한 발 뛰뛰기 모델을 이용한 사실적인 달리기 애니메이션

A Realistic Running Animation with One-Leg Hopping Model

강영민, 박선진, 조환규
그래픽스 응용 연구실
부산대학교 자연과학대학 전자계산학과
부산시 금정구 장전동 산 30
phone : 051-582-5009 fax : 051-515-2208
{*ymkang,sjpark,hgcho*}*@pearl.cs.pusan.ac.kr*

Abstract

캐릭터 애니메이션의 가장 중요한 목표는 효율적인 방법으로 캐릭터를 제어하는 것이다. 지금까지 인간의 보행 동작을 생성하기 위한 많은 기법들이 제안되었는데, 최근에는 파라미터 보간, 동작 데이터 매핑등을 이용하여 "활발한 보행"이나 "지친 걸음"과 같은 감정 표현을 이용하여 동작을 생성하는 기법들이 연구되고 있다. 이러한 감정 표현은 자동화하기가 매우 어렵기 때문에 지금까지는 애니메이터의 창의적인 감각에 의해 생성되어 왔다. 본 논문은 하나의 다리를 가진 평면 호퍼 (planar hopper)에 기반한 인간의 주행 동작을 모델링하여 동작을 제어하는 기법과 현실적인 애니메이션을 위해 주행시 캐릭터가 균형을 유지하도록 하는 기법을 제안한다. 캐릭터의 동작은 이 주행 동작 모델과 균형잡기 모델에 적용되는 파라미터들에 의해 결정되는데, 이 파라미터들의 집합을 유전자로 표현하여 애니메이터가 지정한 요구조건을 만족하면서 가장 자연스러운 동작을 생성하는 기법도 다룬다. 자연스러운 동작은 동작중에 소모되는 에너지를 최적화함으로써 구할 수 있다는 기본 가정에서 에너지를 최적화하는 기법을 설명하고 "활발한 동작"과 같이 애니메이터가 원하는 감정적 상태로 캐릭터가 동작하도록 하는 새로운 에너지 제한조건 제어 기법도 설명된다.

Keywords: 캐릭터 애니메이션, 주행 모델, 균형 잡기, 유전자 프로그래밍, 에너지 제한조건.

1 서론

물리적 법칙을 적용하여 무생물의 움직임을 사실적으로 묘사하는 기법들은 많이 제안되었다. 그러나, 인간이나 동물과 같이 자율적인 의지에 의해 운동하는 객체의 동작을 생성하는 것은 뉴턴 물리 법칙을 단순히 적용시키는 것만으로는 이루기 힘들다.

컴퓨터 애니메이션의 중요한 목표는 캐릭터가 애니메이터의 의도에 정확히 일치하는 동작을 하도록 하는 것과 그 동작을 효율적인 방법으로 생성하는 것이다.

자동화를 이용한 동작 생성 기법 가운데 일반적인 방법은 역학적 시물레이션을 이용하여 캐릭터의 움직임과 물리적 법칙이 일치하도록 하는 방법이 있다. 이러한 방법은 생성된 동작이 매우 사실적이라는 장점이 있지만 많은 계산을 필요로 하고 자발적인 행위를 표현하는데는 적합하지 못하다는 단점을 가지고 있다. 다른 접근 방법으로는 신경 회로망을 이용한 "Sensor Actuator Network"을 사용하여 캐릭터의 동작을 생성하는 기법이 있다 [7]. 이 방법은 캐릭터의 동작을 표현하기 위한 수학적 도구 없이도 쉽게 동작을 생성할 수 있다는 장점이 있지만, 애니메이터가 그 동작을 예측할 수 없다는 큰 단점을 가지고 있다. 최근에는 Spacetime Constraints 기법이 제시되어 캐릭터가 따라야할 동작을 개념적으로 명시함으로써 애니메이션을 생성할 수 있게 되었다 [6, 11]. 그러나 이 기법은 동작에 대한 초기 추측에 따라 생성된 애니메이션의 질이 달라질 수 있다는 단점과 함께 애니메이션 생성을 위해 최적화되어야 하는 기준이 반드시 에너지라는 것을 장담할 수 없다는 한계를 가지고 있다. 이러한 모든 자동화 기법들은 캐릭터의 동작을 충분히 제어할 수 없다는 한계를 가지고 있다.

1.1 연구 배경

본 논문에서는 인간의 주행 동작만을 다룬다. 현재까지 많은 애니메이션 기법들이 제안되어 인간 골격 모델의 애니메이션을 생성하는데 이용되었다 [3, 4, 7]. 가장 단순한 방법인 모션 캡쳐는 기계적 장비를 이용하여 실제 동작 데이터를 얻어 이를 골격 모델에 매핑하는 것이다. 이러한 방법은 고가의 하드웨어 장비를 필요로 하며, 가상 생물의 동작이나 만화적으로 과장된 동작은 생성할 수 없다는 단점을 가지고 있다. 다른 방법으로, 동작 특성 파라미터를 보간(interpolation)하거나 외삽(extrapolation)하는 기법이 있다[10]. 예를 들어 지친 걸음과, 활발한 걸음의 두 가지 보행 동작 데이터를 가지고 있는 경우 더욱 지친 걸음걸이나 더욱 활발한 걸음은 가지고 있는 보행 데이터를 외삽(extrapolation)하여 구할 수 있다. 이 방법은 간단하다는 장점이 있지만, 동작

데이터가 없는 상태에서는 애니메이션을 생성할 수 없다는 한계를 가진다. 본 논문은 모션 캡쳐등에 의해 생성된 어떤 데이터도 없이 다양한 파라미터를 통해 효율적으로 인간의 주행 동작을 생성하고 이를 쉽게 제어할 수 있는 방법에 대한 연구의 결과이다.

또한 현실적인 동작을 생성하기 위해서는 캐릭터가 균형을 유지하려고 하는 동작을 표현해야만 한다. 균형 유지의 근본은 무게 중심을 지지하는 다리 위에 수직으로 위치시키는 것으로, 이는 로보틱스 분야에서 집중적으로 연구되고 있는 분야이며 [2, 5], 그래픽스 분야에서는 Boulic과 Mas가 캐릭터의 균형잡기를 역 운동학(inverse kinematics)의 서브태스크(subtask)로 사용하여 구현하는 방법을 연구하였다 [1]. 이러한 연구들은 주로 동작이 정적일 때나 지지하는 다리의 위치가 변하지 않는 상황에 관한 연구들이다. 본 논문에서는 인간 주행과 같이 지지하는 다리의 위치가 지속적으로 바뀌는 상황에서 효과적으로 사용할 수 있는 효율적인 균형잡기 동작 표현 방법을 제시한다.

2 시스템 개요

2.1 캐릭터 모델의 구조

본 논문의 실험을 위해 그림 1과 같은 캐릭터 모델을 사용하였다. 이 모델은 다양한 자유도를 가지는 11개의 관절로 이루어져 있다. 인체 모델을 정확히 표현하기 위해서는 더욱 많은 관절이 필요하지만 본 논문 기법은 인간의 주행만을 다루고 있으므로 이를 위해 꼭 필요한 관절들로 인간 모델을 단순화하였다.

