Proc. of Pacific Graphics '98, pp. 169-176, 1998. Singapore, IEEE CS press

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

Young-Min Kang, Sun-Jin Park
and Hwan-Gue Cho
Graphics Application Lab
Dept. of Computer Science
Pusan National University
Keumjeong-Ku, Pusan, 609-735, Korea.
{ymkang, sjpark, hgcho}@pearl.cs.pusan.ac.kr

Ee-Taek Lee *
Visual Information Processing Section
Electronics and Telecommunication
Research Institute
etlee@etri.re.kr

Abstract

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, and 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-leg planar hopper with a self-balancing mechanism. The proposed technique exploits genetic programming to find an optimal movement. We extend the energy minimization technique to generate various motions in accordance with emotional specifications, for instance, "brisk running."

1. Introduction

Several methods have been introduced to create realistic motions for inanimate objects such as chains or articulated machines by simulating physical laws. However, animating life-like motions for humans or animals cannot be achieved by a simple and direct application of Newton's laws of motion, since the motions of such beings are affected by more than physical laws [3, 13].

A common approach to character animation is to simulate the dynamics of an articulated figure [17, 7, 10]. 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 behaviors. Another technique, SAN (Sensor Actuator Network) [15] exploits the neural network model. Although this approach has an advantage in that it requires no $a\ priori$ knowledge of complicated motion equations, animators cannot predict the results. Witkin and Kass introduced a Spacetime Constraints approach to create character animation with high-level control schemes such as "jump from here to there" [17]. One disadvantage of this kind of automation techniques is that they suffer from a lack of control over the detailed behavior of characters.

1.1 Motivation

In this work, we only consider the animation of running humans. Many techniques have been proposed to simulate the motions of the human skeletal structure [3, 4, 8, 14, 13]. The straight-forward 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 [11]. 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 [13]. 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 extrapolating the given motion data. One disadvantage is that it cannot create any motions if there is no motion data. So, we cannot apply this animation technique to virtual or imaginary crea-

*This research was supported in part by ETRI

tures, which often appear in cartoon-like situations. This paper describes a new approach to human running animation without any motion data.

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 [2, 9]. In the field of computer graphics, Boulic and Mas implemented the balancing motion using inverse kinematics [1]. However, these methods need 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, so the energy consumption can be a good parameter for animation. In order to find an optimized running dynamics, we use the genetic programming approach, where each character model has its own 'gene' sequences. Many works have shown the good performance of genetic programming approach to animation of virtual characters [6, 12, 16].

2. System overview

Our character model is shown in Figure 1. This model has 11 joints with various degrees of freedom.

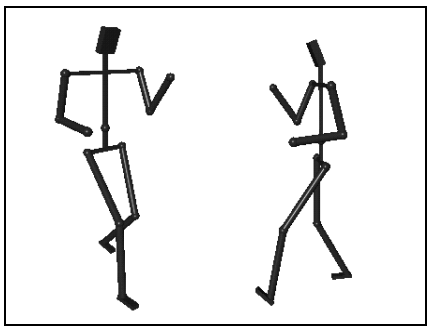

Figure 1. Character model

The user interface of our system is shown in Figure 2. For the modeling of the running motion, we first developed a one-leg planar hopper, which has been studied extensively in robotics [5], then extended this one-leg hopper to a two-leg planar hopper by adding more parameters. The realistic human running motion requires not only realistic movements of legs but also balancing mechanism to support the upper body.

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 generates a population of character models and forces the characters to compete with each other. Each character model has its own gene (parameters for running model) and runs forward with a given energy. Assuming that we prefer a character that runs faster but expends little energy, the system evaluates the performance of each character based on its speed and 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. So, fitter models, which consume smaller amounts of energy, would survive to the next generation.

