공학석사학위논문

비인간형 캐릭터에 대한 실시간 조종

Online Motion Puppetry for Non-human Characters

2014년 8월

서울대학교 대학원
전기·컴퓨터 공학부
윤민지

초록

동작 센서의 성능이 점점 뛰어나짐에 따라 실시간으로 캡처된 3차원 동작데이터를 이용한 응용분야가 넓어지고 있다. 가상 캐릭터 조종 문제는 그 중의 하나로, 동작 센서 앞에서 사용자가 특정 동작을 수행했을 때, 실시간으로 가상 캐릭터가 같은 의미의 동작을 수행하도록 조종하는 문제이다. 기존의 연구들은 비인간형 캐릭터가 사용자의 의도에 맞는 동작을 수행하도록 조종하는데 성공했으나 캐릭터의 동작이 고유의 동작 패턴을 유지하지 못한 채 이루어졌다. 예를 들어 4족보행하는 코끼리는 고유의 패턴을 유지하지 못한 채 2족보행하는 사람의 패턴에 맞춰 걷게 된다. 이렇게 되면 캐릭터 동작이 부자연스러워 보일 뿐 아니라 동작 흐름에 끊김이 생긴다. 이 연구에서는 캐릭터 고유의 동작 패턴이 유지되는 새로운 비인간형 캐릭터 조종 알고리즘을 제안한다. 고유의 동작 패턴은 주기적 동작으로 표현된다는 점을 이용하여 주기적 동작과 비주기적 동작으로 나누어 서로 다른 매핑을 수행해준다. 비주기적 동작에 대해서는 자세 특징 벡터를 기반으로 한 자세 매핑을, 주기적 동작에 대해서는 동작 특징 벡터를 기반으로 한 동작 매핑을 한다. 그 결과 사람과 캐릭터간의 동작 패턴이 다름에도 이를 유지하는 캐릭터 조종이 가능해진다.

주요어 : 캐릭터 조종, 비인간형 캐릭터, 실시간, 애니메이션, 동작

학번 : 2012-23226

목차

그림 목차

표 목차

제 1 장 서론

동작 센서의 성능이 점점 뛰어나짐에 따라 실시간으로 캡처된 3차원 동작데이터를 이용한 응용분야가 넓어지고 있다. 예를 들어, 마이크로소프트 키넥트 동작 센서는 사용자가 특정 동작을 수행했을 때, 실시간으로 사람의 각 관절 3차원 상 위치 값을 계산해준다. 가상 캐릭터 조종 문제는 계산된 위치 값을 이용하여 캐릭터의 동작을 실시간으로 조종하는 것이다. 기존의 연구들은 사람과 비슷한 골격 구조를 가진 캐릭터에 한해서 사람 동작 데이터를 캐릭터에 단순히 매핑시켜 조종 해왔다. 그렇다면 비인간형, 즉 인간과 골격 구조가 다른 캐릭터들은 어떻게 사람 동작 데이터를 이용하여 조종 할 수 있을까? 예를 들면 거미와 같은 경우 다른 개수의 관절을 가지기 때문에 단순한 매핑을 통한 조종은 어렵다. 왜냐하면 사람 동작데이터를 비인간형 캐릭터에게 리타겟팅(retargeting)하는 것이 애니메이터에게 어마어마한 양의 작업을 요구하기 때문이다. 또한 리타겟팅은 임의의 동작에 대해서는 매핑할 방법이 없다는 한계가 있다.

현재 존재하는 비인간형 캐릭터 컨트롤 알고리즘 중에는 특징 벡터 매핑(Feature mapping)과 동작 데이터 커플링(motion coupling)의 조합으로 접근[1]하거나 3차원 데이터(arbitrary 3d point source sequence)와 메쉬 정보(mesh target sequence) 사이의 매핑을 학습함으로써 캐릭터를 실시간 조종하는 접근[2]이 있었다. 이러한 알고리즘들은 사람이 수행하는 동작의 의미에 맞게 비인간

형 캐릭터들을 성공적으로 컨트롤했다. 하지만 수행하는 동작에 대한 캐릭터 고유의 패턴은 유지하지 못하였다. 예를 들어 4족보행하는 코끼리는 고유의 패턴을 유지하지 못한 채 2족보행하는 사람의 패턴에 맞춰 걷게 된다. 이렇게 되면 캐릭터 동작이 부자연스러워 보일 뿐 아니라 동작의 흐름에 끊김이 생긴다.