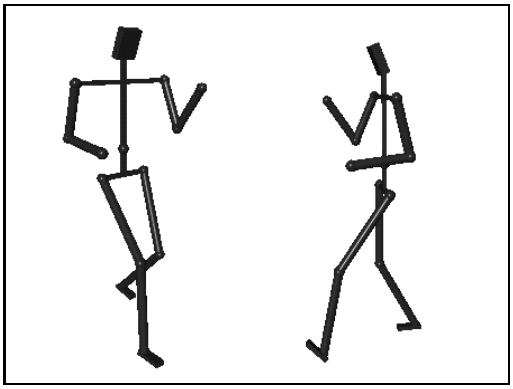

Figure 1: 11개의 관절을 가진 캐릭터 모델

2.2 애니메이션 기법 개요

본 논문에서 제시하는 기법을 구현한 시스템의 사용자 인터페이스는 그림 2과 같다. 본 논문의 기법은 사실적인 인간의 주행 동작을 생성하기 위해 몇개의 파라미터로 인간의 주행을 표현할 수 있는 모델을 구현하였다. 이 모델을 구현하기 위해 우선 하나의 다리를 가진 평면 호퍼 (planar hopper) 모델을 구현하였고 이러한 동작 모델을 인간의 다리에 적용할 수 있도록 두 개의 다리를 고려하여 몇가지 파라미터를 추가함으로써 두 다리를 가진 평면 호퍼로 확장하였다. 이 모델을 기반으로 인간 하체의 주행시 동작을 생성하였고, 상체의 자연스러운 움직임을 위하여 무게 중심이 다리에 의해 지탱될 수 있도록 하는 균형 모델을 구현하였다. 상체와 하체의 움직임을 표현하는 이러한 모델들은 모두 파라미터에 의해 동작이 결정된다. 이 기본 동작 모델은 사용자의 목적에 따라 제어될 수 있어야 하는데, 주행 동작 파라미터를 직접 다루는 방식은 불편할 뿐 아니라, 예측가능한 제어가 쉽지 않다. 이를 해결하기 위해 파라미터들의 값의 집합을 유전자 형태로 표현하였으며 주어진 조건에 대해 최적의 동작을 생성하기 위해 유전자 프로그래밍을 이용한 모의 진화를 사용하여 적합한 유전자, 즉, 파라미터 집합을 생성하였다. 이러한 유전자 프로그래밍은 가상 캐릭터의 동작을 생성하기 위한 최적화의 도구로 이미 많이 사용된 것이다 [3, 8, 9].

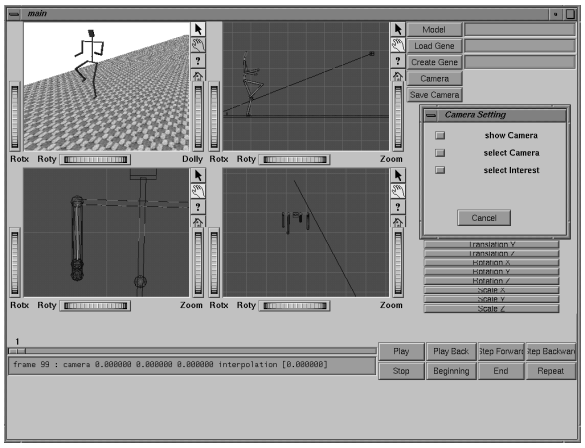

Figure 2: 시스템 인터페이스

동작을 생성하기 위해 시스템은 다수의 캐릭터 모델을 생성한다. 각 캐릭터 모델은 자신만은 파라미터를 가지고 있고, 이 파라미터를 기본 주행 모델에 적용하여 다양한 동작을 취할 수 있다. 이러한 주행 파라미터들은 유전자의 형태로 저장되어 있다. 시스템은 각 객체들을 기준에 따라 평가하여 순위를 매긴다. 이렇게 평가된 캐릭터의 유전자는 평가 결과에 따라 교배나 돌연변이를 통해 새로운 유전자를 만들게 된다. 이러한 모의 진화는 사용자의 요구에 더욱 적합한 동작이 생성되도록 유전자들을 변화시킨다. 이런 방법으로 최적의 유전자가 얻어지면 애니메이션 시스템이 최적의 유전자를 사용하여 객체의 동작을 생성한다.

3 동작 모델

3.1 한 다리 호퍼 (One-Leg Hopper)

인간 주행을 모델링하기 위해 그림 3와 같이 하나의 다리로 구성된 호퍼를 모델링한다. 그림 3에 나타난 x축이

다리가 놓여지는 지면의 역할을 하며 호퍼는 이 x축 양의 방향으로 이동한다. 호퍼의 다리 길이는 변할 수 있으며, 다리의 길이가 원래의 값보다 줄어들면 다리는 스프링의 역할을 하게 된다. 이 호퍼의 상단 부분의 위치를 HIP이라 하고 (x, y)의 2차원 벡터형태로 표현된다. 발 부분의 위치는 $FOOT$로 나타내고 이 값도 역시 (f_x, f_y) 형태의 2차원 벡터이다. 호퍼의 주행을 결정하는 파라미터는 다리의 스프링 상수(κ), 질량(m), 다리의 길이(l), 착지 각도(θ_0), 수평 이동 속도($\dot{x}$)가 사용된다.

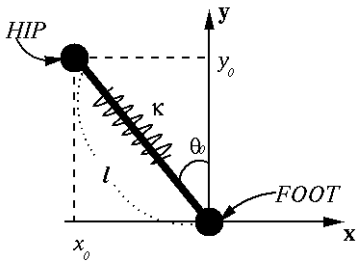

Figure 3: 하나의 다리를 가진 호퍼의 초기 상태

주행 시작 시간에 호퍼의 위치는 다음과 같다.

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

이 한 다리로 움직이는 호퍼의 주행을 모델링하기 위해 주행 상태를 두가지로 나눈다. 첫 번째 상태는 발이 땅에 닿아 있는 착지 상태($FOOT_y = 0$)이며, 두 번째 상태는 공중에 떠 있는 도약 상태이다. 이 두가지 상태를 번갈아 반복하면 호퍼의 주행 동작이 나타나게 된다.

3.1.1 한 다리를 가진 호퍼의 착지 상태

그림 4는 착지 상태동안 HIP의 위치에 영향을 주는 요소들을 나타낸다. 단순화를 위해 착지 상태에서 호퍼의 x축 속도는 그대로 유지된다. y축 속도는 중력 가속도에 의한 낙하 운동과 길이 d로 압축된 다리의 탄성에 의한 운동의 합으로 나타난다. 따라서, 프레임간의 시간 간격을 h라고 할 때, 각 시간 t에서의 HIP의 위치는 다음과 같다.

