θ_t is the angle between the leg and the y axis at time t. Although we assumed the horizontal velocity as a

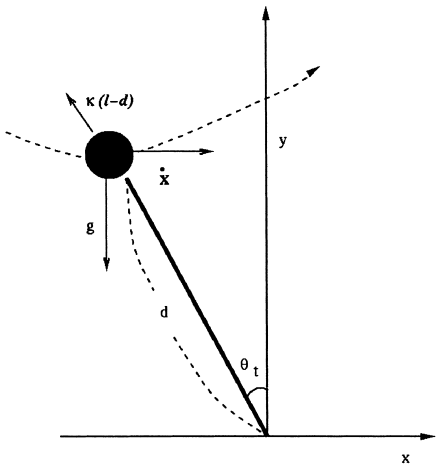


Figure 5. Key elements determining standing phase

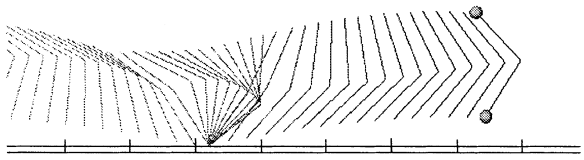


*Figure 6. Motion of one-legged hopper (*h=*0·005 s,* l=*2 m,* $\dot{x}$=*10 m s*$^{-1}$, κ=*3000,* $\theta_0=\pi/4$, m=*50 kg)*

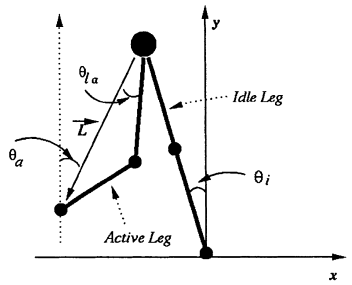


Figure 7. Initial state of two-legged hopper

constant, acceleration or deceleration in the horizontal direction can be considered for more realistic motion.

The hopper should be in the standing phase if the leg length d is shorter than the initial leg length l. If d becomes longer than l, the running phase is switched to the jumping phase and the system calculates the key variables for the creation of a jumping motion. These variables include T, which is the time to be taken during the jumping phase. During the jumping phase the movement of the hip of the hopper is determined only by gravity. The next subsection will explain how to calculate T.

Jumping phase of one-legged hopper. During the jumping phase the location of the hip can be calculated as

$$HIP_t=(x_t,y_t)=\left(x_{t-h}+\dot{x}h, y_0+\dot{y}_0 t+\frac{gt^2}{2}\right) \quad (3)$$

where $\dot{y}_0$ denotes the initial velocity of the hip in the jumping phase, y_0 denotes the y co-ordinate value of the hip location at the starting time of the jumping phase and t denotes the current time variable after the initial time of the jumping phase.

The jumping time T is the time interval between the initial jumping time and the time when y becomes the height at which the hopper can switch to the standing phase. The height is the same as the initial height of the standing phase. Thus y_f, the height at the completion time of jumping, is $l\cos\theta_0$ and the jumping time T is t that satisfies the equation $y_0+y_0t+gt^2/2=y_f$. Thus T can be calculated as

$$T=\frac{-\dot{y}_0\sqrt{\dot{y}_0^2+2gc}}{g} \quad (4)$$

where $c=y_f-y_0$.

During the time interval T the hopper jumps and the location of the hip is affected by gravity. The angle between the leg and the y axis (θ) is interpolated to be θ_0 after T. Figure 6 shows the running animation of the one-legged hopper model which has a joint to represent a human leg.

Two-legged Hopper

The technique for the one-legged hopper can be extended to fit a two-legged hopper. As shown in Figure 7, a two-legged hopper requires the additional parameter $\vec{L}$ for specifying an initial state. Note that $\vec{L}$ is defined as $FOOT-HIP$.