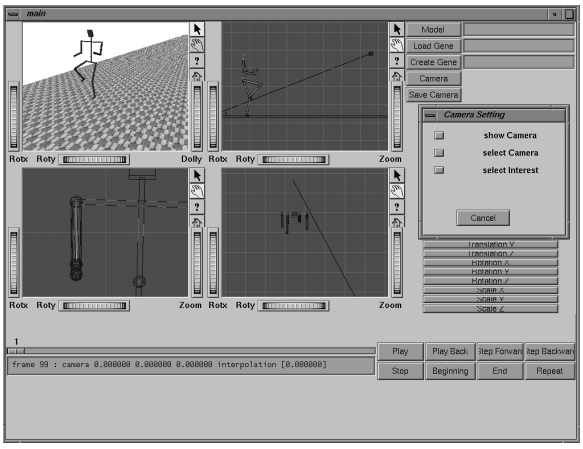

Figure 2. System interface

3. The running model

3.1. A one-leg hopper

Let us consider a one-leg hopper shown in Figure 3 as a basic model for human running. The initial state of the one-leg hopper is shown in Figure 3. This hopper is 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 the length of the leg is shorter than the initial length. HIP denotes the hip location of this hopper, and $FOOT$ denotes the the location of the foot. The position of HIP and $FOOT$ is 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 location of hip and foot at the initial state (HIP_0 and $FOOT_0$) are given in the following:

$$\begin{aligned} HIP_0 &= (-l \cdot \sin\theta_0, l \cdot \cos\theta_0) \\ FOOT_0 &= (0, 0) \end{aligned} \quad (1)$$

This running model consists of two alternating phases, a standing phase and a jumping phase. The standing phase

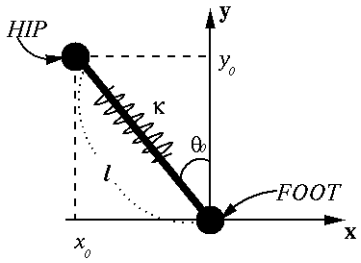

Figure 3. Initial state of a one-leg hopper

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 ($f_y > 0$). Running of one-leg hopper can be described by alternately repeating these two phases.

3.1.1 Standing phase of a one-leg hopper

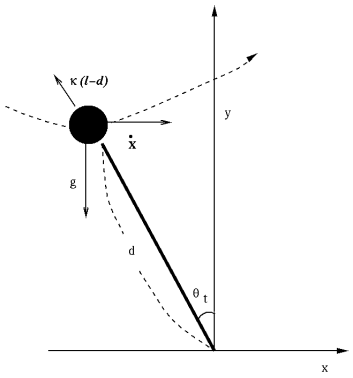

Figure 4. Key elements determining the standing phase

Figure 4 shows four key elements that affect the trajectory of a hip during the standing phase (gravity g, horizontal velocity $\dot{x}$, spring constant κ, and the angle between leg and y axis θ). We assumed that the horizontal velocity $\dot{x}$ is constant in running, and 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 location of the hip (x_t, y_t) at time t is described as follows:

$$\begin{aligned} x_t &= x_{t-h} + \dot{x} \cdot h \\ y_t &= g \cdot h^2 + \frac{\kappa(l-d) \cdot h^2 \cdot \cos\theta_t}{m} + 2 \cdot y_{t-h} - y_{t-2h} \end{aligned} \quad (2)$$

where θ_t is the angle between the leg and the y axis at time t.

The hopper should be in the standing phase if the length of leg 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 movements of the hip of the hopper is determined only by gravity. The next section will explain how to calculate T.

3.1.2 Jumping phase of a one-leg hopper

During a jumping phase, the location of the hip can be calculated as follows:

$$\begin{aligned} HIP_t &= (x_t, y_t) \\ &= (x_{t-h} + \dot{x} \cdot h,\ y_0 + \dot{y}_0 \cdot t + \frac{g \cdot t^2}{2}) \end{aligned} \quad (3)$$

where $\dot{y}_0$ denotes the initial velocity of the hip in a jumping phase, and y_0 denotes the y coordinate 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. So, y_f, the height at the completion time of jumping is $l \cdot \cos\theta_0$, and the jumping time T is t that satisfies the equation $y_0 + \dot{y}_0 t + g \cdot t^2/2 = y_f$. Thus, T can be calculated as follows:

$$\begin{aligned} T &= \frac{-\dot{y}_0 - \sqrt{\dot{y}_0^2 + 2 \cdot g \cdot c}}{g} \\ &, where\ \ c = y_f - y_0 \end{aligned} \quad (4)$$