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

이 착지 상태는 d가 l보다 작은 동안 지속된다. d가 l보다 커져서 상태가 바뀔 때 시스템은 도약 상태의 동작을 생성하기 위해 필요한 값들을 계산한다. 이 값들은 도약 상태가 얼마나 지속될 것인지와 도약 상태동안 발이 얼마나 움직여야 하는 가이다. 도약 상태는 오로지 중력에 의해서만 호퍼의 무게 중심이 가속되므로 도약 상태가 얼마나 지속될 것인지를 나타내는 도약 시간 T는 정확하게 계산할 수 있다. 이렇게 계산된 도약 시간 T에 따라

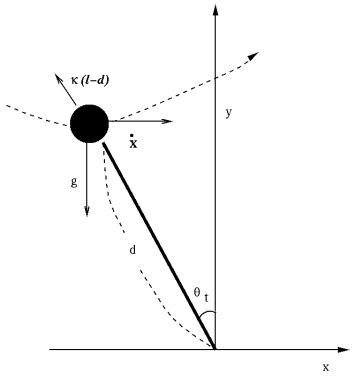

Figure 4: 착지 상태에 HIP의 위치에 영향을 미치는 요소들

호퍼가 도약 상태동안 어떻게 다리의 각도를 변경하여 착지할 것인지를 계산할 수가 있다. 다음절에서 이 T의 계산 방법과 그 계산을 통해 얻어진 T에 따라 어떻게 다리를 움직일 것인가를 살펴볼 것이다.

3.1.2 한 다리를 가진 호퍼의 도약 상태

도약 상태에서 HIP의 운동은 착지 상태의 마지막 순간에 가진 호퍼의 속도 $(\dot{x},\dot{y})$가 중력 가속도(g)에 의해서만 변화하는 운동이므로 도약 상태동안 HIP의 위치는 다음과 같이 변화한다.

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

이때 $\dot{y_0}$는 착지 상태의 마지막 y축 속도, 즉, 도약 상태의 초기 y축 속도이며, y_0는 이 때의 높이 y, 즉 도약 상태의 초기 높이를 의미한다.

도약의 지속시간(T)은 y가 다시 착지 상태로 바뀔 수 있는 높이가 될 때까지이다. 이 높이는 착지 상태의 초기 y값인 $l \cdot \cos\theta_0$과 동일하다. 이 값을 도약 상태의 최종 높이 y_f라고 할 때, $y_0 + \dot{y_0}t + g \cdot t^2/2 = y_f$를 만족하는 t가 도약 시간이 된다. 이 방정식은 두개의 해를 가지는데, 그 중 큰 값을 도약 시간으로 한다. 따라서, T는 다음과 같이 구할 수 있다.

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

이렇게 계산된 도약 시간 T 동안 호퍼의 HIP은 중력에 의한 운동을 하고 다리는 이 시간 T동안 도약 상태 초기의 각도, 즉 착지 상태의 마지막 다리 각도에서 착지 상태의 초기 각도로 변화하게 된다.

그림 6과 그림 7는 이러한 방법을 이용하여 하나의 다리를 가진 호퍼의 움직임을 구현한 것이다. 하나의 다리를 가진 호퍼 모델의 다리 길이는 가변적이지만, 인

간 주행을 위한 것이므로 그림 5와 같이 HIP과 $FOOT$중간에 관절 $JOINT$를 넣어 $|HIP-FOOT|$의 값은 하나의 다리를 가진 호퍼 모델과 동일하게 유지하면서 전체 다리의 길이, 즉, $|HIP-JOINT|+|JOINT-FOOT|$의 값은 l로 유지하였다. 하나의 다리를 가진 호퍼 모델에서 HIP와 $FOOT$ 사이의 거리가 d일 때 $HIP-JOINT$ 벡터와 $HIP-FOOT$ 벡터가 이루는 각을 θ_l이라 하며 이 값은 다음과 같이 간단히 구해져서 쉽게 $JOINT$의 위치를 결정할 수 있다.

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

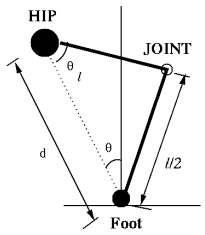

Figure 5: Joint를 이용해 표현한 하나의 다리를 가진 호퍼

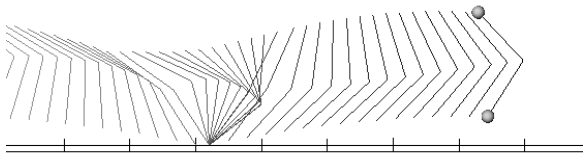

Figure 6: 하나의 다리를 가진 호퍼의 동작 1 ($h = 0.005sec,\ l = 2m,\ \dot{x} = 10m/sec,\ \kappa = 3000,\ \theta_0 = \pi/4,\ m = 50kg$)

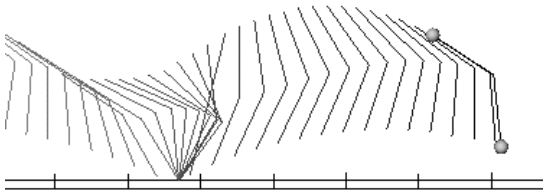

Figure 7: 하나의 다리를 가진 호퍼의 동작 2 ($h = 0.01sec,\ l = 2m,\ \dot{x} = 5m/sec,\ \kappa = 3000,\ \theta_0 = \pi/4,\ m = 50kg$)

하나의 다리를 가진 호퍼의 동작은 $l, \dot{x}, \kappa, \theta_0, m$에 의해 결정된다. 따라서 하나의 다리를 가진 호퍼의 주행 함수는 $R_{one_leg}(l, \dot{x}, \kappa, \theta_0, m)$으로 나타낼 수 있다.

3.2 두 다리 호퍼 (Two-Leg Hopper)

앞에서 설명한 한 다리 호퍼를 구현하는데 이용된 방법들은 두 다리를 가진 호퍼로 확장될 수 있다. 그림 8와 같이 두 다리를 가진 호퍼는 초기 상태를 결정하는 파라미터로 $\vec{L}$이 필요하다. 이 $\vec{L}$은 착지 상태에서 들려진 발과 HIP에 의해 결정된다.

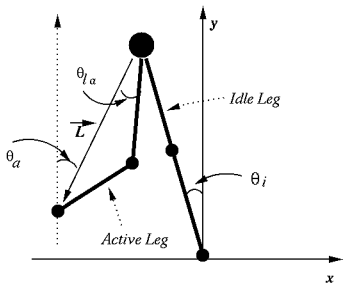

Figure 8: 두 다리를 가진 호퍼의 초기 상태

3.2.1 두 다리를 가진 호퍼의 착지 상태

두 다리를 가진 호퍼의 동작 역시 착지 상태와 도약 상태로 상태로 나뉜다. 착지 상태동안에는 지면과 닿은 다리가 하나의 다리를 가진 호퍼와 동일한 방법으로 움직인다. 이 때 HIP의 속도가 $(\dot{x}, \dot{y})$라고 하면 들려 있는 다리의 움직임은 들려 있는 발($FOOT_a$)의 위치 (f_{x_a}, f_{y_a})에 의해 결정되는데 이 값은 이 발의 속도 $(\dot{f}_{x_a}, \dot{f}_{y_a})$에 의해 결정된다. 움직이는 발의 속도는 다음과 같은 방법을 사용하여 구하였다.

