이학석사 학위논문

평면 뛰뛰기 모델과 에너지 제한 조건을 이용한 효율적인 인간 주행 동작 생성을 위한 제어 기법

1999년 2월

부산대학교 대학원

전 자 계 산 학 과

강영민

이학석사 학위논문

평면 뛰뛰기 모델과 에너지 제한 조건을 이용한 효율적인 인간 주행 동작 생성을 위한 제어 기법

지도교수 조 환 규

1999년 2월

부산대학교 대학원

전 자 계 산 학 과

강영민

강영민의 이학석사 학위논문을 인준함

1998년 12월

주 심 권 혁 철 (인)

위 원 차 의 영 (인)

위 원 조 환 규 (인)

An Efficient Control over Running Animation with One-leg Hopper Model and Energy Constraints

Young-Min Kang

Department of Computer Science, Pusan National University

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 through tedious and time-consuming work. 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."

감사의 글

두 해가 지나 다시 삶의 한 마디를 마감하는 때가 되었습니다. 어떤 사람이 성장하고 발전한다는 것이 스스로의 힘에만 달린 것이 아니라 누구를 만나 어떤 영향을 받느냐도 무시할 수 없이 중요하다는 점에서 지나간 석사과정은 제게 행운의 시간이었고, 또한 감히 성장과 발전이 시기였다고 할 수 있습니다. 2년간 한 걸음 한 걸음 앞으로 나아갈 수 있도록 학문적 지식과 공부하는 자세를 가르쳐 주셨을 뿐만 아니라 살아가는 방법까지도 지도해 주신 스승 조환규 교수님께 감사를 드리며, 심사를 맡아 주셨던 권혁철 교수님과 차의영 교수님, 그리고 수업 안팎을 통해 많은 것을 가르쳐 주셨던 다른 모든 교수님들께도 감사를 드립니다.

신입생으로 대학원 생활을 시작했을 때, 선배로서 여러 도움과 생활의 즐거움까지 주었던 친구 상현, 재권에게 고마움을 전하며, 부족한 것을 고칠 수 있도록 해 준, 대학원 선배들, 동규형, 영중형, 용빈, 민아에게도 감사를 드립니다. 또한, 연구실의 대선배이자 어른으로서 모든 면에서 큰 도움이 되어 주셨던 이도훈 박사님과 조미경 박사님께도 감사를 드리며, 2년간의 대학원 생활을 통해 누구보다도 더 많은 시간을 같이 보내면서 같이 힘들어 하고, 같이 즐거워 했던 동기들, 훈희, 세영, 호열, 성수에게도 감사와 애정을 드립니다. 그리고, 때론 선배들의 장난 상대로, 때론 유능한 보조로, 때론 연구실의 일꾼으로서 힘이 되었던 후배들, 양수, 환철, 선진, 정민에게도 고마운 마음을 드립니다. 또한 남들보다 조금 늦은 공부로 나이 많은 막내가 되어버린 정현형. 그 부지런함과 노력으로 여러 일을 도와주신 것에 감사 드립니다.

마지막으로, 2년동안 언제나 힘과 용기을 주셨던 부모님들께 말로 다할 수 없는 사랑과 감사를 드리고, 자주 집을 찾지도 않는 장남을 대신하여 장남의 역할을 했던 동생 성민, 바른 모습으로 형의 생활을 다잡아 주었던 막내 세민에게 사랑하는 마음을 표현하며 두 해동안의 결실인 이 논문을 가족 모두에게 바칩니다.

차 례

그림 차례

I 서론

컴퓨터를 이용한 캐릭터 애니메이션은 캐릭터들을 생성하여 특정한 행동을 하게 하는 것이다. 이러한 캐릭터 애니메이션에서 사용되는 동작 생성 기법의 중요한 목표는 캐릭터를 애니메이터의 의도에 따라 정확히 제어하는 것과 최소의 작업을 통해 효율적으로 캐릭터의 동작을 생성할 수 있는 자동화된 방법을 제공하는 것이다. 지금까지 소개되어 사용되고 있는 동작 생성 기법들은 제어와 자동화라는 두 가지 목표에서 하나의 목표에만 치중하여 다른 하나의 목표는 희생할 수밖에 없었다. 이것은 제어와 자동화가 서로 상충하는 목표이기 때문이다. 컴퓨터 애니메이션 연구의 주요한 목적 가운데 하나는 바로 이 제어와 자동화라는 상충하는 목표를 동시에 이룰 수 있는 효율적인 동작 생성 기법을 찾는 것이다. 본 논문은 이러한 목표를 이루기 위한 연구의 결과로, 물리적 모델에 기반한 주행 동작을 복잡한 파라미터가 아니라, 속도나 주행의 활기참과 같은 개념적인 파라미터를 통해 효율적으로 제어하는 방법을 제시한다.

현재 캐릭터 애니메이션에서 가장 널리 이용되고 있는 방법은 키프레임 기법으로, 사용자가 동작의 중요한 순간들을 일일이 지정하고, 그 사이의 동작을 여러 가지 보간(interpolation) 방법을 이용하여 생성하는 것이다. 이 키프레임 기법은 사용자의 의도에 따라 캐릭터를 제어하는 데에는 큰 장점을 가지지만, 동작을 생성하기 위해서는 오랜 시간이 필요하다는 점과, 동작의 사실성이 오로지 사용자의 능력에 달려 있다는 단점을 가지고 있다. 이러한 단점을 해소하기 위해 제안된 자동화 기법들 역시 한계를 가지고 있다. 효율적인 동작 생성을 보장하기 위한 자동화 기법은 사용자의 개입을 최소화하여 효율적으로 동작을 생성하기는 하지만, 생성되는 동작을 사용자 자신의 의도에 따라 정확하게 제어하기가 어렵다는 한계를 가진다.

본 논문이 제안하는 기법은 캐릭터의 주행동작을 자동화된 방법으로 생성하면서 개념적인 제어 파라미터를 통해 쉽게 동작을 제어할 수 있도록 하는 것으로, 자동화와 제어라는 상충하는 목표를 함께 이룰 수 있는 방법을 찾기 위한 연구의 결과이다.

기본적인 주행 동작은 물리적 시뮬레이션에 기반을 둔 모델로, 이 주행 모델에 적용되는 다양한 파라미터에 따라 사실적인 동작을 자동적으로 생성한다. 그러나, 주행 모델에 적용되는 파라미터들은 사용자가 직접 조작하기가 어려운 물리적 파라미터들이기 때문에, 사용자가 이를 적절히 설정하여 원하는 동작을 생성한다는 것은 거의 불가능한 일이다. 본 논문은 이런 제어 문제를 해결하고 사용자에게 쉬운 제어 방법을 제공하기 위해 유전자 프로그래밍을 이용한다. 따라서, 사용자는 캐릭터의 주행 모델을 제어하는 물리적 파라미터가 아니라 동작의 활기참이나 속도와 같은 개념적인 파라미터만을 이용하여 동작을 표현하고, 애니메이션 시스템은 이러한 개념적 파라미터를 만족하는 실제 파라미터들, 즉, 주행 모델에 적용되는 파라미터들을 유전자 프로그래밍을 이용하여 생성하는 것이다. 이러한 접근 방법은 자동화된 동작 생성 방법을 제공하는 동시에, 사용자가 개념적으로 동작을 제어할 수 있도록 한다.