Standing phase of two-legged hopper. The running of a two-legged hopper also consists of a standing phase and a jumping phase. During the standing phase the leg on the ground moves exactly the same as in the one-legged hopper. Let $FOOT_a$ denote the lifted foot. The location (f_{x_a}, f_{y_a}) of $\mathrm{FOOT_a}$ is calculated as

$$\dot{f}_{x_a}=\dot{x}S_x \qquad \dot{f}_{ya}=\dot{y}S_y \quad (5)$$

where $\dot{x}$ and $\dot{y}$ are the horizontal and vertical velocities of the hip respectively. S_x and S_y are the scale parameters for

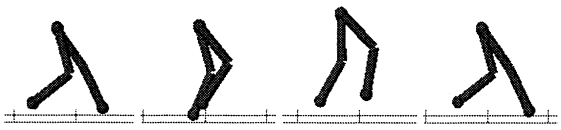


*Figure 8. Movement of two-legged hopper (*h=*0·01 s*, l=*1·5 m*, $\dot{x}$=*5 m s*$^{-1}$, κ=*9000*, θ_o=π/*6*, m=*50 kg*, α=*0·9*, S_x=*1·5*, S_y=*1·0)*

$\dot{x}$ and $\dot{y}$ and are calculated automatically during the motion generation process. To compute suitable S_x and S_y, we will later define the connectivity γ.

Jumping phase of two-legged hopper. In the jumping phase the angles of the joints are interpolated during time interval T. The angles of the lifted leg in the previous standing phase must be changed to support the body in the next standing phase, and the angles of the supporting leg in the previous standing phase must be lifted in the next standing phase. The following equations are used for the interpolation of the angles:

$$\theta_{l_t} = \theta_{l_0} + (\theta_{l_T} - \theta_{l_0}) B_l(t)$$
$$\theta_{c_t} = \theta_{c_0} + (\theta_{c_T} - \theta_{c_0}) B_c(t) \quad (6)$$

where θ_{l_t} denotes the angle between the vectors $JOINT - HIP$ and $FOOT - HIP$, and θ_{c_t} is the angle between $HIP - FOOT$ and the y axis at time t. θ_{l_o} and θ_{c_o} denote the angles at time 0, the beginning of jumping. θ_{l_T} and θ_{c_T} denotes the angles at time T, the completion time of jumping.

$B_l(t)$ and $B_c(t)$ which were used in the above interpolation equation are functions of t, and both are set to zero when $t=0$ and to one when $t=T$. $B_l(t)$ is defined as $B_l(t)=t/T$ to linearly interpolate θ_{l_0} and θ_{l_T} during time T. $B_c(t)$ is defined as

$$B_c(t) = \left(\frac{1}{T^2(1-2\alpha)}\right)t^2 + \left(\frac{-2\alpha}{T+(1-2\alpha)}\right)t \quad (7)$$

where α is a parameter that controls the maximum (if $\alpha>0{\cdot}5$) or minimum (if $\alpha<0{\cdot}5$) value of $B_c(t)$ to be αT. If $\alpha=0{\cdot}9$, the difference between θ_{c_t} and θ_{c_0} becomes maximum when $t=0{\cdot}9T$. If we set $\alpha=1$, $B_c(t)$ increases monotonically and keeps $|\theta_{c_t} - \theta_{c_0}|$ greater than $|\theta_{c_t} - \theta_{c_0}|$ in the whole jumping phase. Therefore this parameter determines the movements of the legs in the jumping phase. The whole motion of the two-legged hopper is determined by control parameters which include l, $\dot{x}$, κ, θ_0, m, α, S_x, S_y and $\vec{L}$. Figure 8 shows the running animation of the two-legged hopper.

Balancing the Upper Body

In order to animate natural human running, it is important to make a balanced posture according to the movements of the arms, shoulders and legs. The movement of the upper body is decomposed into gravitational movement (passive) and human muscular movement (active).

To balance the upper body, we defined the target centre of mass and the current centre of mass. The target centre of mass is the location where the centre of mass of the character should be located, and the current centre of mass is the actual centre of mass of the character. The target centre of mass is assumed to be the foot placed on the ground in the standing phase. For physically correct motion we should consider the zero-moment point (ZMP) rather than the centre of mass. However, we just considered the centre of mass for simplicity, because the goal of our work is not to simulate the physically correct motion of humans or animals but simply to generate plausible animation.