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

이 때, S_x와 S_y는 각각 HIP의 x축, y축 속도를 스케일하는 변수이다. 이 값은 사용자가 직접 사용하기가 어려운 파라미터로 나중에 설명될 자동화 기법에서 자동으로 찾아지는 값이다. 이 S_x와 S_y는 발의 속도가 다음 도약 상태에서 일어나는 발의 동작과 부드럽게 연결되도록 정해져야 한다. 이 부드러운 정도를 연결도로 정의하고 가장 이상적인 연결이 이루어질 때 0의 값을 가지고 연결이 부드럽지 않을 수록 큰 값을 가지는 연결도 γ를 정의하여 이를 최소화하는 방법이 나중에 설명될 것이다.

3.2.2 두 다리를 가진 호퍼의 도약 상태

두 다리를 가진 호퍼의 도약 상태 역시 하나의 다리를 가진 호퍼 모델에서 구한 도약 시간 T동안 이루어진다. 이 시간동안 지면에 놓여 있던 다리($FOOT_i$)는 들려 있던 다리의 초기 상태와 같은 관절 각도로 변하고, 반대로 들려 있던 다리는 지면에 주행 초기에 지탱 다리와 y축이 이루었던 θ_0의 각도로 닿게 된다. 이를 위해서는 도약 초기에 각 다리 관절들의 각도와 도약 종료, 즉 착지시에 도달해야 하는 각도를 보간하여 중간 프레임의 각 관절 각도를 계산한다. 도약 시작 시간을 t_0라 하고 t_0 이후에

흐른 시간을 t라고 할 때, 각 관절은 다음과 같은 식으로 계산한다.

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

이 때, θ_{l_t}는 $(JOINT - HIP)$ 벡터와 $(FOOT - HIP)$ 벡터가 시간 t일 때에 이루는 각도이며 θ_{c_t}는 $(HII - FOOT)$이 y축과 이루는 각이다. θ_{l_0}과 θ_{c_0}는 각각 도약 초기의 θ_l과 θ_c 값을 나타내고 θ_{l_T}과 θ_{c_T}는 도약 종료, 즉 다시 지면에 발이 닿았을 때 이루어져야 할 각도이다. 이 값은 주행 초기 좌우 다리가 가졌던 값을 서로 바꾸어 가지게 된다. $B_l(t)$와 $B_c(t)$는 모두 $t = 0$일 때 0의 값을 가지고, $t = T$일 때 1의 값을 가지는 함수이다. $B_l(t)$는 단순히 $B_l(t) = t/T$로 θ_l의 값을 선형 보간하며 $B_c(t)$는 다음과 같은 2차 함수를 사용하였다.

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

이 때 α 값은 $B_c(t)$의 극대값이 $\alpha \cdot T$가 되도록 하여 도약 시간동안 다리의 움직임을 조절할 수 있는 파라미터이다. 이 α 값 역시 나중에 설명할 자동화 기법을 통해 생성되는 값이다.

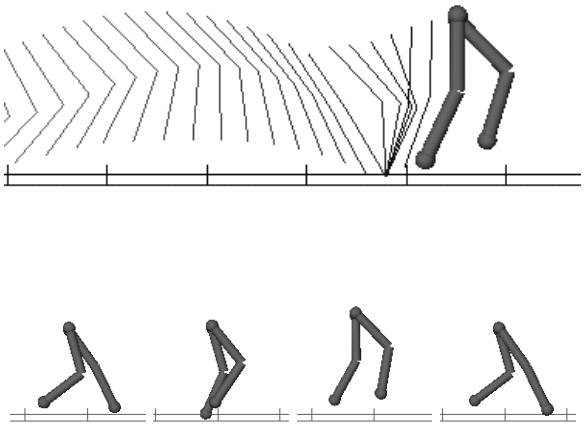

Figure 9: 두 다리를 가진 호퍼의 동작 $(h = 0.01sec,\ l = 1.5m,\ \dot{x} = 5m/sec,\ \kappa = 9000,\ \theta_0 = \pi/6,\ m = 50kg,\ \alpha = 0.9,\ S_x = 1.5,\ S_y = 1.0)$

두 다리를 가진 호퍼의 동작은 l, $\dot{x}$, κ, θ_0, m, α, S_x, S_y, $\vec{L}$에 의해 결정된다. 따라서 두 다리를 가진 호퍼의 주행 함수는 $R_{two_leg}(l, \dot{x}, \kappa, \theta_0, m, \alpha, S_x, S_y, \vec{L})$으로 나타낼 수 있다.

3.3 상체의 균형 잡기

인간 주행 동작을 사실적으로 표현하기 위해서는 몸이 쓰러지지 않도록 하는 균형잡기 동작을 표현하여야 한다. 이때 무게 중심의 위치는 균형상태를 판단하는데 유용하게 사용될 수 있는 것으로 무게 중심을 중력방향으로 평면에 프로젝션하였을 때에 그 위치가 지지하는 점이나 영역위에 놓인다면 균형이 잘 잡혀있는 상태이다. 따라서 균형의 정도는 그림 10과 같이 객체의 무게 중심을 중력방향으로 평면에 프로젝션한 위치와 지지하는 위치 사이의 거리로 판단할 수가 있다. 균형을 잡기 위해서는 이 무게 중심을 지지하는 발의 위치 위로 옮겨주기 위한 제어 방법이 제공되어야 한다.

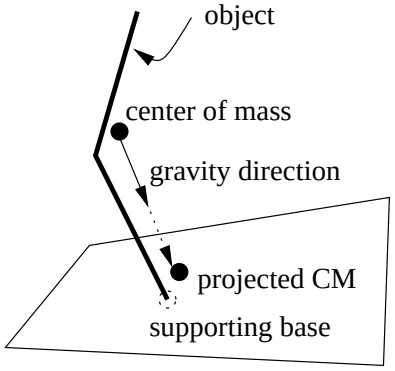

Figure 10: 균형잡기 동작의 기본 개념

로보틱스등의 분야에서 이 균형잡기 문제는 매우 중요하고도 어려운 문제이다. 이러한 분야에서는 객체의 무게 중심이 지지평면과 객체가 만나는 위치의 수직 상공에 0의 속도로 도달할 수 있도록 하는 많은 제어방법들이 개발되었다. 이러한 기법들은 물리적 법칙에 충실하여 매우 사실적인 시뮬레이션을 생성할 수가 있지만, 많은 계산량과 구현의 어려움이 따른다. 본 논문은 이러한 균형잡기를 쉽게 구현할 수 있는 효율적인 방법을 제시한다. 본 논문에서 다루는 균형잡기 동작은 주행중의 자세를 사실적으로 나타내기 위해 캐릭터의 무게중심을 지속적으로 지지 위치 상에 놓으려는 힘을 가하는 방법을 사용한다. 주행중의 상체 움직임은 물리적 관성에 의한 움직임과 무게 중심이 다리에 의해 지탱되도록 하는 자발적 운동으로 나뉘어 진다. 관성에 의한 힘은 이전까지의 동작을 통해 계산된 각 관절의 각속도와 각가속도를 이용하여 구할 수 있으며, 본 논문은 중심을 잡으려는 자발적 운동을 어떻게 표현할 것인지를 제시한다.