이 연구에서는 신체 구조뿐만 아니라 캐릭터 고유의 동작 패턴이 유지되는 가상 캐릭터 조종 알고리즘을 제안한다. 연구의 가장 기본이 되는 아이디어는 캐릭터 고유의 동작 패턴은 주기적 동작으로 표현된다는 점에서 출발했다. 즉 동작 패턴을 유지하기 위해서는 주기적 동작의 주기성을 유지해주어야 하는 것이다. 이를 위해 주기적 동작과 비주기적 동작을 구분하여 다르게 처리한다. 주기적 동작의 경우, 자세 단위로 매핑을 처리한 이전 연구들과 달리 동작 단위로 특징 벡터(feature)를 찾아내서 캐릭터의 주기적 동작으로 매핑 한다. 비주기적 동작의 경우 자세 단위로 매핑을 처리한다.

알고리즘은 미리 필요한 정보를 계산해 놓는 전처리 작업 단계와 사용자가 동작하는 동안 계산이 이루어지는 실시간 작업 단계를 통해 수행된다. 전처리 작업 단계는 비주기적 동작을 위한 자세 매핑을 찾는 과정과 주기적 동작을 위한 동작 매핑을 찾는 과정으로 나뉜다. 비주기적 동작을 위한 자세 매핑은 Seol et al.[1]의 연구를 참고한다. 신체구조가 서로 다른 사람과 캐릭터 사이를 가장 효율적으로 매핑해주는 특징벡터를 자세 단위에서 선택한 뒤 캐릭터 특징벡터와의 매핑함수를 계산해준다. 이때, 선택된 자세 단위의 특징벡터를 기본 동작 단위로 합쳐서 동작 특징 벡터를 만든다. 만들어진 동작 특징 벡터는 주기적 동작을 위한 동작 매핑에서 사용

된다. 동작 매핑을 위한 첫 번째 단계에서는 서포트 벡터 머신(Support Vector Machine)을 이용하여 동작 분류기를 생성한다. 동작 분류기에 임의의 동작을 넣으면 가장 유사한 주기적 동작을 알려준다. 두 번째 단계에서는 캐릭터 데이터에 대한 동작 그래프(motion graph)를 생성한다. 동작 그래프란 주어진 동작 이외의 동작을 생성할 수 있게 새로운 이동 경로를 제공하는 일종의 그래프이다. 기존의 동작 그래프와는 다르게 각 꼭짓점을 한 프레임이 아닌 단위 동작으로 한다.

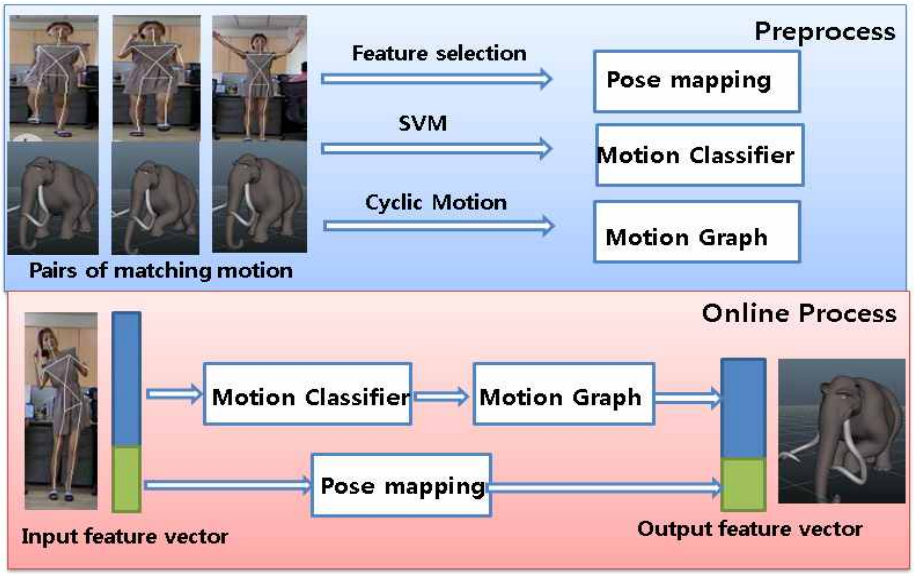

그림 1.1 전체 시스템 작동 방식

실시간 작업 단계에서는 먼저 동작 센서 앞에서 사람이 수행하는 일련의 동작을 기본 동작 단위로 끊어 동작 특징 벡터를 만들어낸다. 벡터 값을 앞서 전처리 단계에서 만들어둔 사람 동작 분류기에 넣어주면 의미적으로 가장 가까운 주기적 동작을 알려준다.