캐릭터 애니메이션에서는 고려해야 하는 또 다른 문제는 사람뿐 아니라 다양한 동물이나 가상의 생물, 심지어 무생물까지 행동의 주체가 될 수 있다는 것이다. 현재 각광 받고 있는 모션 캡쳐 기법은 기계적 장비를 이용하여 실제 동작 데이터를 얻어 이를 골격 모델에 매핑하는 것으로 사실적인 동작을 생성하는데 매우 효율적인 방법이다. 하지만, 이 방법은 고가의 하드웨어 장비를 필요로 하며, 가상 생물의 동작이나 만화적으로 과장된 동작은 생성할 수 없다. 또한 생성된 동작 데이터는 하나의 동작만을 표현하기 때문에 서로 다른 동작을 생성하거나, 다른 모델에 적용할 때에는 동작을 다시 캡쳐하거나, 동작을 변형하는 기술을 사용하여야 한다는 제약을 가지고 있다. 그러나, 본 논문이 제안하는 자동화 기법과 개념적 제어라는 접근법은 이러한 모션 캡쳐 기법의 한계를 극복하여 다양한 모델의 다양한 동작을 자동화된 작업을 통해 생성하고 제어할 수 있는 방법을 제공한다.

본 논문은 이러한 접근 방법을 통해 인간의 주행 동작을 생성하는 애니메이션 기법을 제공한다. 현재까지 많은 애니메이션 기법들이 제안되어 인간 골격 모델의 애니메이션을 생성하는데 이용되었으며[4, 6, 8, 10], 자동화된 주행 동작 생성으로는 로보

틱스 기술에 기반한 Raibert와 Hodgins의 연구가 대표적이다. Raibert와 Hodgins는 뛰뛰기(hopper) 모델을 컴퓨터 애니메이션에 적용하여 두 다리, 네 다리 및 평면 캥거루 등의 주행 애니메이션을 구현하였다[10]. 이 방법은 물리적 법칙에 기반한 로보틱스 기법을 사용함으로써 사실적인 주행 동작을 생성할 수 있지만, 개념적 제어 방법을 제공하지 않으므로 동작을 제어하기가 어렵다. 주행 동작을 자동화된 기법을 통해 생성하는 다른 접근 방법으로는 신경 회로망에 기반한 "Sensor Actuator Network"을 사용하는 van de Panne과 Fiume의 방법이 있다[8]. 이 방법은 캐릭터의 동작을 표현하기 위한 수학적 도구 없이도 쉽게 동작을 생성할 수 있다는 장점이 있지만, 다른 자동화 기법들과 마찬가지로 자동화에만 치중하여 동작의 제어 능력이 떨어진다. 최근에는 Grzeszczuk와 Terzopoulos, 그리고 Hinton이 빠른 학습 속도를 가진 신경회로망을 이용하여 물리적 시뮬레이션의 계산 부담을 줄이고 목표를 달성할 수 있는 NeuroAnimator라는 기법을 제안하였다[5]. 이 기법은 자동화 기법이면서도 물리적 계산 부담을 줄였다는 장점이 있지만, 역시 목표를 달성하는 과정의 동작을 사용자가 의도대로 제어하는 능력이 떨어진다.

본 논문의 기법이 사용하는 개념적 파라미터는 "활기참"이라는 감정적 상태를 포함한다. 캐릭터의 움직임을 감정에 기반하여 제어하는 것은 최근에 활발히 연구가 진행되고 있는 분야로 동작 특성 파라미터를 보간(interpolation)하거나 외삽(extrapolation)하는 기법들이 사용된다[12]. 이러한 방법들은 동작 생성이 간단하다는 장점이 있지만, 보간이나 외삽에 필요한 동작 데이터가 없는 상태에서는 애니메이션을 생성할 수 없다는 한계를 가진다. 본 논문은 모션 캡쳐 등에 의해 생성된 어떤 데이터도 없이 동작의 감정적 상태를 제어할 수 있는 방법을 제공한다.

II 애니메이션 시스템 개요

2.1 캐릭터 모델의 구조

본 논문의 실험을 위해 그림 2.1과 같은 캐릭터 모델을 사용하였다. 이 모델은 다양한 자유도를 가지는 11개의 관절로 이루어져 있다. 인체 모델을 정확히 표현하기 위해서는 더욱 많은 관절이 필요하지만, 본 논문에서는 인간의 주행만을 다루고 있으므로 이를 위해 꼭 필요한 관절들로 인간 모델을 단순화하였다.

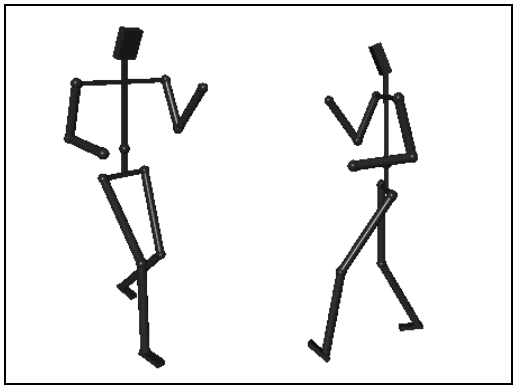

그림 2.1: 11개의 관절을 가진 캐릭터 모델

2.2 애니메이션 기법 개요

본 논문에서 제시하는 기법을 구현한 시스템의 사용자 인터페이스는 그림 2.2과 같다. 이 시스템은 본 논문에서 다루는 애니메이션 기법을 이용하여 인간형 캐릭터 모델의 주행 동작 데이터를 생성하는 시스템으로, 본 논문에서 설명되는 동작 생성 기법에 필요한 기존적인 입력뿐 아니라, 모델을 변경하고 주행 경로를 설정하는 것들과 같이 주행 동작을 더욱 자세히 표현하기 위한 다양한 입력을 제공한다.

본 논문의 기법은 사실적인 인간의 주행 동작을 생성하기 위해 몇 개의 파라미터로 인간의 주행을 표현할 수 있는 모델을 사용하였다. 이 모델을 구현하기 위해 우