In the jumping phase the target centre of mass is calculated by interpolating the two feet in accordance with time t. The target centre of mass can be calculated as

$$CM_{\text{tar}} = FOOT_{\text{i}} \quad (8)$$

if mode is *STANDING*

$$CM_{\text{tar}} = FOOT_i + (FOOT_a - FOOT_i)t/T$$

if mode is *JUMPING*

Note that t is the time variable and T is the total time needed for the jumping phase. The current centre of mass can be calculated as

$$CM_{\text{cur}} = \frac{\Sigma_i^n M_i L_i}{\Sigma_i^n M_i} \quad (9)$$

where M_i denotes the mass of each point, L_i denotes the location of each point and n denotes the number of points. We define the error between the current centre of mass and the target centre of mass as

$$\vec{\delta} = CM_{\text{tar}} - CM_{\text{cur}} \quad (10)$$

where δ_x and δ_z denote the x axis and z axis components of $\vec{\delta}$ respectively. When δ_x is positive, i.e. the target centre of mass is in front of the current centre of mass in the x axis direction, the body must bend forward to make a stable posture. In other words, the body needs more

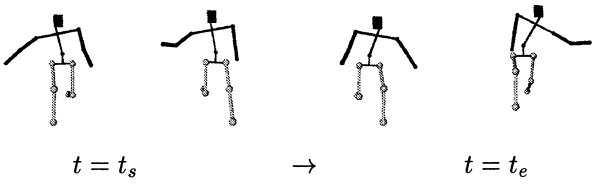


Figure 9. Balancing motion ($\eta = 5$, $\dot{x} = 1$)

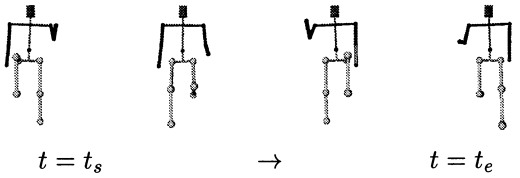


Figure 10. Balancing motion ($\eta = 5$, $\dot{x} = 5$)

rotation about the z axis. Let zd_b denote the rotation about the z axis to compensate the error; then the degree of rotation of the body about the z axis, Z_b should be

$$Z_{b_t} = Z_{b_{t-h}} + \frac{\dot{Z}_{b_{t-h}} h + \ddot{Z}_{b_{t-h}} h^2 + zd_{b_{t-h}}}{2} \tag{11}$$

where $\dot{Z}_{b_t}$ and $\ddot{Z}_{b_t}$ are the angular velocity and angular acceleration of Z_b at time t respectively. zd_{b_t} was used to move the current centre of mass to the position of the target centre. This zd_{b_t} enables the character to take a realistic balancing motion. In this paper we simply defined zd_b as

$$zd_b = \eta \frac{\delta_x h}{\dot{x}} \tag{12}$$

where h is the time interval between two consecutive frames, $\dot{x}$ is the horizontal velocity and the parameter η controls the sensitivity to the error. If η increases, the character becomes more sensitive to the error and the balancing effect works rapidly. To simulate the balancing motion, other joints also respond to the error. For example, the body also responds to δ_z, and the response can be expressed as $xd_{b_t} = \eta(-\delta\zeta)h/\dot{x}$ to make the body rotate about the x axis. Figures 9 and 10 show the results of our balancing technique with $\eta = 5$ and different velocities from time t_s to t_e, where the body and arms are forced to balance the motion.

We tested this balancing technique with a staggering walk. In the staggering motion a character spends a shorter time when the staggering leg is supporting the whole body compared with when the other leg is supporting it. The result is shown in Figure 11. We can see that the centre of mass is biased to the leg that is not staggering. Figure 12 shows the trajectory path along the target centre of mass and the current centre of mass. The broken line shows the trajectory of the target centre of mass and the full line is the path of the actual centre.

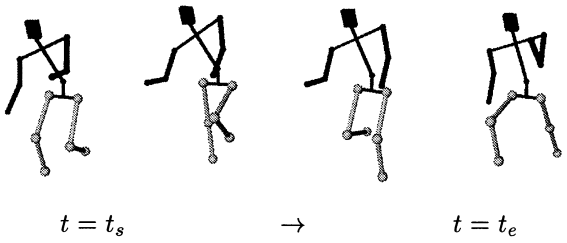


Figure 11. Balancing with a staggering leg

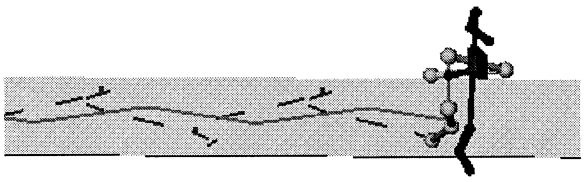


Figure 12. Trajectory path of centre of mass ($\eta = 5$, $\dot{x} = 2$)