분류기의 결과 값을 바탕으로 캐릭터의 동작 그래프 내 현 꼭짓점에서 이동 할 수 있는 이동 경로 중 해당 주기적 동작을 수행할 확률이 가장 높은 꼭짓점으로 이동한다. 사람의 동작을 기준으로 동작 그래프 내에서 의미가 맞는 방향으로 탐색해 나가면 원하는 캐릭터 동작이 고유의 동작 패턴을 유지하면서 결과로 나오게 된다. 여기까지 캐릭터 고유의 동작 패턴을 지키기 위한 주기적 동작에 대한 처리가 끝났다. 마지막으로 주기적 동작과 관련 없는 신체 부위에 대해서 전처리 단계에서 만든 자세 매핑 함수를 이용해서 비주기적 동작을 만들어 앞서 만든 주기적 동작과 합쳐주면 최종 동작이 완성된다.

이 알고리즘을 통해 이족보행의 사람과 다족보행의 캐릭터 간의 보행 패턴이나 속도가 다름에도 이를 유지하는 동작 조종이 가능해진다. 그 결과로, 이전 실시간 캐릭터 조종 알고리즘들에 비해 캐릭터가 훨씬 다양한 동작을 끊김 없이 자연스럽게 수행할 수 있게 된다.

제 2장 관련 연구

동작 센서의 발달에 맞춰 다양한 동작 조종 연구들이 진행되어 왔다. Shiratori 와 Hodgins은 신체 부위의 가속을 캡처하는 동작 센서를 이용해 운동 컨트롤러를 생성했고[3] Liu는 다양한 온몸 애니메이션을 생성[4]해냈다. Lockwood와 Singh는 마이크로소프트 서페이스(Micorosoft Surface) 터치스크린 장치 위에서 움직이는 손가락의 움직임에서 특징 값들을 추출한 뒤 분류함으로써, 캐릭터 운동으로 변환시켰다[5]. 이러한 연구들은 입력 데이터로부터 동작을 가장 잘 묘사해내는 특징 값들을 추출해내는 것이 주요 문제였다. 비행시간(TOF, Time-Of-Flight) 카메라로부터 얻어낸 깊이 데이터는 사용자의 자세를 알아내거나[6] 얼굴 애니메이션을 생성하거나 조종알고리즘을 개발하는데[7] 사용되어 왔다. 최근에는 chen et al [8]이 키넥트로부터 사람 자세를 입력 값으로 받아 다양한 기하학적 구조들을 조종하는 시스템을 선보였다. 그들의 주요 목표는 사람의 자세 입력 값을 직접 위치 제약조건으로 사용하는 메쉬(mesh) 변형 방법을 개발하는 것이었다. 하지만 메쉬 변형이 사람의 신체 구조를 본떠 이루어지기 때문에 메쉬의 동작 패턴은 사람의 동작 패턴과 동일한 선에서만 이루어진다.

신체구조와 동작 패턴이 사람과 다른 가상의 캐릭터를 실시간으로 조종하기 위한 새로운 연구들이 진행되어 왔다. 최근에 제시된 방법론으로는 자세 매핑을 기반으로 하는 Seol et al.[1]의 연구

와 동작 매핑을 기반으로 하는 Helge et al[2]의 연구가 있다.

먼저 이 연구의 기반이 되는 Seol의 알고리즘은 입력된 사람의 동작을 여러 가지 다른 매핑을 기반으로 미리 저장해둔 동작과 합성하여 캐릭터 동작을 생성해 냈다. Seol의 연구에서는 캐릭터의 동작을 간단한 동작과 복잡한 동작으로 나누어서 접근하였다. 몸을 양쪽으로 흔드는 동작같이 캐릭터의 각 부위를 사람이 쉽게 흉내 낼 수 있는 동작들을 간단한 동작으로 분류하고 다족 보행과 같이 사람이 흉내 내기 어려운 동작을 복잡한 동작으로 분류한다. 입력된 사람의 자세 단위로 특징 벡터를 만들어낸 뒤 캐릭터의 자세로 매핑 하여 간단한 동작을 만들어낸다. 이 때, 매핑 매트릭스는 여러 특징 값들 중에 캐릭터 특징 값을 가장 잘 표현할 수 있는 값들을 선택하도록 학습을 통해 만들어진다. 복잡한 동작은 미리 디자인된 동작 쌍을 바탕으로 동작 커플링을 통해 만들어낸다. 간단한 동작과 복잡한 동작을 합성하여 최종 동작을 생성한다. 그 결과, 랜덤한 사람의 동작으로부터 랜덤한 캐릭터 애니메이션을 생성할 수 있다. Seol은 입력 데이터의 3차원 위치 값만을 사용한 게 아니라 그 값으로부터 여러 특징 값들을 뽑아 내여 특징벡터-특징벡터 매핑을 하였기 때문에 동작의 다양성을 처리할 수 있었다. Helge의 연구는 데이터 공간 단위의 매핑을 통해 동작 매핑을 처리하였다. Helge는 캐릭터의 고유 패턴을 유지하는 알고리즘을 제안했지만 이 알고리즘은 디자이너가 만들어놓은 캐릭터 동작만을 조종할 수 있다는 한계점을 가지고 있다. 즉 캐릭터는 미리 만들어놓은 동작 외의 다양한 동작은 수행할 수 없다.