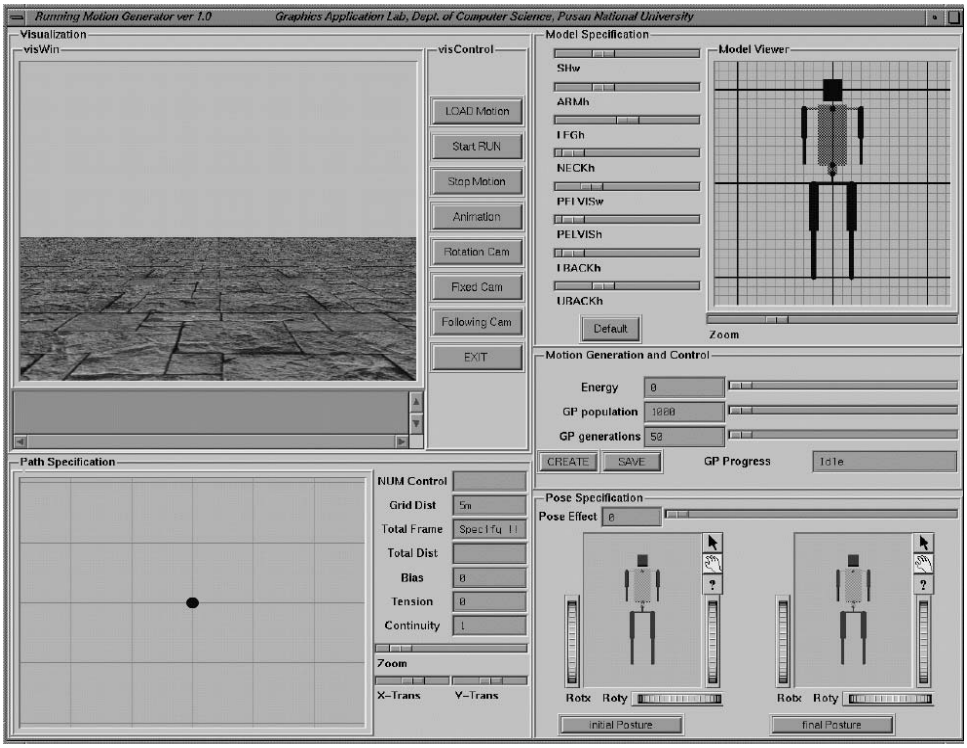

그림 2.2: 시스템 인터페이스

선 하나의 다리를 가진 평면 뛰뛰기(planar hopper) 모델을 구현하였고 이러한 동작 모델을 인간의 다리에 적용할 수 있도록 두 개의 다리를 고려하여 몇 가지 파라미터를 추가함으로써 두 다리를 가진 평면 뛰뛰기 모델로 확장하였다. 이러한 뛰뛰기 모델은 로보틱스 분야에서 활발히 연구된 것이다[3]. 본 논문은 이 뛰뛰기 모델을 기반으로 인간 하체의 주행 동작을 생성하였고, 상체의 자연스러운 움직임을 위하여 무게 중심이 다리에 의해 지탱될 수 있도록 하는 균형 모델을 구현하였다. 상체와 하체의 움직임을 표현하는 이러한 모델들이 생성하는 동작은 모델에 적용되는 다양한 파라미터에 의해 결정된다. 그런데, 앞서 말한 것처럼, 사용자가 주행 동작을 제어하기 위해 모델에 적용되는 파라미터들을 직접 다루는 방식은 불편할 뿐 아니라, 사용자 입력에 따라 어떤 동작이 생성될지를 예측하는 것도 쉽지가 않다. 이를 해결하기

위해 모델에 적용되는 파라미터들을 유전자 형태로 표현하였으며 주어진 조건에 대해 최적의 동작을 생성하기 위해 유전자 프로그래밍을 이용하였다.

본 논문을 구현한 애니메이션 시스템이 동작을 생성해 내는 과정의 개요는 다음과 같다. 동작을 생성하기 위해 시스템은 다수의 캐릭터 모델을 생성한다. 각 캐릭터 모델은 자신만의 파라미터를 가지고 있고, 이 파라미터를 기본 주행 모델에 적용하여 다양한 동작을 취할 수 있다. 이러한 주행 파라미터들은 유전자의 형태로 저장되어 있다. 시스템은 각 캐릭터 객체들이 가진 다양한 파라미터에 의해 생성되는 여러 동작들을 평가하여 순위를 매긴다. 이때, 평가의 기준은 사용자가 입력한 동작의 "활기참"과 같은 개념적 파라미터에 의해 결정된다. 이렇게 평가된 캐릭터의 유전자는 평가 결과에 따라 순위가 높은 유전자가 다음 세대 유전자를 생성하는데 많은 기여를 하도록 설정된 조건 아래에서 교배나 돌연변이 과정을 거치게 된다. 이 과정이 지나면 다음 세대의 새로운 유전자들이 생성되고, 이 유전자들을 다시 캐릭터 모델에 적용하여 평가를 반복하게 된다. 이러한 모의 진화는 사용자의 요구에 더욱 적합한 동작이 생성되도록 유전자들을 변화시킨다. 이런 방법을 통해 최적의 유전자가 얻어지면 애니메이션 시스템이 최적의 유전자를 사용하여 캐릭터의 동작을 생성하여 사용자에게 제공한다. 이러한 유전자 프로그래밍은 컴퓨터 애니메이션 분야에서 동작을 생성하기 위한 최적화의 도구로 이미 많이 사용된 것이다 [4, 9, 11]. 그림 2.3는 전체 시스템 중에서 속도와 에너지 입력을 받아 동작을 생성하는 부분의 흐름의 보이고 있다.

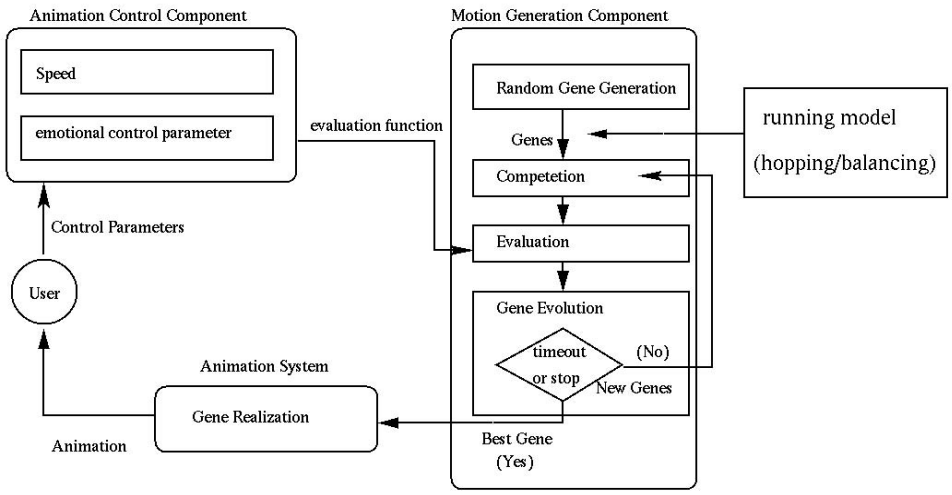
Animation Control Component
Speed
emotional control parameter
evaluation function
Motion Generation Component
Random Gene Generation
Genes
running model
(hopping/balancing)
Competetion
Evaluation
Gene Evolution
timeout
or stop
(No)
New Genes
Control Parameters
User
Animation System
Gene Realization
Animation
Best Gene
(Yes)

그림 2.3: 시스템 개요

III 동작 모델

이 장에서는 자동화된 동작 생성을 위해 구현한 동작 모델을 설명한다. 동작 모델은 뛰뛰기 모델과 균형 잡기 모델로 이루어진다. 뛰뛰기 모델은, 하나의 다리를 가진 평면 뛰뛰기 모델을 구현한 뒤, 이를 두 개의 다리로 확장하여 인간형 모델에 적용하였고, 여기에 균형 잡기 모델을 결합하여 인간 주행 동작을 모델링하였다.

3.1 한 다리 뛰뛰기 모델(One-Leg Hopper)

인간 주행을 모델링하기 위해 그림 3.1와 같이 하나의 다리로 구성된 뛰뛰기 모델을 구현하였다. 그림 3.1에 나타난 x축이 다리가 놓여지는 지면의 역할을 하며 뛰뛰기 모델은 이 x축 양의 방향으로 이동한다. 뛰뛰기 모델의 다리 길이는 변할 수 있으며, 다리의 길이가 원래의 값보다 줄어들면 다리는 스프링의 역할을 하게 된다. 이 뛰뛰기 모델의 상단 부분의 위치를 HIP이라 하고 (x, y)의 2차원 벡터형태로 표현한다. 발 부분의 위치는 $FOOT$으로 나타내고 이 값도 역시 (f_x, f_y) 형태의 2차원 벡터이다. 뛰뛰기 모델의 주행을 결정하는 파라미터는 다리의 스프링 상수(κ), 질량(m), 다리의 길이(l), 착지 각도(θ_0), 수평 이동 속도($\dot{x}$)가 사용된다.