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. Figures 6 and 7 show the running animation of the one-leg hopper model. Our character model has a joint between the hip and the foot, as shown in Figure 5 to keep the length of the leg $|HIP - JOINT| + |JOINT - FOOT|$ constant. The angle between the vector $HIP - JOINT$ and $HIP - FOOT$ is denoted as θ_l, and we can easily compute the value as follows:

$$\theta_l = cos^{-1}(d/l) \quad (5)$$

3.2. A two-leg hopper

The technique for the one-leg hopper can be extended to fit a two-leg hopper. As shown in Figure 8, a two-leg

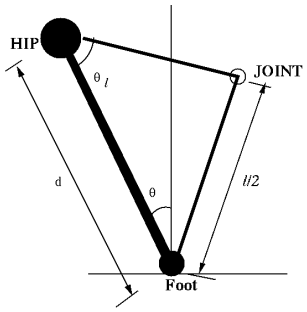

Figure 5. A One-leg hopper with a joint between hip and foot

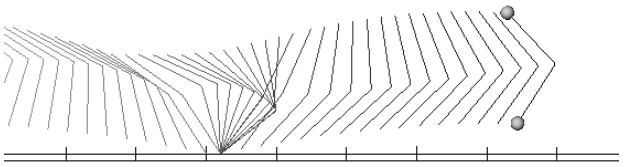

Figure 6. Motion of one-leg hopper 1 ($h = 0.005sec,\ l = 2m,\ \dot{x} = 10m/sec,\ \kappa = 3000,\ \theta_0 = \pi/4,\ m = 50kg$**)**

hopper requires additional parameter $\vec{L}$ for specifying an initial state. Note that $\vec{L}$ is defined as $FOOT - HIP$.

3.2.1 Standing phase of two-leg hopper

The running of a two-leg 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 the one-leg hopper. Let $FOOT_a$ denote the lifted foot. The location of $FOOT_a$ (f_{x_a}, f_{y_a}) is calculated as follows:

$$\begin{aligned} \dot{f}_{x_a} &= \dot{x} \cdot S_x \\ \dot{f}_{y_a} &= \dot{y} \cdot S_y \end{aligned} \tag{6}$$

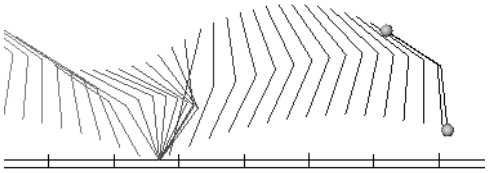

Figure 7. Motion of one-leg hopper 2 ($h = 0.01sec,\ l = 2m,\ \dot{x} = 5m/sec,\ \kappa = 3000,\ \theta_0 = \pi/4,\ m = 50kg$**)**

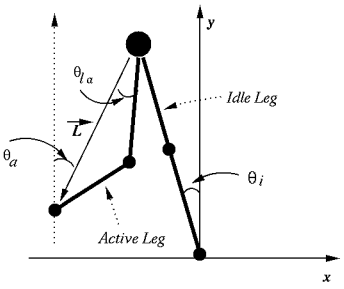

Figure 8. The initial state of a two-leg hopper

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

3.2.2 Jumping phase of a two-Leg hopper

In the jumping phase, the angles of 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:

$$\begin{aligned} \theta_{l_t} &= \theta_{l_0} + (\theta_{l_T} - \theta_{l_0}) \cdot B_l(t) \\ \theta_{c_t} &= \theta_{c_0} + (\theta_{c_T} - \theta_{c_0}) \cdot B_c(t) \end{aligned} \tag{7}$$

where θ_{l_t} denotes the angle between the vector $(JOINT - HIP)$ and $(FOOT - HIP)$, and θ_{c_t} is the angle between $(HIP - FOOT)$ and y axis at time t. θ_{l_0} and θ_{c_0} denote the angles at time 0, the beginning of jumping. θ_{l_T} and θ_{c_T} denote 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 0 when $t = 0$, and 1 when $t = T$. $B_l(t)$ is defined as $B_l(t) = t/T$ to interpolate linearly θ_{l_0} and θ_{l_T} during time T. $B_c(t)$ is defined as follows:

$$B_c(t) = \left(\frac{1}{T^2 \cdot (1-2\alpha)}\right) t^2 + \left(\frac{-2\alpha}{T + (1-2\alpha)}\right) t \tag{8}$$