3.3.1 무게 중심의 위치

주행 도중 캐릭터의 무게 중심이 놓여야 하는 위치는 xz평면상의 위치만 고려하기로 한다. 착지 상태일 때 캐릭터가 무게 중심을 이동시키려고 하는 곳, 즉 목표 무게 중심(CM_{tar})은 지탱하고 있는 발의 위치$(FOOT_i)$로 가정한다. 도약 상태에서는 이전 착지 상태 때에 지탱하던 발$(FOOT_i)$과 공중에 떠서 움직이던 발$(FOOT_a)$의 위치를 도약 시간 T동안 선형 보간하여 얻는다. 즉, 캐릭터가 목표로 하는 무게 중심의 위치는 다음과 같이 얻을 수 있다.

$$\begin{aligned} CM_{tar} &= FOOT_i \\ &\quad , if\ mode\ is\ in\ STANDING \end{aligned} \quad (9)$$

$$CM_{tar} = (FOOT_i) + (FOOT_a - FOOT_i) \cdot \frac{t}{T}$$
$$, \; if \; mode \; is \; in \; JUMPING$$

이때, t는 도약이 시작된 이후 흐른 시간, T는 도약에 소모되는 총 시간이다. 캐릭터의 실제 무게 중심은 다음과 같이 구할 수 있다.

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

M_i는 질량이 부여된 캐릭터내의 각 질점의 질량을 나타내며, n은 이 질점들의 총 갯수이다. 그리고 L_i는 이 질점들의 위치를 나타낸다.

3.4 균형 잡기 동작의 생성

그림 11와 같이 식 (9)와 (10)을 통해 구한 목표 위치 CM_{tar}와 실제 무게 중심 CM_{cur}의 차를 오차, $\vec{err}$로 한다. 즉,

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

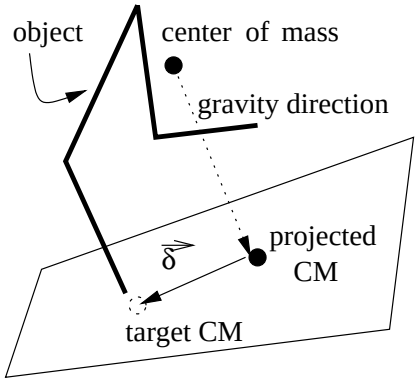

Figure 11: 목표 무게 중심과 실제 무게 중심의 차이 ($\vec{err}$)

이 오차의 x성분을 err_x, z성분을 err_z라 할 때 상체의 움직임은 이 오차의 x성분, z성분에 의해 결정된다. 몸통의 경우, err_x가 양수인 경우, 즉 목표 무게 중심이 실제 무게 중심보다 진행 방향으로 앞에 있는 경우는 몸을 앞으로 더 숙여야 한다. 즉 z축에 대해 양의 방향으로 회전이 더 일어나야 한다는 의미가 된다. 이 때, 몸통이 앞으로 숙이려고 하는 각도를 zd_b라고 하면 z축에 대한 몸통의 회전 Z_b는 다음과 같이 변한다.

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

이때, $\dot{Z}_b$와 $\ddot{Z}_b$는 각각 Z_b의 각속도와 각가속도이다. 이들은 관성에 의한 상체의 움직임을 나타내기 위한 것이다. 이 각속도, 각가속도와 함께 무게중심의 오차를 수정하기 위한 움직임 zd_b이 같이 영향을 미쳐 몸통의 z축에 대한 회전각이 변하게 된다. 이러한 균형잡기 모델의 동작은 그림 12과 같이 표현할 수 있다. 시스템은 계속해서 현재 동작과 목표 동작의 오차를 계산하여 동작 제어 모듈에 그 값을 주고 이를 받은 제어 모듈은 이 값을 줄이기 위한 동작 신호를 모델에게 보내 새로운 동작을 생성하는 일은 캐릭터가 움직이는 동안 계속해서 반복하는 것이다.

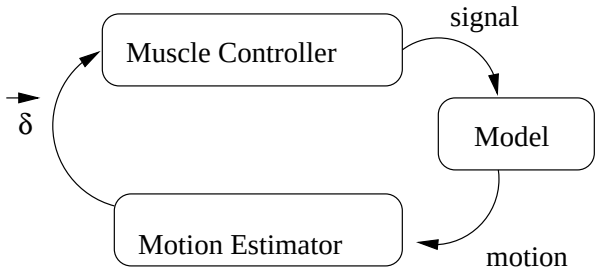

Figure 12: 균형 잡기의 흐름

본 논문의 실험을 위해서 오차를 수정하려는 zd_b를 다음과 같이 구하여 사용하였다.

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

여기서 h는 프레임간 간격이며, $\dot{x}$는 x축 진행 속도, 그리고 η는 오차에 대한 반응정도를 나타내는 파라미터이다. 이 η의 값을 크게 하면 무게 중심을 목표 위치로 옮기려는 동작이 커지고, 반대의 경우는 작아진다. 이 값이 커지면 몸이 휘청거리게 되고 작아지면 상체를 조금만 움직이며 주행을 하게 된다. 진행 속도 $\dot{x}$로 오차를 나눈 이유는 주행시 속도가 빠른 경우는 목표 무게 중심이 좌우의 발사이에서 빠르게 전환되기 때문에 무게중심을 많이 움직이지 않아도 되지만, 주행시 속도가 느릴 수록 목표 무게 중심의 위치가 천천히 변해 실제 무게중심을 목표 무게중심으로 빨리 옮겨주지 않으면 넘어진다는 점을 고려한 것이다. 이와 같은 방법으로 몸통의 z축 회전뿐 아니라 그림 13와 같이 다른 관절에도 목표 무게 중심과 실제 무게 중심의 오차를 줄이려는 관절 운동을 정의할 수 있다. 그림 13에서는 몸통과 어깨 관절의 예만을 보였다. 이때, err_z에 대한 몸통의 x축 회전의 반응을 나타내는 xd_b는 $xd_b = \eta \cdot (-err_z) \cdot h/\dot{x}$로 나타낼 수가 있으며, 다른 관절들도 비슷한 방법을 사용하여 정의할 수 있다.

그림 14과 15는 이러한 방법을 사용하여 주행중 몸의 균형을 잡는 동작을 보이고 있다. 그림 14는 η가 5, 진행 속도, $\dot{x}$가 1인 경우의 동작을 보이며, 그림 15는 $\eta = 5$, $\dot{x} = 5$인 경우를 보이고 있다. 그림에서 볼 수 있는 바와 같이 동일한 η값을 가지는 경우 속도가 느릴 수록 몸을 많이 움직여 균형을 잡는다.