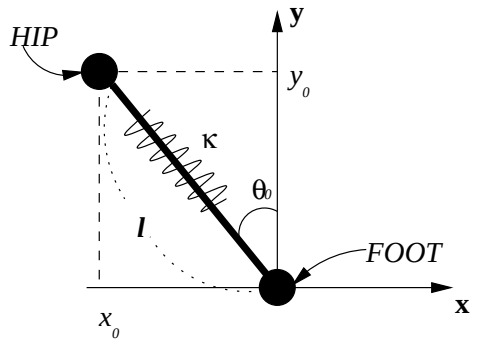

그림 3.1: 하나의 다리를 가진 뛰뛰기 모델의 초기 상태

주행 시작 시간에 뛈뛰기 모델의 위치는 다음과 같다.

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

하나의 다리로 움직이는 뛈뛰기 모델의 주행을 모델링하기 위해 주행 상태를 두 가지로 나눈다. 첫 번째 상태는 발이 땅에 닿아 있는 착지 상태($FOOT_y = 0$)이며, 두 번째 상태는 공중에 떠 있는 도약 상태이다. 이 두 가지 상태를 번갈아 반복하면 뛈뛰기 모델의 주행 동작을 생성할 수 있다.

3.1.1 한 다리를 가진 뛈뛰기 모델의 착지 상태

그림 3.2는 착지 상태동안 HIP의 위치에 영향을 주는 요소들을 나타낸다.

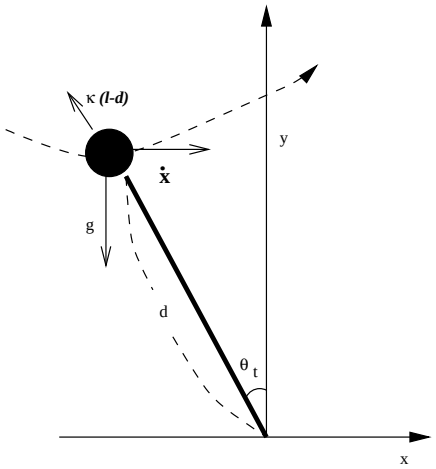

그림 3.2: 착지 상태에 HIP의 위치에 영향을 미치는 요소들

평면 뛈뛰기 모델의 HIP은 고정된 수평 속도 $\dot{x}$로 진행한다. y축 속도는 중력 가속도에 의한 낙하 운동과 길이 d로 압축된 다리의 탄성에 의한 운동의 합으로 나타난다. 따라서, 프레임간의 시간 간격을 h, 중력 가속도를 g, 시간 t에 뛈뛰기 모델과

y 축이 이루는 각도를 θ_t, HIP의 질량을 m이라고 할 때, 시간 t에서의 HIP의 위치는 다음과 같다.

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

이것은 x축 양의 방향으로는 $\dot{x}$의 속도로 전진하면서, 높이는 중력 가속도 g와 다리의 스프링에 영향을 받는 것을 표현한 것이다. 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 어떻게 움직여야 하는 가이다. 도약 상태에서는 오로지 중력에 의해서만 뛰뛰기 모델의 무게 중심이 가속되므로 도약 상태가 얼마나 지속될 것인지를 나타내는 도약 시간 T는 정확하게 예상하여 계산할 수 있다. 이렇게 계산된 도약 시간 T에 따라 뛰뛰기 모델이 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 뛰뛰기 모델의 도약 상태

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

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

이때 t는 도약이 시작한 이후에 흐른 시간이며, $\dot{y_0}$는 착지 상태의 마지막 y축 속도, 즉, 도약 상태의 초기 y축 속도이다. 그리고, y_0는 이 때의 높이인 도약 상태의 초기 높이를 의미한다.

도약의 지속시간(T)은 y가 다시 착지 상태로 바뀔 수 있는 높이가 될 때까지 걸리는 시간이다. 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 되므로 T는 다음과 같이 구할 수 있다.

$$T = \frac{-\dot{y_0} - \sqrt{\dot{y_0}^2 + 2 \cdot g \cdot c}}{g} \quad , where \quad c = y_f - y_0 \tag{4}$$

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

하나의 다리를 가진 뛈뛰기 모델의 다리 길이는 가변적이지만, 인간 주행을 위한 것이므로 그림 3.3과 같이 HIP과 $FOOT$중간에 관절 $JOINT$를 넣어 $|HIP - FOOT|$의 값은 하나의 다리를 가진 뛈뛰기 모델과 동일하게 유지하면서 전체 다리의 길이, 즉, $|HIP - JOINT| + |JOINT - FOOT|$의 값은 l로 유지하였다.

그림 3.4과 그림 3.5는 이러한 방법을 이용하여 하나의 다리를 가진 뛈뛰기 모델의 움직임을 구현한 것이다. 하나의 다리를 가진 뛈뛰기 모델의 동작은 l, $\dot{x}$, κ, θ_0, m에 의해 결정된다. 따라서 하나의 다리를 가진 뛈뛰기 모델의 주행 함수는 $R_{one_leg}(l, \dot{x}, \kappa, \theta_0, m)$으로 나타낼 수 있으며, 그림 3.4과 그림 3.5는 서로 다른 파라미터를 적용한 결과이다.

3.2 두 다리 뛈뛰기 모델(Two-Leg Hopper)

앞에서 설명한 한 다리 뛈뛰기 모델을 구현하는데 이용된 방법들은 그림 3.6과 같은

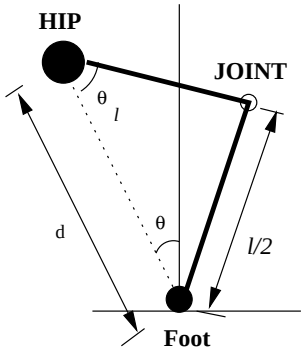

그림 3.3: Joint를 이용해 표현한 하나의 다리를 가진 뛈뛰기 모델

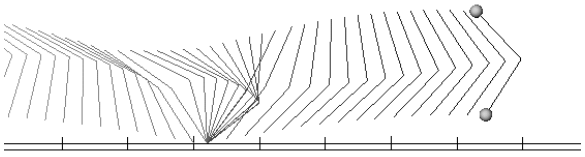

그림 3.4: 하나의 다리를 가진 뛈뛰기 모델의 동작 1 ($h = 0.005sec,\ l = 2m,\ \dot{x} = 10m/sec,\ \kappa = 3000,\ \theta_0 = \pi/4,\ m = 50kg$)

두 다리를 가진 뛈뛰기 모델로 확장될 수 있다. 두 개의 다리를 가진 뛈뛰기 모델은 두 개의 다리가 번갈아 가면서 한 다리 뛈뛰기 모델의 역할을 수행하는데, 이렇게 한 다리 뛈뛰기 모델의 역할을 수행하는 다리를 유휴(idle) 다리라 하고, 한 다리 뛈뛰기 모델의 역할을 수행하지 않고 공중에서 움직이는 다리를 활성(active) 다리라고 정의한다. 본 기법의 동작 생성에서는 유휴 다리가 활성 다리보다 더 중요한 역할을 하지만, 보행 동작 생성 등에 관한 기존의 많은 연구에서 지면과 닿아 있는 다리를 유휴 다리라 하고, 공중에서 움직이는 다리를 활성 다리라고 정의하기 때문에 이를 따른다.