where α is a parameter that controls the maximum (if $\alpha > 0.5$) or minimum (if $\alpha < 0.5$) value of $B_c(t)$ to be $\alpha \cdot T$. If α is 0.9, the difference between θ_{c_t} and θ_{c_0} becomes maximum when $t = 0.9 \cdot T$. If we set α to 1, $B_c(t)$ increases monotonically and keeps $|\theta_{c_t} - \theta_{c_0}|$ greater than $|\theta_{c_f} - \theta_{c_0}|$ in the whole jumping phase. So, this parameter determines

the movements of the legs in the jumping phase. The whole motion of the two-leg hopper is determined by control parameters which include l, $\dot{x}$, κ, θ_0, m, α, S_x, S_y, and $\vec{L}$. Figure 9 shows the running animation of the two-leg hopper.

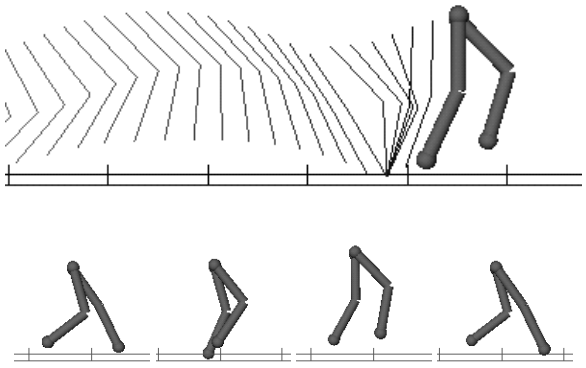

Figure 9. Movement of the two-leg hopper ($h = 0.01sec$, $l = 1.5m$, $\dot{x} = 5m/sec$, $\kappa = 9000$, $\theta_0 = \pi/6$, $m = 50kg$, $\alpha = 0.9$, $S_x = 1.5$, $S_y = 1.0$**)**

3.3. Balancing the upper body

In order to animate natural human running, it is important to make a balanced posture according to the movements of 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 center of mass and the current center of mass. The target center of mass is the location where the center of mass of character should be located, and the current center of mass is the actual center of mass of the character. The target center of mass is assumed to be the foot placed on the ground in the standing phase. In the jumping phase, the target center of mass is calculated by interpolating the two feet in accordance with time t. The target center of mass can be calculated as follows:

$$\begin{aligned} CM_{tar} &= FOOT_i, \quad (9) \\ &\quad if\ mode\ is\ STANDING \\ CM_{tar} &= (FOOT_i) + (FOOT_a - FOOT_i) \cdot \frac{t}{T}, \\ &\quad if\ mode\ is\ JUMPING \end{aligned}$$

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

$$CM_{cur} = \frac{\sum_i^n M_i \cdot L_i}{\sum_i^n M_i} \quad (10)$$

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

$$\vec{\delta} = CM_{tar} - CM_{cur} \quad (11)$$

δ_x and δ_z denote the x axis and the z axis components of $\vec{\delta}$, respectively. When δ_x is positive, i.e. target center of mass is in front of the current center of mass in the x axis direction, the body must bend forward to make a stable posture. In other words, a body needs more rotation about the z axis. Let zd_b denote the rotation about the z axis to compensate the error, and the degree of rotation of the body about the z axis, Z_b, should be as follows:

$$Z_{b_t} = Z_{b_{t-h}} + \frac{\dot{Z}_{b_{t-h}} \cdot h + \ddot{Z}_{b_{t-h}} \cdot h^2 + zd_{b_{t-h}}}{2} \quad (12)$$

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

$$zd_b = \eta \cdot \frac{\delta_x \cdot h}{\dot{x}} \quad (13)$$

where h is the time interval between two consecutive frames, $\dot{x}$ is horizontal velocity, and parameter η controls the sensitivity about the error. If η increases, the character becomes more sensitive to the error, and 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 $xd_{b_t} = \eta \cdot (-\delta_z) \cdot h/\dot{x}$ to makes the body rotate about the x axis. Figures 10 and 11 show the results of our balancing technique with η=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 walking. In the staggering motion, a character spends shorter time when a staggering leg is supporting the whole body, compared to when the other leg is supporting. The result is shown in Figure 12. We can see that the center of mass is biased to the leg that is not staggering. Figure 13 shows the trajectory path along the target center of mass and the current center of mass. The dash line shows the trajectory of the target center of mass and the solid line is the path of the actual center.