Motion Optimization

Our control strategy is to exploit the genetic programming technique to find the optimized motion for a given specification of an animator. The motion of a character is expressed as a 'gene sequence'. Each gene consists of various parameters for a running model, such as l, $\dot{x}$, κ, θ_0, m, a, S_x, S_y, $\vec{L}$ and η.

In the case where a user defines the horizontal velocity ($\dot{x}$) of a character, the genes expressing the velocity are fixed in accordance with the specified velocity. Other parameters such as length of the legs and mass are set as predefined values. The other variable parameters are set randomly. As the genetic programming process evolves, the parameters are optimized to fit the motion that the animator wants. After the input phase the system generates a population of character models with various initial gene sequences for evolution. Each character executes a running motion according to its own gene, and the system evolves the motions of character to find better motions by repeated crossover and mutation. This simulated evolution process generates the optimal motion to match the animator's intention.

Gene Evaluation

We consider the case where the user specification is related only to the horizontal velocity. As mentioned before, if a user specifies the horizontal velocity of a character, the gene element which defines the velocity of each character is fixed and other parameters are created randomly. A large number of randomly created gene sequences show physically impossible motion. Such gene sequences are eliminated from the population.

Since running motion is cyclic, the motion needed for competition and evaluation is only from the initial standing phase to the beginning of the next standing phase. After each character executes its own motion, the system evaluates the motion. For evaluation we consider 'how naturally the standing phase and jumping phase are connected' and 'how much energy has been consumed'. To measure the connectivity and the energy consumed, we defined the degree of connectivity (motion smoothness) γ and the energy consumption rate E_r as

$$\gamma = \sum_{i}^{n} \left(\frac{\max\{\dot{\theta}_{ji}, \dot{\theta}_{si} + \ddot{\theta}_{si} h\}}{\min\{\dot{\theta}_{ji}, \dot{\theta}_{si} + \ddot{\theta}_{si} h\}} - 1 \right)^2$$

$$E_r = \frac{\int_{t_s}^{t_f} \Sigma_{i=1}^{n} |\ddot{\theta}_{it}|^2 I_i \, dt}{\int_{t_s}^{t_f} \dot{x}_t \, dt} = \frac{\Sigma_{t=t_s}^{t_f} \Sigma_{i=1}^{n} |\ddot{\theta}_{it}|^2 I_i}{x_{t_f} - x_{t_s}} \quad (13)$$

where $\dot{\theta}_{j_i}$ and $\dot{\theta}_{s_i}$ denote the angular velocities of the ith joint at the beginning of the jumping phase and the end of the standing phase respectively, $\ddot{\theta}_{s_i}$ is the angular acceleration at the end of the standing phase, t_s, used in the calculation of E_r, is the starting time of running, t_f is the ending time of running, $\ddot{\theta}_{i_t}$ is the angular acceleration of the ith joint at time t, and I_i is the moment of inertia of the ith joint. The connectivity γ is zero if the expected angular velocity of the ith joint at the beginning of the jumping phase ($\dot{\theta}_{s_i} + \ddot{\theta}_{s_i} h$) equals the actual angular velocity of the ith joint at that moment ($\dot{\theta}_{j_i}$). The energy consumption rate E_r is defined as the integration of energy from time t_s to t_f divided by the translation distance. We assumed that a natural motion is generated when these two values are minimized. Therefore we evaluate a model with γ and E_r using $eval(\gamma, E_r)$. If the value of the function $eval(\gamma, E_r)$ is small, the standing phase and the jumping phase are connected smoothly and the character consumes less energy. We drive the evolution of the character model to minimize the value of $eval(\gamma, E_r)$ using genetic programming. Here $eval(\gamma, E_r)$ is described as

$$eval(\gamma, E_r) = 1 - \frac{1}{2(1+\gamma)} - \frac{1}{2(1+E_r)} \quad (14)$$

The function $eval(\gamma, E_r)$ is minimized when γ and E_r are minimal. The value of $eval(\gamma, E_r)$ goes to one when the two values γ and E_r become infinite. After evaluation of a gene with this function, genetic programming selects the superior model, which runs more naturally with less energy, to produce the next generation of characters.