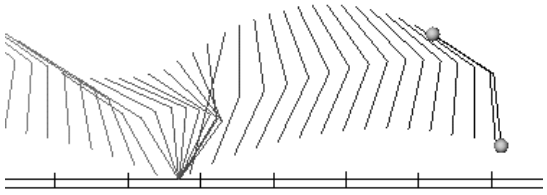

그림 3.5: 하나의 다리를 가진 뛰뛰기 모델의 동작 2 ($h = 0.01sec,\ l = 2m,\ \dot{x} = 5m/sec,\ \kappa = 3000,\ \theta_0 = \pi/4,\ m = 50kg$)

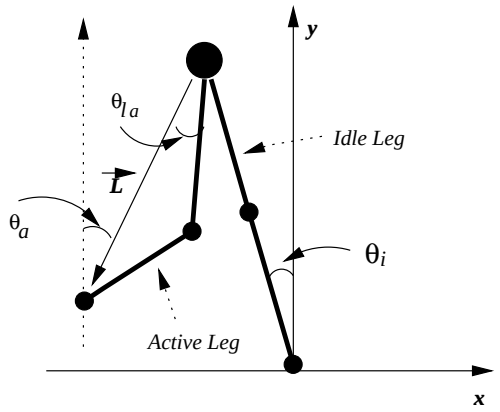

그림 3.6: 두 다리를 가진 뛰뛰기 모델의 초기 상태

3.2.1 두 다리를 가진 뛰뛰기 모델의 착지 상태

두 다리를 가진 뛰뛰기 모델의 동작 역시 착지 상태와 도약 상태로 나뉜다. 착지 상태동안에는 지면과 닿은 유휴 다리가 하나의 다리를 가진 뛰뛰기 모델과 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 발의 움직임을 생성하는 것은 HIP의 움직임을 적절히 스케일하는 단순한 방법을 사용하거나 발의 움직임을 표현하는 함수 등을 사용하여 구현할 수 있다. 이 활성 다리의 움직임은 파라미터를

통해 조정할 수 있게 하여 이 파라미터들을 나중에 설명할 유전자 프로그래밍을 통해 자동적으로 결정한다. 이 파라미터들을 결정할 때에는 착지 상태에서 이루어지는 활성 다리의 움직임과 도약 상태에서 보간을 통해 생성되는 다리의 동작이 부드럽게 연결되도록 파라미터를 결정해야 하는데, 이 부드러운 정도를 연결도로 정의하고 가장 이상적인 연결이 이루어질 때 0의 값을 가지고 연결이 부드럽지 않을 수록 큰 값을 가지도록 하여 이를 최소화하는 방법이 나중에 설명될 것이다.

3.2.2 두 다리를 가진 뛰뛰기 모델의 도약 상태

두 다리를 가진 뛰뛰기 모델의 도약 상태 역시 하나의 다리를 가진 뛰뛰기 모델에서 구한 도약 시간 T동안 이루어진다. 이 시간동안 지면에 놓여 있던 유휴 다리의 발($FOOT_i$)은 들려 있던 활성 다리의 착지 상태 초기 위치로 이동하고, 반대로 들려 있던 활성 다리는 착지 상태 초기에 모델을 지탱하던 유휴 다리와 y축이 이루었던 θ_0의 각도로 지면에 닿게 된다. 이를 위해서는 도약 초기에 각 다리 관절들의 각도와 도약 종료, 즉 착지 때에 도달해야 하는 각도를 보간하여 중간 프레임의 각 관절 각도를 계산한다. 도약 시작 시간을 t_0라 하고, t_0 이후에 흐른 시간을 t, 도약 지속 시간을 T라고 할 때, 각 관절은 다음과 같은 식으로 계산한다.

$$\begin{aligned} \theta_{l_t} &= \theta_{l_0} + (\theta_{l_T} - \theta_{l_0}) \cdot B_l(t) \\ \theta_{c_t} &= \theta_{c_0} + (\theta_{c_T} - \theta_{c_0}) \cdot B_c(t) \end{aligned} \tag{5}$$

이 때, θ_{l_t}는 $(JOINT - HIP)$ 벡터와 $(FOOT - HIP)$ 벡터가 시간 t일 때에 이루는 각도이며 θ_{c_t}는 $(HIP - FOOT)$이 y축과 이루는 각이다. 이 식은 유휴 다리와 활성 다리의 구분 없이, 도약 상태 동안 두 다리 모두에 적용되는 보간식이다. 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 보간하

였고, $B_c(t)$는 다음과 같은 2차 함수를 사용하였다.

$$B_c(t) = \left(\frac{1}{T^2 \cdot (1-2\alpha)}\right) t^2 + \left(\frac{-2\alpha}{T + (1-2\alpha)}\right) t \tag{6}$$

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 유전자 프로그래밍 기법을 통해 사용자의 개념적 제어 파라미터에 적합한 값으로 생성된다.

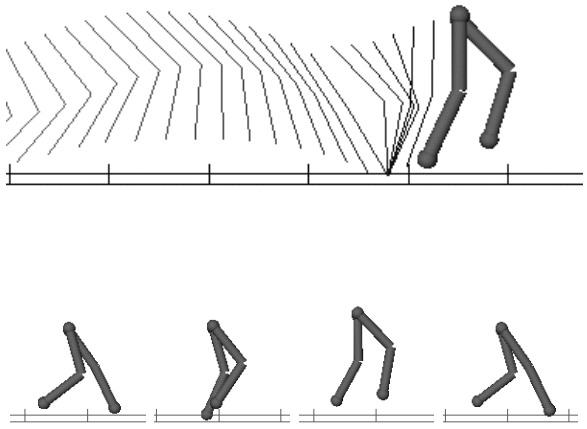

그림 3.7: 두 다리를 가진 뛰뛰기 모델의 동작 $(h = 0.01sec,\ l = 1.5m,\ \dot{x} = 5m/sec,\ \kappa = 9000,\ \theta_0 = \pi/6,\ m = 50kg,\ \alpha = 0.9)$

3.3 상체의 균형 잡기

인간 주행 동작을 사실적으로 표현하기 위해서는 몸이 쓰러지지 않도록 하는 균형 잡기 동작을 표현하여야 한다. 이때 무게 중심의 위치는 균형상태를 판단하는데 유용하게 사용될 수 있는 것으로 무게 중심을 중력방향으로 지면에 내렸을 때에 그 위치가 몸을 지탱하는 점이나 영역 위에 놓인다면 균형이 잘 잡혀있는 상태이다. 따라서 균형의 정도는 그림 3.8과 같이 객체의 무게 중심을 중력방향으로 평면에 내린 위

치와 지탱하는 위치 사이의 거리로 판단할 수가 있다. 따라서 균형을 잡기 위해서는 무게 중심을 지지하는 발의 위치 위로 옮겨주기 위한 제어 방법이 제공되어야 한다.