이 균형잡기 동작을 테스트하기 위해 다리를 저는 동작을 실험하였다. 다리를 저는 동작을 표현하기 위해 저는 다리가 지탱하는 동안 캐릭터의 진행 속도를 증가시켜 저는 다리가 지탱하는 시간이 짧아지도록 하였다. 이

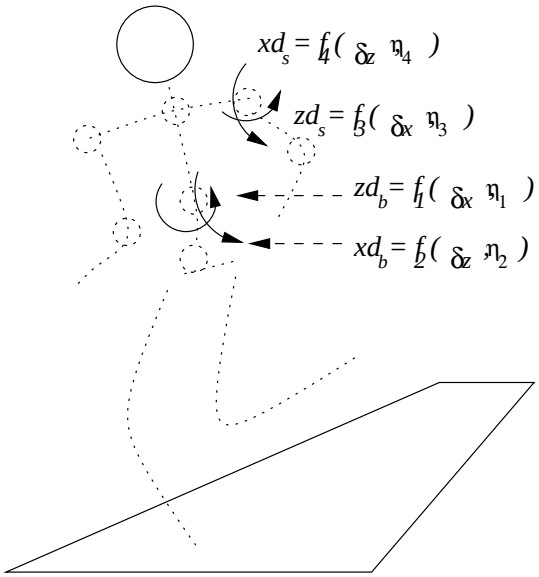

Figure 13: 각 관절의 균형 잡기 운동 (일부 관절의 예)

Figure 14: 균형잡기 동작 $\eta = 5$ $\dot{x} = 1$

는 실제 다리를 절 때 저는 다리에 무게 중심이 오래 머물지 못하도록 몸을 빨리 움직여 다른 다리가 지탱하도록 하는 동작을 표현한 것이다. 이 실험의 결과가 그림 16이다. 이 그림에서 볼 수 있듯이 한 쪽 다리로 지탱하는 시간이 짧을 경우 몸의 무게 중심이 다른 쪽, 즉 절지 않는 다리에 의해 지탱되도록 치우쳐짐을 볼 수가 있다. 그림 17는 목표 무게 중심과 실제 무게 중심의 이동 경로를 보여주고 있다. 점선으로 표시된 것이 목표 무게 중심의 이동 경로이고 실선이 실제 무게 중심이 이동한 경로이다.

3.5 균형잡기 파리미터의 생성

앞에서 설명한 균형잡기 모델은 모든 관절의 η의 값을 동일하게 설정하였다. 그러나 실제 애니메이션 생성에서는 각 관절에 서로 다른 η 값을 주어 무게 중심을 옮기는 동작을 이 η 값들의 벡터로 정의할 수 있게 한다. 즉,

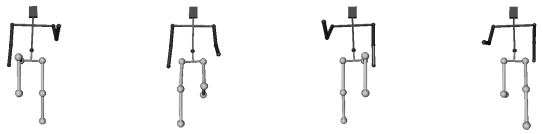

Figure 15: 균형잡기 동작 $\eta = 5$ $\dot{x} = 5$

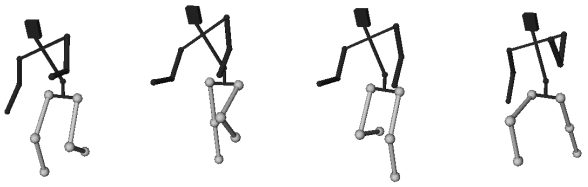

Figure 16: 다리를 저는 동작에서의 균형잡기 동작

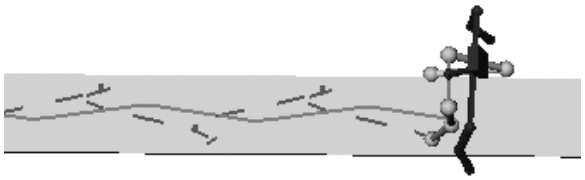

Figure 17: 무게 중심의 이동 경로 $\eta = 5$ $\dot{x} = 2$

$(\eta_0, \eta_1, \eta_2, \cdots, \eta_n)$과 같이 각 관절의 각 회전 방향에 대한 η 값들의 집합을 하나의 벡터로 표현하고 이를 유전자의 일부로 사용한다. 따라서 각 관절의 η 값들은 다음에 설명될 유전자 프로그래밍을 통한 제어 기법에 의해 자동적으로 생성된다. 이때 적당한 η 값들을 생성하는 방법은 몸 전체에 작용하는 토크(torque)를 최소화하는 η 벡터를 구하는 것이다. 이 방법은 뒤에서 설명할 에너지를 최적화를 이용한 애니메이션 제어 기법에서 중력에 의해 캐릭터에 작용하는 토크를 에너지의 일부로 계산하여 그 값을 최소화하는 것으로 구현된다.

4 제어

호퍼 모델과 앞에서 설명한 균형잡기 기법을 인간 주행 애니메이션에 적용하였다. 이제 이 기법들이 적용된 애니메이션을 제어하는 기술이 필요하다. 본 논문에서 사용하는 제어 기법은 사용자가 지정한 제한조건에 적합한 동작을 유전자 프로그래밍을 통해 찾는 것이다.

유전자 프로그래밍은 최적화 문제를 해결하는데 유용하며, 특히 컴퓨터 그래픽스 분야에서는 최적화된 동작을 생성하는데 많이 사용되었다. 유전자 프로그래밍은 최적화해야 하는 문제에 대한 정의와 평가 기준만 마련되면 쉽게 최적화 문제를 해결할 수 있으므로 애니메이션 동작 생성에 적용할 경우 동작 최적화에 필요한 복잡한 역학적 문제를 쉽게 해결할 수 있는 방법을 제공한다. 본 논문에서 제시하는 기법은 최적화의 대상으로 캐릭터가 사용하는 에너지와 두 개의 주행 상태(착지와 도약)의 연결 정도를 설정하였고, 이 두가지를 평가할 수 있는 측정 방법을 제공한다.

모든 캐릭터의 동작은 유전자 형태로 표현된다. 각 유전자는 캐릭터의 동작을 특정하는 파라미터들의 집합으로 l, $\dot{x}$, κ, θ_0, m, α, S_x, S_y, $\vec{L}$와 같이 두 다리를 가진 호

퍼 모델에 사용되는 파라미터와 η 벡터와 같이 균형잡기에 적용되는 파라미터가 있다. 사용자가 캐릭터의 속도를 지정하는 경우 속도를 표현하는 유전자는 사용자가 지정한 속도에 맞게 고정되고 다리의 길이나 질량같은 파라미터는 미리 지정된 값으로 설정된다. 사용자가 제어를 위해 속도등의 파라미터를 설정하면, 시스템이 유전자 프로그래밍을 위해 다수의 캐릭터 모델을 생성하고 이들에게 무작위의 유전자를 부여한다. 각 캐릭터는 자신의 유전자에 따라 주행 동작을 수행하고 시스템은 최적의 동작을 찾기 위해 각 캐릭터의 동작을 평가하여 우수한 유전자들을 골라내어 이들을 교배하거나 돌연변이를 일으켜 새로운 유전자를 생성하는 동작을 반복한다. 이러한 진화를 반복하여 생성된 마지막 유전자를 이용하여 애니메이터가 원하는 동작을 생성한다.