Emotion Control with an Energy Constraint

Minimizing the energy consumed and the angular velocity connectivity γ produces a natural-looking motion. We can control the motions of the character and express the emotions of the character by controlling the energy consumption rate. This idea is based on the fact that 'brisk motion' uses more energy than the optimized motion, and 'tired motion' can be generated by providing the least energy for movement.

Instead of minimizing the energy consumption, we can create emotions by controlling the energy consumption rate. To control the energy consumption rate, we modify the evaluation function as

$$eval(\gamma, E) = 1 - \frac{1}{2(1+\gamma)} - \frac{1}{2(1+E)}$$

$$E = [E_r - \beta/(1-\beta)]^2 \quad (15)$$

where β is a parameter for controlling the energy consumption rate. Note that $0 \leqslant \beta < 1$. E is the replacement for E_r in the previous $eval(\gamma, E_r)$ function. Our new evaluation function is needed to minimize the values of γ and E. If $\beta = 0$, E is minimized when $E_r = 0$. As β increases, E_r minimizing E also increases. Therefore, using this evaluation function, we can control the energy consumption rate of the character. If we want to make a character move using the optimal energy consumption rate, we must set $\beta = 0$. If we want to make a character move briskly, we must increase β to make the character move consuming more energy. However, it seems to be hard to generate the 'tired' motion only by minimizing the energy consumption. We think that some other physical or physiological effects should be considered to create a tired motion.

Experimental Results

We implemented this animation system using C++ and the Open Inventor library with SGI Indigo^2. Figure 13 and

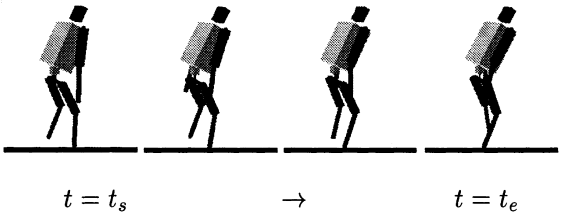


Figure 13. Tired walk

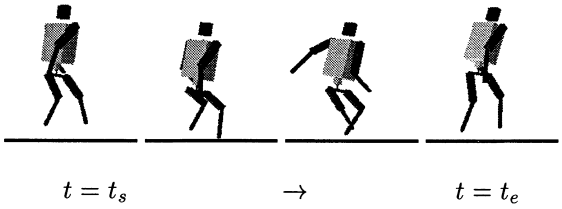


Figure 14. Brisk walk

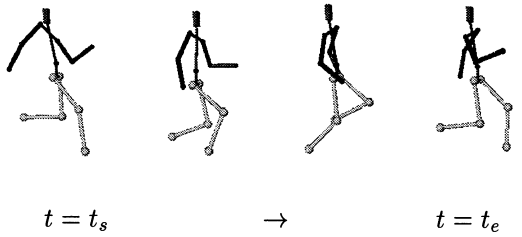


Figure 15. Optimized motion for a given speed $\dot{x}=2$ m s^{-1} ($\eta=5$, $\beta=0$)

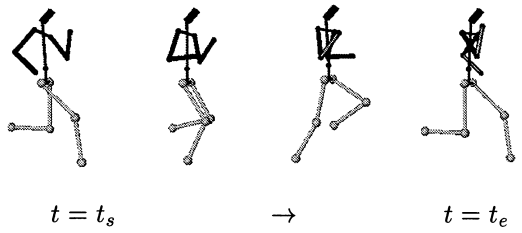


Figure 16. Optimized motion for a given speed $\dot{x}=5$ m s^{-1} ($\eta=5$, $\beta=0$)

14 show that the energy can be used as a control parameter. Figure 13 illustrates the result when a small amount of energy has been given to the character. Figure 14 illustrates the result when a large amount of energy has been given to the character. We can control the motion of a character to look tired (Figure 13) or brisk (Figure 14) by controlling the energy consumption.