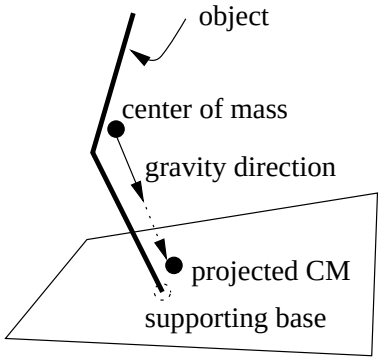

그림 3.8: 균형 잡기 동작의 기본 개념

로보틱스 등의 분야에서 이 균형 잡기 문제는 매우 중요하고도 어려운 문제이며, 객체의 무게 중심이 지지평면과 객체가 만나는 위치의 수직 상공에 0의 속도로 도달할 수 있도록 하는 많은 제어 방법들이 연구되었으며[2, 7], 컴퓨터 그래픽스 분야에서도 활발히 연구되고 있다[1]. 이러한 기법들은 물리적 법칙에 충실하여 매우 사실적인 동작을 생성할 수 있지만, 많은 계산량과 구현의 어려움이 따른다. 로보틱스 분야와 달리 컴퓨터 그래픽스 분야에서는 물리적인 정확성을 요구하는 것이 아니고 사실감이 있는 동작이면 충분하기 때문에, 본 논문은 이러한 균형 잡기를 쉽게 구현할 수 있는 효율적인 방법을 제시한다. 본 논문에서 구현한 균형 잡기 모델은 주행중의 자세를 사실적으로 나타내기 위해 지속적으로 캐릭터의 무게중심을 지탱하는 위치상에 놓으려는 힘을 가하는 방법을 사용한다. 주행중의 상체 움직임은 물리적 관성에 의한 움직임과 무게 중심이 다리에 의해 지탱되도록 움직이는 자발적 운동으로 나뉘어 진다. 관성에 의한 힘은 이전까지의 동작을 통해 계산된 각 관절의 각속도와 각가속도를 이용하여 구할 수 있다. 균형을 잡으려는 자발적 운동을 일으키는 것이 본 논문에서 제시하는 균형 잡기 모델이다.

3.3.1 무게 중심의 위치

주행 도중 캐릭터의 무게 중심이 놓여야 하는 위치는 xz평면상의 위치만 고려하기로 한다. 캐릭터가 무게 중심을 이동시키려고 하는 곳을 목표 무게 중심(CM_{tar})이라 정의한다. 착지 상태에서 목표 무게 중심은 지탱하고 있는 발의 위치($FOOT_i$)로 가정하고, 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.

$$\begin{aligned} CM_{tar} &= FOOT_i \\ &\quad , \; if \; mode \; is \; in \; STANDING \\ CM_{tar} &= (FOOT_i) + (FOOT_a - FOOT_i) \cdot \frac{t}{T} \\ &\quad , \; if \; mode \; is \; in \; JUMPING \end{aligned} \tag{7}$$

이때, t는 도약이 시작된 이후 흐른 시간, T는 도약에 소모되는 총 시간이다. 캐릭터의 실제 무게 중심(CM_cur)은 다음과 같이 구할 수 있다.

$$CM_{cur} = \frac{\sum_i^n M_i \cdot L_i}{\sum_i^n M_i} \tag{8}$$

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

3.3.2 균형 잡기 동작의 생성

그림 3.9와 같이 식 (9)와 (10)을 통해 구한 목표 위치 CM_{tar}와 실제 무게 중심 CM_{cur}의 차를 오차, $\vec{\delta}$로 한다. 즉,

$$\vec{\delta} = CM_{tar} - CM_{cur} \tag{9}$$

이 오차의 x성분을 δ_x, z성분을 δ_z라 할 때 상체의 움직임은 이 δ_x와 δ_z에 따라 결정된다. δ_x가 양수인 경우에는 목표 무게 중심이 실제 무게 중심보다 진행 방향으

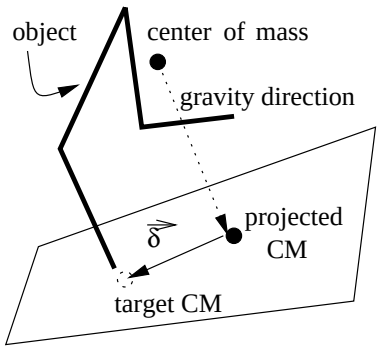

그림 3.9: 목표 무게 중심과 실제 무게 중심의 차이 ($\vec{\delta}$)

로 앞에 있으므로 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.

$$Z_{b_t} = Z_{b_{t-h}} + \frac{(\dot{Z}_{b_{t-h}} + \ddot{Z}_{b_{t-h}} \cdot h) \cdot h + zd_{b_{t-h}}}{2} \tag{10}$$

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_b가 같이 영향을 미쳐 몸통의 z축에 대한 회전각이 변하게 된다. 이러한 균형 잡기 모델의 동작은 그림 3.10과 같이 표현할 수 있다. 시스템은 계속해서 실제 무게 중심과 목표 무게 중심의 차를 계산하여 동작 제어 모듈에 그 값을 주고, 이를 받은 제어 모듈은 이 값을 줄이기 위한 동작 신호를 모델에게 보내 새로운 동작을 생성한다. 이러한 작업을 캐릭터가 움직이는 동안 반복하여 캐릭터의 무게중심을 목표 무게중심에서 멀어지지 않도록 하는 것이다.

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

$$zd_b \quad = \quad \eta \cdot \frac{\delta_x \cdot h}{\dot{x}} \tag{11}$$

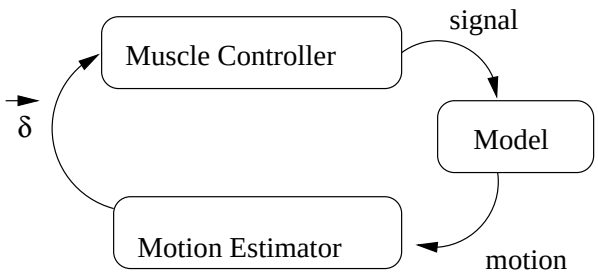

그림 3.10: 균형 잡기의 흐름

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. 따라서, 이 값이 커지면 몸이 휘청거리게 되고, 작아지면 상체를 조금만 움직이며 주행을 하게 된다. 진행 속도 $\dot{x}$로 오차를 나눈 이유는 주행을 할 때 속도가 빠른 경우는 목표 무게 중심이 좌우의 발 사이에서 빠르게 전환되기 때문에 무게중심을 많이 움직이지 않아도 되지만, 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, 다른 관절들도 비슷한 방법을 사용하여 정의할 수 있다. 예를 들어, 몸통이 x축에 대해 일으키는 회전 반응을 의미하는 xd_b는 $xd_b = \eta \cdot (-\delta_z) \cdot h/\dot{x}$로 나타낼 수 있다.

그림 3.11과 3.12는 이러한 방법을 사용하여 주행 중에 몸의 균형을 잡는 동작을 보이고 있다. 그림 3.11는 η가 5, 진행 속도, $\dot{x}$가 1인 경우의 동작을 보이며, 그림 3.12는 $\eta = 5$, $\dot{x} = 5$인 경우를 보이고 있다. 그림에서 볼 수 있는 바와 같이 동일한 η값을 가지는 경우 속도가 느릴 수록 몸을 많이 움직여 균형을 잡는다.

이 균형 잡기 동작을 테스트하기 위해 다리를 저는 동작을 실험하였다. 다리를 저는 동작을 표현하기 위해 저는 다리가 지탱하는 동안 캐릭터의 진행 속도를 증가

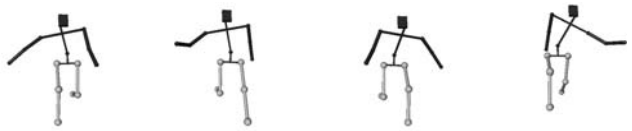

그림 3.11: 균형 잡기 동작 $\eta = 5 \;\; \dot{x} = 1$

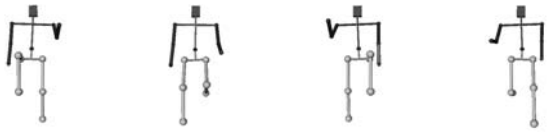

그림 3.12: 균형 잡기 동작 $\eta = 5 \;\; \dot{x} = 5$

시켜 저는 다리가 지탱하는 시간이 짧아지도록 하였다. 이는 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 3.13이다. 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 3.14는 목표 무게 중심

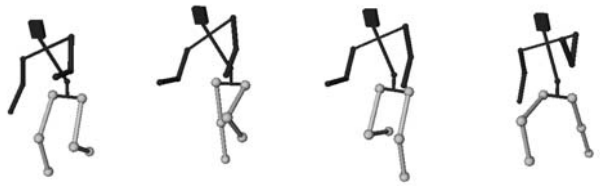

그림 3.13: 다리를 저는 동작에서의 균형 잡기 동작

과 실제 무게 중심의 이동 경로를 보여주고 있다. 점선으로 표시된 것이 목표 무게 중심의 이동 경로이고 실선이 실제 무게 중심이 이동한 경로이다.