4.1 유전자의 평가

앞에서 말한 바와 같이 사용자가 속도를 지정하면 진화를 위해 경쟁할 각 캐릭터들의 유전자중 속도를 표현하는 부분은 지정한 속도로 고정된다. 나머지 유전자는 무작위로 생성되고 각 캐릭터는 자신의 유전자에 따라 움직이게 된다. 무작위로 생성된 캐릭터들의 상당수는 물리적으로 불가능한 동작을 보이게 된다. 이러한 캐릭터들은 경쟁에서 제외되고 다음 세대의 유전자를 생성하지 못하도록 제거된다.

주행 동작은 주기를 가지고 반복되는 것이므로 경쟁을 위한 동작은 초기 착지 상태에서 시작해 도약 상태가 끝나는 때까지를 수행한다. 이 동작이 끝나고 나면 시스템이 이들을 평가하는데 이때 동작이 얼마나 자연스러운가를 기준으로 평가를 할 수가 있다. 이때, 고려해야 할 점은 착지 상태에서 도약 상태로 전환될 때 발의 동작이 얼마나 자연스럽게 연결되는가와 전체 주행 동작에서 에너지를 얼마나 낭비없이 사용하는가이다. 이를 각각 연결도 γ, 에너지 소모율 E_r으로 나타내기로 한다. 이들은 다음과 같이 나타낼 수 있다.

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

이때, $\dot{\theta}_{f_i}$는 i번째 관절의 비행 초기 시간에서의 각속도, $\dot{\theta}_{s_i}$는 i번째 관절의 착지 상태의 마지막 시간에 가지는 각속도, 그리고 $\ddot{\theta}_{s_i}$는 그때의 각가속도를 나타낸다. 에너지 소모율 계산에서 사용된 t_s는 주행 시작 시간, t_f는 주행이 끝난 시간 그리고 $\ddot{\theta}_{i_t}$는 시간 t에서 i번째 관절의 각가속도를 의미한다. I_i는 i번째 관절의 회전 관성이다. 그리고 τ_t는 시간 t일 때에 중력에 의해 캐릭터에 작용하는 토크(torque)이다. 이 값은 무게중심의 위치와 목표 무게 중심의 위치를 계산하여 구할 수 있다.

연결도 γ는 비행 시작 시간의 각 관절의 실제 각속도 $\dot{\theta}_{f_i}$와 착지상태의 마지막 순간의 각속도와 각가속도에 의한 예상 각속도 $(\dot{\theta}_{s_i} + \ddot{\theta}_{s_i})$ 중 큰 값을 작은 값으로 나누어 이값이 1일 때, 즉 동일한 경우에 가장 작은 값을 갖도록 하였고, 에너지 소모율은 매 순간 각 관절의 에너지를 $|\ddot{\theta}_i|^2 \cdot I_i$라고 할 때 모든 관절의 에너지 총합을 시간에 대해 적분한 뒤 이를 이동 거리로 나누어 구하였다.

유전자의 평가를 위해 연결도와 에너지 소모율 값이 최소화될 때 자연스러운 동작이 나올 것이라고 가정하였다. 이 가정에 따라 연결도 γ와 에너지 소모율 E_r에 대해 다음과 같은 평가 함수 $eval(\gamma, E_r)$를 사용한다.

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

이 $eval(\gamma, E_r)$의 값은 γ와 E_r이 0일 때 최소값 0를 가지고 두 값이 무한대가 될 때 1이 되는 함수이다.

이렇게 유전자들을 평가하고 이 $eval(\gamma, E_r)$이 작은 값을 가지는 모델들을 선택하여 유전자를 서로 섞어 교배하는 유전자 프로그래밍 과정을 반복하게 된다. 이러한 유전자 프로그래밍이 끝나고 나면 최종적으로 가장 우수한 유전자를 선택하여 이를 애니메이션으로 생성한다.

4.2 에너지를 이용한 감정 상태의 제어

앞에서 설명한 기법은 단순히 에너지 소모율 E_r과 착지 상태와 도약 상태 사이의 각 관절의 속도 연결도 오차 γ를 최소화하기 때문에 자연스러운 동작을 생성할 수는 있지만 최적의 상태로 수렴되기 때문에 다양한 제어가 불가능하다. 실제 동작에서는 최적화된 동작을 취하는 것이 아니라 다양한 감정에 따라 서로 다른 동작을 나타낸다. 우리는 에너지 소모율을 조정함으로써 이러한 다양한 동작을 제어할 수 있도록 하였다. 즉, 주어진 제한 조건을 만족하면서 최소의 에너지 소모율로 동작하는 것이 아니라 에너지 소모율을 조정할 수 있게 하여 동작을 제어할 수 있게 하였다. 이것은 활기찬 동작은 최적의 동작보다 더 에너지를 소모할 것이라는 기본 아이디어에서 출발한다. 주행 중의 에너지 소모율 E_r을 통해 캐릭터를 제어하는 방법은 이 E_r을 최소화하지 않고 특정한 값에 수렴하도록 하는 것이다. 이를 위해 에너지 소모율의 최적화 정도를 결정하는 파라미터 β를 사용하였다. 이 값은 0보다 크거나 같고 1보다는 작은 값을 가진다. ($0 \le \beta < 1$) 앞에서 계산한 E_r 값을 최소화 하지 않고, 이 β를 사용하여 E_r이 수렴되는 위치를 β 값에 의해 제어할 수 있도록 하였다. 따라서 앞 절에서 설명한 평가 함수 $eval(\gamma, E_r)$은 다음과 같이 $eval(\gamma, E)$로 수정된다.

$$\begin{aligned} eval(\gamma, E) &= 1 - \frac{1}{2(1+\gamma)} - \frac{1}{2(1+E)} \\ & where\ E = \left(\frac{\beta}{1-\beta} - E_r \right)^2 \end{aligned} \quad (16)$$

여기서 E는 이전 $eval(M)$ 함수의 E_r을 대신하는 것으로 새로운 평가 함수는 γ와 이 E 값을 최소화하기 위

한 것이다. E는 β와 E_r의 함수로 이 값을 최소화하려면 β에 따라 E_r이 다르게 변해야 한다. β가 0이면 E_r값이 0이 될 때 E가 최소화되고, β가 1에 가까워질 수록 E를 최소화하는 E_r의 값이 증가하게 된다. 따라서, 이 평가 함수를 사용하면 E_r을 무조건 최소화하는 것이 아니라 β값을 통해 캐릭터의 에너지 소모율을 조정할 수가 있다. 캐릭터가 최적의 에너지 소모를 하며 움직이게 하려면 β를 0으로 설정하고 캐릭터가 활발하기 움직이게 하고 싶으면 이 β 값을 증가시켜 더 큰 에너지 소모율로 주행하게 할 수 있다.