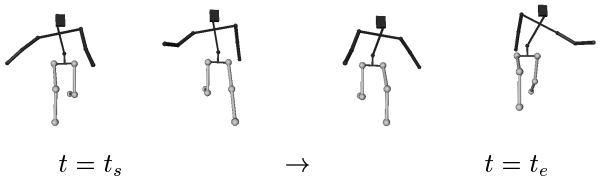

Figure 10. Balancing motion ($\eta = 5$ $\dot{x} = 1$)

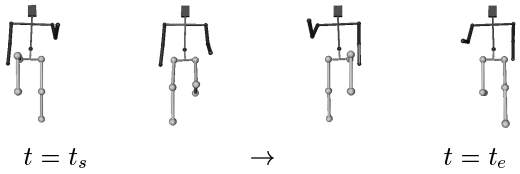

Figure 11. Balancing motion ($\eta = 5$ $\dot{x} = 5$)

4. Motion optimization

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

In case that a user defines the horizontal velocity ($\dot{x}$) of a character, the genes expressing the velocity are fixed in accordance with the specified velocity, and other parameters such as length of legs and mass are set as a predefined value. 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 characters to find better motions by repeating crossover and mutation. This simulated evolution process generates an optimal motion to match an animator's intention.

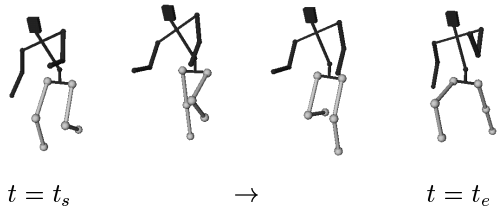

Figure 12. Balancing with a staggering leg

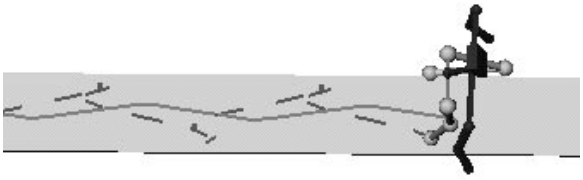

Figure 13. Trajectory path of the center of mass $\eta = 5$ $\dot{x} = 2$

4.1. Gene evaluation

we consider the case that user specification is related only to horizontal velocity. As mentioned before, if a user specifies the horizontal velocity of a character, the gene element defining the velocity of each character is fixed, and other parameters are created randomly. A great number of randomly created gene sequences show physically impossible motion, and such gene sequences are eliminated.

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 energy consumed, we defined the degree of connectivity (= motion smoothness) γ, and energy consumption rate E_r as follows:

$$\begin{aligned} \gamma &= \sum_{i}^{n} \left(\frac{\max\{\dot{\theta}_{j_i}, \dot{\theta}_{s_i} + \ddot{\theta}_{s_i}\}}{\min\{\dot{\theta}_{j_i}, \dot{\theta}_{s_i} + \ddot{\theta}_{s_i}\}} - 1 \right)^2 \\ E_r &= \frac{\int_{t_s}^{t_f} \sum_{i=1}^{n} |\ddot{\theta}_{i_t}|^2 I_i \, dt}{\int_{t_s}^{t_f} \dot{x}_t dt} \\ &= \frac{\sum_{t=t_s}^{t_f} \sum_{i=1}^{n} |\ddot{\theta}_{i_t}|^2 I_i}{x_{t_f} - x_{t_s}} \end{aligned} \tag{14}$$

where $\dot{\theta}_{j_i}$ and $\dot{\theta}_{s_i}$ denote angular velocity of the i-th joint at the beginning of the jumping phase and ending time 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 energy consumption rate E_r is the starting time of running. t_f is the ending time of running, and $\ddot{\theta}_{i_t}$ is an angular acceleration of the i-th joint at time t. I_i is the moment of inertia of the i-th joint. The connectivity γ is 0 if expected angular velocity of the i-th joint at the beginning of the jumping phase ($\dot{\theta}_{s_i} + \ddot{\theta}_{s_i}$) equals the actual angular velocity of the

i-th joint at the 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 translation distance. We assumed that a natural motion is generated when these two values are minimized. So, 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 smaller, the standing phase and the jumping phase are connected smoothly and consumes less energy. We drive the evolution of the character model to minimize the value of $eval(\gamma, E_r)$ using genetic programming. $eval(\gamma, E_r)$ is described as follows:

$$eval(\gamma, E_r) = 1 - \frac{1}{2(1+\gamma)} - \frac{1}{2(1+E_r)} \qquad (15)$$