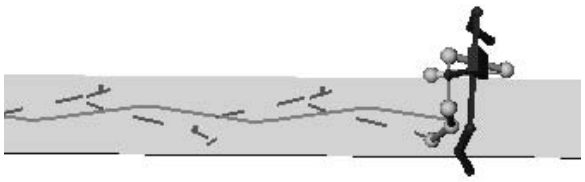

그림 3.14: 무게 중심의 이동 경로 $\eta = 5 \;\; \dot{x} = 2$

3.3.3 균형 잡기 파리미터의 생성

앞에서 설명한 균형 잡기 모델은 모든 관절의 η 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, $(\eta_0, \eta_1, \eta_2, \cdots, \eta_n)$과 같이 각 관절의 각 회전 방향에 대한 η 값들의 집합을 하나의 벡터로 표현하고 이를 유전자의 일부로 사용한다. 따라서 각 관절의 η 값들은 유전자 프로그래밍을 통한 제어 기법에 의해 자동적으로 생성된다.

IV 제어

뛰뛰기 모델과 균형 잡기 모델을 인간 주행 애니메이션에 적용하였다. 이제 이 두 모델이 생성하는 주행 동작을 제어하기 위한 방법이 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) 사이의 연결 정도를 설정하였고, 이 두 가지를 평가할 수 있는 측정 방법을 제공한다.

모든 캐릭터의 동작은 유전자 형태로 표현된다. 각 유전자는 캐릭터의 동작을 제어하는 파라미터들의 집합으로 l, $\dot{x}$, κ, θ_0, m, 등과 같이 한 다리를 가진 뛰뛰기 모델에 사용되는 파라미터와 두 다리를 뛰뛰기 모델을 구현하기 위해 사용되는 α 등의 파라미터와 착지상태에서 활성 다리의 동작을 표현하는 파라미터들, 그리고 η 벡터와 같이 균형 잡기 적용되는 파라미터 등을 포함한다. 사용자가 주행 동작을 제어하기 위해 "동작의 활기참"이라는 개념적 파라미터를 설정하면, 시스템은 유전자 프로그래밍을 통해 이 개념적 파라미터를 만족하는 실제 파라미터들을 찾는다. 이 장에서는 본 논문이 사용하는 유전자 프로그래밍이 어떻게 동작의 활기참이라는 개념적 파라미터로 동작을 제어할 수 있는지를 설명한다.

4.1 유전자의 평가

앞에서 말한 바와 같이 사용자가 개념적 파라미터를 이용하여 동작을 지정하면, 시

스템은 다수의 캐릭터에게 무작위의 유전자를 부여하며, 각 캐릭터는 자신의 유전자에 따라 움직이게 된다. 무작위로 생성된 캐릭터들의 상당수는 물리적으로 불가능한 동작을 보이게 되는데, 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 끝날 때까지 이루어진다. 이 동작이 끝나면 시스템이 이들을 평가하는데, 동작이 얼마나 자연스러운가를 기준으로 평가를 할 수가 있다. 이 때, 고려해야 할 점은 착지 상태에서 도약 상태로 전환될 때 발의 동작이 얼마나 자연스럽게 연결되는가와 전체 주행 동작에서 에너지를 얼마나 낭비 없이 사용하는가이다. 연결도 γ, 에너지 소모율 E_r이라고 할 때, 이들은 다음과 같이 나타낼 수 있다.

$$\begin{aligned} \gamma &= \sum_{i}^{n} \left(\frac{\max\{\dot{\theta}_{f_i}, \dot{\theta}_{s_i} + \ddot{\theta}_{s_i}\}}{\min\{\dot{\theta}_{f_i}, \dot{\theta}_{s_i} + \ddot{\theta}_{s_i}\}} - 1 \right)^2 \\ E_r &= \frac{\int_{t_s}^{t_f} \left(\sum_{i=1}^{n} |\ddot{\theta}_{i_t}|^2 I_i + \tau_t\right) dt}{\int_{t_s}^{t_f} \dot{x}_t dt} \\ &= \frac{\sum_{t=t_s}^{t_f} (\sum_{i=1}^{n} |\ddot{\theta}_{i_t}|^2 I_i + \tau_t)}{x_{t_f} - x_{t_s}} \end{aligned} \tag{12}$$

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 착지상태 마지막 순간의 각속도와 각가속도에 의한 예상 각속도 $(\dot{\theta}_{s_i} + \ddot{\theta}_{s_i})$ 중 큰 값을 작은 값으로 나누어 이 값이 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 시간

에 대해 적분한 뒤 이를 이동 거리로 나누어 구하였다. 즉, 연결도는 착지상태의 마지막 관절 각도, 각속도, 각가속도에 의한 도약상태의 초기 관절 예상값과 실제 도약상태 초기 관절각이 일치할 경우에 0의 값을 가지고, 도약상태 초기의 관절 각도와 착지상태 마지막 순간의 각도, 각속도, 각가속도에 의한 예상 관절 각도의 차이가 클수록 1에 가까워 진다. 또한, 에너지 소모율은 단위 거리를 움직이는데 소모하는 에너지의 양을 나타낸 것이다.

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. 이 평가 함수는 γ의 값과 E_r의 값이 0에 가까워질수록 $\frac{1}{2(1+\gamma)}$과 $\frac{1}{2(1+E_r)}$이 각각 1/2에 가까워지므로 0에 가까워지는 함수이다. 따라서 이 평가 함수를 최소화함으로써 연결도 γ와 에너지 소모량 E_r을 최소화할 수 있다.

$$eval(\gamma, E_r) = 1 - \frac{1}{2(1+\gamma)} - \frac{1}{2(1+E_r)} \tag{13}$$

이 $eval(\gamma, E_r)$의 값은 γ와 E_r이 0일 때 최소값 0을 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 나타낸

다. 이러한 다양한 동작을 "활기참"이라는 파라미터를 이용해 생성하기 위해 에너지 소모율을 조정하는 방법을 사용하였다. 즉, 주어진 제한 조건을 만족하면서 최소의 에너지 소모율로 동작하는 것이 아니라 에너지 소모율을 조정할 수 있게 하여 동작의 활기찬 정도를 제어할 수 있게 하였다. 이것은 활기찬 동작이 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($0 \leq \beta < 1$). 앞에서 계산한 E_r 값을 최소화하지 않고, 이 β를 사용하여 E_r이 수렴되는 위치를 제어할 수 있도록 하였다. 따라서 앞 절에서 설명한 평가 함수 $eval(\gamma, E_r)$은 다음과 같이 $eval(\gamma, E)$로 수정된다.

$$\begin{aligned} eval(\gamma, E) &= 1 - \frac{1}{2(1+\gamma)} - \frac{1}{2(1+E)} \\ & where\ E = \left(\frac{\beta}{1-\beta} - E_r\right)^2 \end{aligned} \tag{14}$$

여기서 E는 이전 $eval(\gamma, E_r)$ 함수의 에너지 소모율 E_r을 대신하는 것으로 새로운 평가 함수는 γ와 이 E 값을 최소화하기 위한 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 평가함수를 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.