5 실험 결과

Figure 18: 속도에 따른 동작 1 ($\dot{x} = 2m/sec$, $\eta = 5$, $\beta = 0$)

Figure 19: 속도에 따른 동작 2 ($\dot{x} = 5m/sec$, $\eta = 5$, $\beta = 0$)

Figure 20: 속도에 따른 동작 3 ($\dot{x} = 7m/sec$, $\eta = 5$, $\beta = 0$)

본 논문에서 제시한 기법을 구현하기 위해 SGI $Indigo^2$에서 $C++$와 Open Inventor 라이브러리를 사용하였다.

그림 18, 그림 19, 그림 20는 각각 속도를 $\dot{x} = 2ms^{-1}$, $\dot{x} = 5ms^{-1}$, $\dot{x} = 7ms^{-1}$로 지정하고 에너지 소모율 E_r과 연결도 오차 γ를 최소하여 얻은 최적 동작이다.

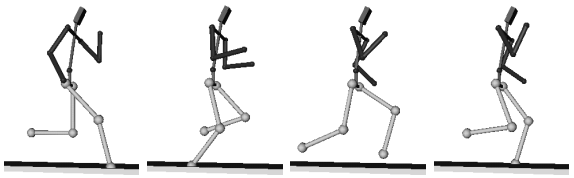

Figure 21: β를 이용한 동작 생성 (보통 주행) ($\beta = 0.2$ $\dot{x} = 10m/sec$)

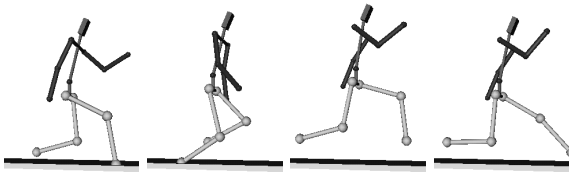

Figure 22: β를 이용한 동작 생성 (활기찬 주행) ($\beta = 0.7$ $\dot{x} = 10m/sec$)

그림 21과 그림 22는 속도는 동일하게 ($\dot{x} = 10ms^{-1}$) 지정하고 β 값이 서로 다르게 하여 동작을 생성한 예를 보이고 있다. 그림 21은 β 값을 0.2로 설정하였고, 그림 22는 β 값을 0.7로 설정한 것이다. 그림에서 볼 수 있듯이 β 값을 1에 가까이 할 수록 더욱 활발한 동작이 생성된다.

그림 23는 속도를 $5m/sec$로 지정하고 β를 0.5로 지정한 뒤 동작을 생성할 때, 각 단계에서 최적으로 평가된 유전자의 평가값들이 어떻게 변화하는지를 보이고 있다. 수평으로 그려진 축은 유전자 프로그래밍에서 진화가 진행되는 시간을 의미하는 세대를 나타낸다. 그리고 수직으로 그려진 축은 유전자 평가에 의해 계산된 평가값을 나타낸다. 그림에서 볼 수 있듯이 진화가 진행됨에 따라 평가값이 최소화한다. 그림 23에 나타난 평가값은 진화의 과정을 쉽게 알아보기 위해 유전자 프로그래밍을 수행하는 동안 가졌던 최대값을 max라 하고, 최소값을 min이라고 할 때 $(eval(\gamma, E_r) - min)/(max - min)$으로 스케일한 것이다.

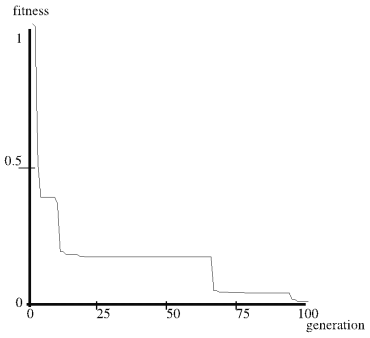

Figure 23: 유전자 평가값의 변화 (100 개체 100 세대)

6 결론 및 향후 연구과제

본 논문을 통해 우리는 캐릭터 모델이 소모하는 에너지를 제한하여 자동적으로 인간의 주행 동작 애니메이션을 생성하는 효율적인 기법을 제시하였다. 본 논문에서 우리는 인간 주행을 속도와 같은 개념적인 파라미터로 동작을 정의할 수 있는 주행 모델과 사실적인 동작을 위한 균형잡기 기법을 구현하였다. 이러한 기본 모델의 동작은 각 캐릭터마다 부여된 유전자에 의해 결정되는데 유전자 프로그래밍을 이용하여 에너지 소모율을 최적화하는 유전자를 생성함으로써 자연스러운 동작을 얻을 수가 있었다. 본 논문은 에너지 소모율을 최적화하여 동작을 자연스럽게 하는 기법을 더욱 확장하여 캐릭터의 에너지 소모율을 파라미터화하여 에너지 소모율을 제어할 수 있게 하였다. 이렇게 에너지 소모율을 최소화하는 대신 에너지의 소모를 조절할 수 있는 최적화 기법을 통해 "활기찬 주행"과 같은 감정적 상태를 제어할 수 있었다. 이러한 제어 기법의 기본적인 아이디어는 캐릭터 모델의 동작이 에너지 소모에 의해 지배된다는 것이다. 이 기법을 실제 애니메이션에 적용할 경우 애니메이터는 캐릭터의 각 관절의 움직임을 상세히 지정하지 않고도 자신이 지정한 속도로 주행하며 원하는 감정적 상태로 움직일 수 있도록 제어할 수 있다. 또한 우리는 본 논문에서 주행 동작에 사실성을 부여할 수 있는 균형잡기 모델을 제시하였다. 본 논문이 제시한 기법은 다음과 같은 장점을 가진다.

- 본 논문의 기법은 속도와 감정적 상태라는 개념적 제어 방법을 제공한다. 애니메이터는 특정한 주행 동작을 생성하기 위한 수학적 기술이나 지식 없이도 쉽게 동작을 생성할 수 있다.
- 감정이 포함된 동작을 생성하기 위해 이전에 제안된 많은 기법들이 필요로 했던 동작 데이터를 필요로 하지 않는다
- 원하는 동작을 생성하는 최적의 유전자가 생성되면, 이 유전자를 캐릭터 모델에 적용하여 실시간에 그 동작을 재생할 수 있다.
- 쉽게 구현가능한 균형잡기 모델을 제시하여 동작의 사실성을 높였다.

본 논문의 기법은 인간 모델의 주행에 관해서만 고려를 하였다. 앞으로의 연구 과제는 인간 모델뿐 아니라 동물이나 가상적인 생물에 대해 일반적으로 적용될 수 있는 주행 모델의 개발하는 것이다. 그리고 주행 동작 이외의 다양한 동작에 대해서 본 논문이 제안한 에너지 소모율 제한 기법을 적용하는 것도 앞으로의 연구 과제이다. 또한 본 논문에서 제시한 균형잡기 모델을 다른 여러 동작이나 캐릭터에 적용하고 개선하는 것도 앞으로의 연구 과제이다.

References

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

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

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

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

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

[8] J. K. Park, Y. M. Kang, S. S. Kim, and H. G. Cho. Expressive character animation with energy constraints. *Proceedings of COMPUGRAPHICS '97*, pages 260–268, Dec. 1997.

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

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

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