If the speed parameter is used, we can generate the optimized motion for the specified speed with energy minimization. Figures 15 and 16 are optimized motions

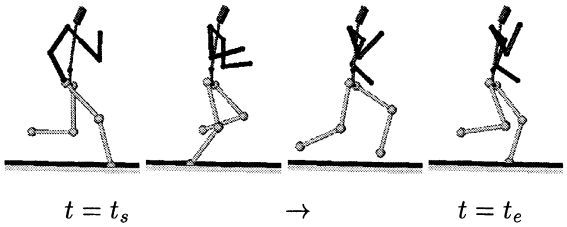


Figure 17. Normal running ($\beta=0.2$, $\dot{x}=10$ m s^{-1})

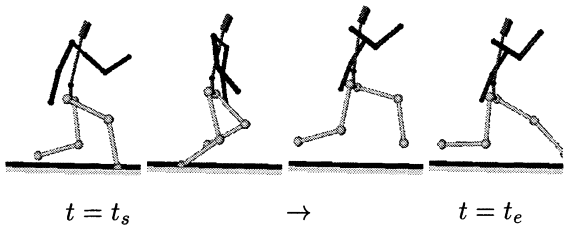


Figure 18. Brisk running ($\beta=0.7$, $\dot{x}=10$ m s^{-1})

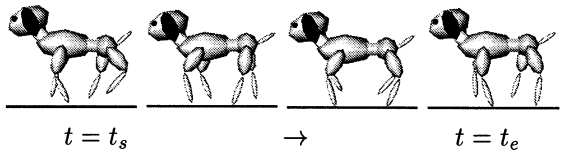


Figure 19. Normal running of a puppy model ($\beta=0.0$, $\dot{x}=3$ m s^{-1})

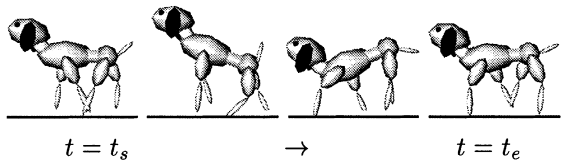


Figure 20. Brisk running of a puppy model ($\beta=0.9$, $\dot{x}=3$ m s^{-1})

obtained by minimizing the connectivity γ and the energy consumption rate E_r with velocities $\dot{x}=2$ and 5 m s^{-1} respectively.

Instead of minimizing the energy consumption rate, we can control the energy consumption rate of a character. By specifying a character to run at a certain speed and spend a large amount of energy, we obtain a brisk running motion at the specified speed. Figures 17 and 18 are snapshots of motions generated with the same velocity ($\dot{x}=10$ m s^{-1}) but different β values.

Figures 19 and 20 show the fact that our control approach can be easily applied to various characters. We slightly modified our human running model to implement

the running model of quadruped animals and applied the same control technique to the energy consumption of a character. Figure 19 shows the normal running of a puppy and Figure 20 shows the brisk running of a puppy.

Conclusion and Future Work

In this paper we have presented a new technique for creating human running animation with energy consumption rate control by applying a genetic programming paradigm. This technique enables animators to control human running motion with high-level commands and provides a plausible balancing technique for realistic motion. The motion of a character is determined thoroughly by its own gene sequence. By searching an optimal gene for minimizing the energy consumption rate, we can generate realistic motion. This technique can be extended to manipulate emotions or motions by parametrizing the energy consumption rate. Using this technique, we could create the motions of a character based on emotions, e.g. creating 'brisk running' by providing abundant energy for action.

This technique enables an animator to efficiently create running motion of human structures without detailed specifications for each joint. Our technique has the following advantages.

- It provides a high-level control over human running animation. An animator can easily generate running motion without any particular knowledge of the complicated dynamics of running.
- No motion data are needed for generating emotion-based motion.
- After some optimal genes are obtained for interesting behaviour in an off-line simulation, we can animate a creature in real time by making it move with the genes previously obtained. Also, by combining these genes, we can easily synthesize other motions.

Our technique does not sufficiently provide users with various methods to edit the motion of characters. Future work will include an extension that enables users to edit or constrain the motion of characters.

ACKNOWLEDGMENTS

This research was supported in part by the Electronics and Telecommunications Research Institute (ETRI). The authors would like to thank the reviewers for their constructive comments for the revision process.