V 실험 결과

그림 5.1: 속도에 따른 동작 1 ($\dot{x} = 2m/sec, \quad \eta = 5, \quad \beta = 0$)

그림 5.2: 속도에 따른 동작 2 ($\dot{x} = 5m/sec, \quad \eta = 5, \quad \beta = 0$)

그림 5.3: 속도에 따른 동작 3 ($\dot{x} = 7m/sec, \quad \eta = 5, \quad \beta = 0$)

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

그림 5.1, 그림 5.2, 그림 5.3는 각각 속도를 $\dot{x} = 2m/sec$, $\dot{x} = 5m/sec$, $\dot{x} = 7m/sec$로 지정하고 에너지 소모율 E_r과 연결도 오차 γ를 최소하여 얻은 최적 동작이다.

그림 5.4: β를 이용한 동작 생성 (보통 주행) ($\beta = 0.2 \quad \dot{x} = 10m/sec$)

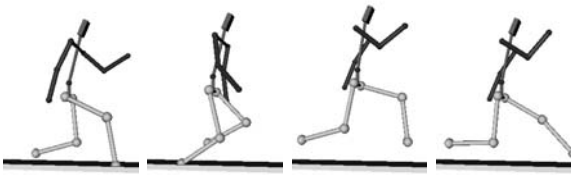

그림 5.5: β를 이용한 동작 생성 (활기찬 주행) ($\beta = 0.7 \quad \dot{x} = 10m/sec$)

그림 5.4과 그림 5.5는 속도는 동일하게 ($\dot{x} = 10m/sec$) 지정하고 β 값을 다르게 하여 동작을 생성한 예를 보이고 있다. 그림 5.4은 β 값을 0.2로 설정하였고, 그림 5.5는 β 값을 0.7로 설정한 것이다. 그림에서 볼 수 있듯이 β 값을 1에 가까이 할 수록 더욱 활발한 동작이 생성된다.

그림 5.6는 본 논문의 기법을 적용한 주행 동작 생성 프로그램을 통해 주행 동작을 생성한 결과로 주행 경로와 완주하는데 걸리는 시간을 지정하고, 활기참의 정도를 0으로 하여 에너지를 최적화하도록 한 결과다.

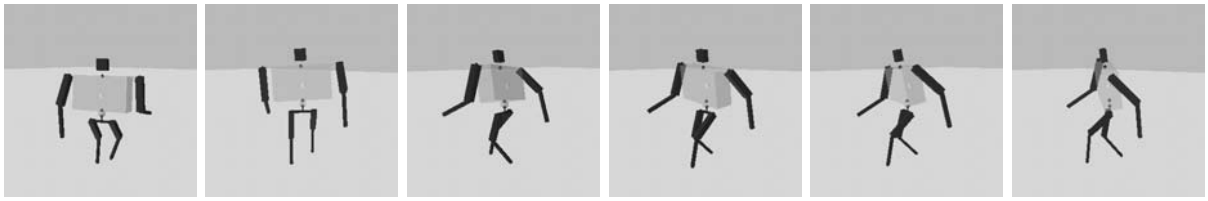

그림 5.6: 생성한 주행 동작의 예

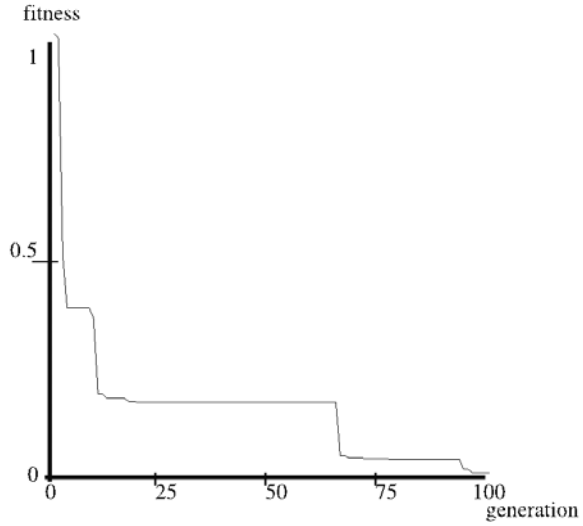

그림 5.7: 가장 우수한 평가를 받은 유전자의 세대별 평가값 변화(100 개체 100 세대)

그림 5.7는 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 5.7에 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.

VI 결론 및 향후 연구과제

본 논문을 통해 캐릭터 모델이 소모하는 에너지를 제어하여 자동적으로 인간의 주행 동작 애니메이션을 생성하는 효율적인 기법을 제시하였다. 이러한 동작 생성 기법을 위해 본 논문에서는 물리적 파라미터로 제어할 수 있는 뛰뛰기 모델과 균형 잡기 기법을 구현하여 주행 동작의 기본 모델로 삼았다. 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 에너지 소모율을 최소화하는 대신 에너지의 소모를 조절할 수 있도록 하였으며, 이러한 기법을 통해 "활기찬 주행"과 같은 감정적 상태에 따라 동작을 제어할 수 있다. 이러한 제어 기법의 기본적인 아이디어는 캐릭터 모델의 동작이 에너지 소모에 의해 지배된다는 것이다. 이 기법을 실제 애니메이션에 적용할 경우 애니메이터는 캐릭터의 각 관절의 움직임을 일일이 지정하지 않고도 자신이 지정한 속도로 주행하며 원하는 감정적 상태로 움직이도록 캐릭터를 제어할 수 있다. 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 사용했던 동작 데이터가 필요하지 않다.
- 다양한 동작 모델을 사용할 경우, 모션 캡쳐로는 동작 생성이 불가능한 가상 생물이나 의인화된 무생물 등의 동작을 생성할 수 있다.
- 모션 캡쳐 기법은 새로운 동작이 필요할 때마다 다시 실험을 하여 데이터를 얻

어야 하지만, 본 논문의 기법은 활기참이라는 개념적 파라미터를 통해 다양한 동작을 생성할 수 있다.

- 원하는 동작을 생성하는 최적의 유전자가 생성되면, 이 유전자를 캐릭터 모델에 적용하여 실시간에 그 동작을 재생할 수 있다.
- 쉽게 구현 가능한 균형 잡기 모델을 제시하여 동작의 사실성을 높였다.

본 논문의 기법은 인간 모델의 주행만을 고려하였다. 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 제어 기법을 적용하는 것도 앞으로의 연구 과제이다. 또한 본 논문에서 제시한 균형 잡기 모델을 다른 여러 동작이나 캐릭터에 적용하고 개선하는 것도 앞으로의 연구 과제이다.

참고문헌

[1] R. Boulic and Ramon Mas. Hierarchical kinematic behaviors for complex articulated figures. In *Interactive Computer Animation*, 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] C. François and C. Samson. Running with constant energy. *Proceedings of the IEEE International Conference on Robotics and Automation*, 1:131–136, May. 1994.

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

[5] R. Grzeszczuk, D. Terzopoulos, and G. Hinton. Neuroanimator: Fast neural network emulation and control of physic-based models. *Proceedings of SIGGRAPH '98*, pages 9–20, Jul. 1998.

[6] 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.

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

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

[9] 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.

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

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

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