Function $eval(\gamma, E_r)$ is minimized when γ and E_r are minimal, and $eval(\gamma, E_r)$ goes to 1 when these 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 characters.

4.2. Emotion control with an energy constraint

Minimizing the energy consumed and angular velocity connectivity γ produces a natural-looking motion. We can control the motions of 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 follows:

$$eval(\gamma, E) = 1 - \frac{1}{2(1+\gamma)} - \frac{1}{2(1+E)}$$
$$, where\ E = \left(\frac{\beta}{1-\beta} - E_r\right)^2 \qquad (16)$$

where β is a parameter for controlling the energy consumption rate. Note that $0 \leq \beta < 1$. E is the substitute 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 β is 0, E is minimized when E_r = 0. As β increases, E_r that minimizes the E also increases. Therefore, using this evaluate function, we can control the the energy consumption rate of the character. If we want to make a character move using the optimal energy consumption rate, we must set β = 0, and if we want to make a character move briskly, we must increase β to make the character move using more energy.

5. Experimental results

We implemented this animation system using C++ and Open Inventor library with SGI $Indigo^2$. Figures 14, 15 and 16 are optimized motions obtained by minimizing the connectivity γ and their energy consumption rate E_r with velocity $\dot{x} = 2m/sec$, $\dot{x} = 5m/sec$, and $\dot{x} = 7m/sec$, respectively.

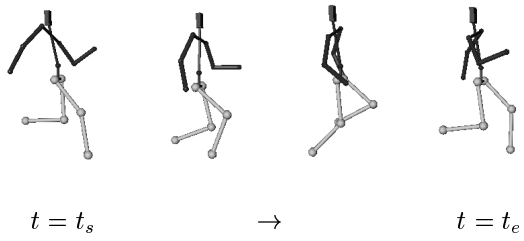

Figure 14. Optimized motion for a given speed $\dot{x} = 2m/sec$ ($\eta = 5, \quad \beta = 0$)

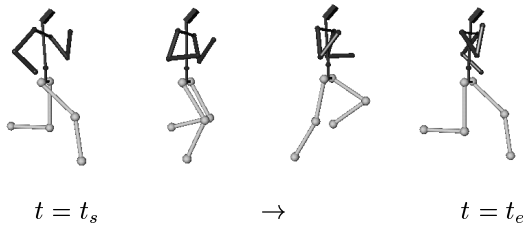

Figure 15. Optimized motion for a given speed $\dot{x} = 5m/sec$ ($\eta = 5, \quad \beta = 0$)

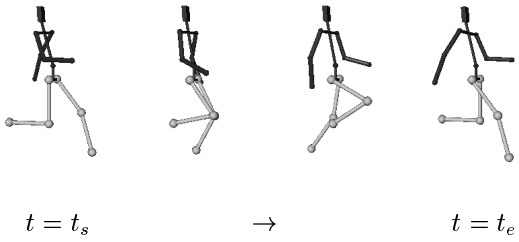

Figure 16. Optimized motion for a given speed $\dot{x} = 7m/sec$ ($\eta = 5, \quad \beta = 0$)

Figures 17 and 18 are snapshots of motions generated with the same velocity ($\dot{x} = 10m/sec$) but different β values.