References

1. J. K. Hahn, 'Realistic animation of rigid bodies', *Computer Graphics (Proceedings of SIGGRAPH*, **22,** 207–216 (1988).
2. A. Witkin and M. Kass, 'Spacetime constraints', *Computer Graphics (Proceedings of SIGGRAPH*, **22,** 159–168 (1988).
3. J. T. Ngo and J. Marks, 'Spacetime constraints revisited', *Computer Graphics (Proceedings of SIGGRAPH*, **27,** 343–350 (1993).
4. M. van de Panne and E. Fiume, 'Sensor–actuator networks', *Computer Graphics (Proceedings of SIGGRAPH*, **27,** 335–342 (1993).
5. A. Brudelin and T. W. Calvert, 'Goal-directed, dynamic animation of human walking', *Computer Graphics (Proceedings of SIGGRAPH*, **23,** 233–242 (1989).
6. J. K. Hodgins, W. L. Wooten, D. C. Brogan and J. F. O'Brien, 'Animating human athletics', *Computer Graphics (Proceedings of SIGGRAPH*, **29,** 8–14 (1995).
7. M. van de Panne, 'Parameterized gait synthesis', *IEEE Computer Graphics and Applications*, **16**(2), 40–49 (1996).
8. M. H. Raibert and J. K. Hodgins, 'Animation of dynamic legged locomotion', *Computer Graphics (Proceedings of SIGGRAPH*, **25,** 349–358 (1991).
9. M. Unuma, K. Anjyo and R. Takeuchi, 'Fourier principles for emotion-based human figure animation', *Computer Graphics (Proceedings of SIGGRAPH*, **29,** 91–96 (1995).
10. S. Brown and K. Passino, 'Intelligent control for an acrobat', *Journal of Intelligent and Robotic Systems*, **18,** 209–248 (1997).
11. A. D. Kuo, 'An optimal control model for analyzing human postural balance', *IEEE Transactions on Biomedical Engineering*, **BE-42,** 87–101 (1995).
12. R. Boulic and R. Mas, 'Hierarchical kinematic behaviors for complex articulated figures, in M. Thalmann and D. Thalmann (eds), *Interactive Computer Animation*, Prentice-Hall, Englewood Cliffs, NJ, 1996, pp. 40–70.
13. L. Gritz and J. K. Hahn, 'Genetic programming for articulated figure motion', *The Journal of Visualization and Computer Animation*, **6,** 129–142 (1988).
14. K. Sims, 'Evolving virtual creatures', *Computer Graphics (Proceedings of SIGGRAPH*, **28,** 15–22 (1994).
15. J. Ventrella, 'Disney meets Darwin: the evolution of funny animated figures', *Proceedings of Computer Animation '95*, 1995, pp. 35–43.
16. C. François and C. Samson, 'Running with constant energy', in *Proceedings of the IEEE International Conference on Robotics and Automation*, Vol. 1, IEEE, New York, 1994, pp. 131–136.

Authors' biographies

Young-Min Kang is a PhD student in the Department of Computer Science at Pusan National University. He received BSc and MSc degrees in computer science from Pusan National University in 1996 and 1999 respectively. His research interests include computer animation and simulation. He is currently working on simulation and animation of flexible objects.

Hwan-Gue Cho received a BSc degree in computer science and statistics from Seoul National University, Seoul, South Korea in 1984. He also received MSc and PhD degrees in computer science from KAIST (Korea Advanced Institute of Science and Technology) in 1986 and 1990 respectively. Since 1990 he has been with the Department of Computer Science, Pusan National University. In 1994 he was a visiting scholar at the Max Planck Institute for Informatics in Saarbrücken. His main research topics are algorithm design/analysis and computational geometry. Currently he works on flexible object animation and bioinformatics, especially phylogenetics.

Ee-Taek Lee received a PhD degree in electronics from KAIST (Korea Advanced Institute of Science and Technology) in 1996. He also received a BSc degree in engineering education and an MSc degree in electronics from Seoul National University in 1987 and 1980 respectively. Dr Lee has been a staff researcher at ETRI (Electronics and Telecommunications Research Institute) and is currently the chief executive of the VR research and development centre at ETRI. His most recent research focus includes virtual reality systems, computer animation, image processing/compression and biometric authentication.