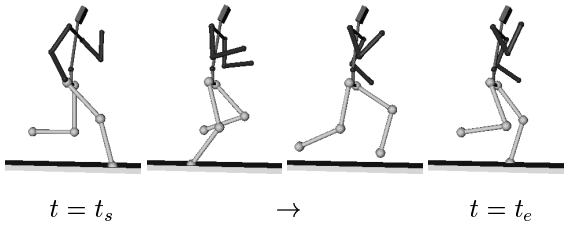

Figure 17. Normal running ($\beta = 0.2 \quad \dot{x} = 10ms^{-1}$)

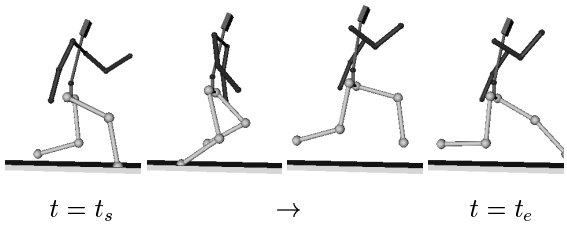

Figure 18. Brisk running ($\beta = 0.7 \quad \dot{x} = 10ms^{-1}$)

6. Conclusion and future work

In this paper we present 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 the 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 optimal gene for minimizing the energy consumption rate, we can generate realistic motion. This technique can be extended to manipulate emotions of motions by parameterizing the energy consumption rate. Using this technique, we could create the motions of a character based on emotions, for example, 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 dynamics. Our technique has the following advantages:

- Our technique provides a high-level control over human running animation. An animator can easily generate running motion without any particular knowledge of complicated dynamics of running.
- No motion data is needed for generating emotion-based motion.
- After some optimal genes are obtained for interesting behavior 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.

In this paper, we only considered the running motion of an articulated human model. In the future, we will develop a general running model that will be applicable to animals or virtual creatures.

References

[1] R. Boulic and R. Mas. Hierarchical kinematic behaviors for complex articulated figures. In M. Thalmann and D. Thalmann, editors, *Interactive Computer Animation*, chapter 3, pages 40–70. Prentice Hall, 1996.

[2] S. Brown and K. Passino. Intelligent control for an acrobat. *Journal of Intelligent and Robotic Systems*, 18:209–248, 1997.

[3] A. Bruderlin and T. W. Calvert. Goal-directed, dynamic animation of human walking. *Proceedings of SIGGRAPH '89*, 23:233–242, Jul. 1989.

[4] D. J. Densley and P. J. Willis. Emotional posturing: A method towards achieving emotional figure animation. *Proceedings of Computer Animation '97*, pages 8–14, Jun. 1997.

[5] C. François and C. Samson. Running with constant energy. *Proceedings of the IEEE International Conference on Robotics and Automation*, 1:131–136, May. 1994.

[6] L. Gritz and J. K. Hahn. Genetic programming for articulated figure motion. *The Journal of Visualization and Computer Animation*, 6:129–142, 1988.

[7] J. K. Hahn. Realistic animation of rigid bodies. *Proceedings of SIGGRAPH '88*, 22:207–216, Aug. 1988.

[8] J. K. Hodgins, W. L. Wooten, D. C. Brogan, and J. F. O'Brien. Animating human athletics. *Proceedings of SIGGRAPH '95*, pages 8–14, 1995.

[9] A. D. Kuo. An optimal control model for analyzing human postural balance. *IEEE Transaction On Biomedical Engineering*, 42(1):87–101, 1995.

[10] J. T. Ngo and J. Marks. Spacetime constraints revisited. *Proceedings of SIGGRAPH '93*, pages 343–350, Aug. 1993.

[11] M. H. Raibert and J. K. Hodgins. Animation of dynamic legged locomotion. *Proceedings of SIGGRAPH '91*, 25(4):349–358, 1991.

[12] K. Sims. Evolving virtual creatures. *Proceedings of SIGGRAPH '94*, pages 15–22, Jul. 1994.

[13] M. Unuma, K. Anjyo, and R. Takeuchi. Fourier principles for emotion-based human figure animation. *Proceedings of SIGGRAPH '95*, pages 91–96, Aug. 1995.

[14] M. van de Panne. Parameterized gait synthesis. *IEEE Computer Graphics and Applications*, 16(2):40–49, 1996.

[15] M. van de Panne and E. Fiume. Sensor-actuator networks. *Proceedings of SIGGRAPH '93*, pages 335–342, Aug. 1993.

[16] J. Ventrella. Disney meets darwin : The evolution of funny animated figures. *Proceedings of Computer Animation '95*, pages 35–43, 1995.

[17] A. Witkin and M. Kass. Spacetime constraints. *Proceedings of SIGGRAPH '88*, 22:159–168, Aug. 1988.