우리의 연구에서는 위의 두 연구의 단점들을 극복하고자 한

다. 동작 단위의 매핑을 통해 캐릭터가 고유의 동작 패턴을 지키면서 미리 저장해둔 동작 외의 새롭고 다양한 동작을 수행할 수 있도록 한다.

제 3장 데이터 구성

동작 센서는 사람의 몸에 부착된 15개의 마커(머리, 목, 몸통, 왼/오른 어깨, 팔꿈치, 손, 골반, 무릎, 발)의 3차원상의 위치 값을 알려준다. 출력된 위치 값들로부터 캐릭터의 동작을 효과적으로 표현해줄 수 있는 여러 가지 특징 값들을 뽑아낸다. 특징 값들로는 각 마커의 절대 위치 값, 상대 위치 값, 각도, 사이 거리, 속도 등이 선택될 수 있다. 또한, 사람과 비인간형 캐릭터 동작 간의 비선형 관계를 표현해주기 위해 제곱, 지수 승, 내적과 같은 여러 가지 커널 함수들이 특징 값을 계산하기 위해 사용된다. 표 3.1에서는 사용된 특징 값들의 종류와 각각의 자유도를 보여준다.

Features	Number of DOFs
Absolute position	3 (x, y, z)
Root-coordinate position	3 (x, y, z)
Distance from root joint	1
Displacement from parent joint	3 (x, y, z)
Ground height	1
Joint angle at parent marker	1
Displacement from end effectors	15 (3×5 end effectors)
Distance from end effectors	5 (for 5 end effectors)
Velocity	3 (x, y, z)
Acceleration	3 (x, y, z)
Square of above features	38
Exponential of above features	38
Dot product of two vector features	45
Multiplication of two scalar features	28

표 3.1 특징 벡터 매핑 과정에서 사용되는 모든 종류의 특징 값. 사람의 몸에 부착된 각각의 마커마다 위의 특징 값들을 가진다.

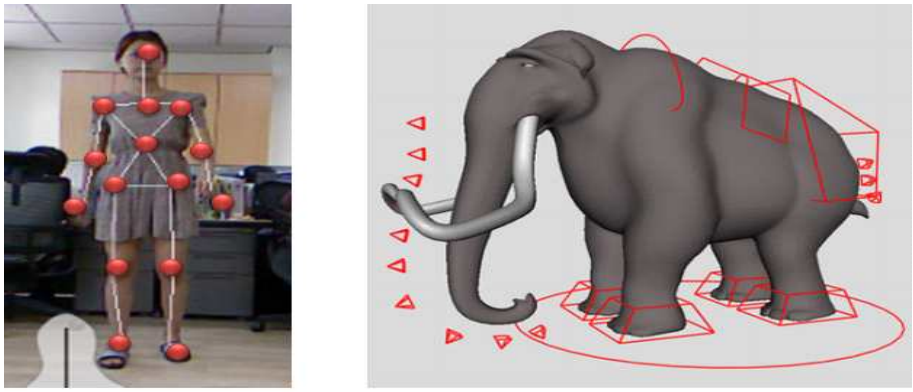

그림 3.1 (a) 사용자의 몸에 있는 15개 마커의 위치 (b) 조종 손잡이(빨간 줄)를 가진 캐릭터 리그

가상 캐릭터 조종을 위해 우리는 메쉬(mesh), 뼈대, 스키닝 계수(skinning weights), 키프레이밍(key framing)을 위한 조종 손잡이(control handles)로 이루어진 캐릭터 리그(character rig)를 사용한다. 캐릭터의 특징 벡터는 조종손잡이의 자유도(Degree Of Freedom) 값으로 구성된다. 조종 손잡이의 자유도 값을 특징 벡터로 선택하여 캐릭터 조종용으로 사용하는 것은 실시간 조종뿐 아니라 애니메이터가 결과물 애니메이션을 보정할 때에도 훨씬 효율적이다. 본 연구에서는 각 조종 손잡이 자유도 값의 범위 정보가 필요하다. 이는 스키닝 품질을 확인하는 과정에서 항상 생성되는 캘리스데닉스(calisthenics)라는 데이터를 통해 쉽게 구할 수 있다. 캘리스데닉스로부터 N개의 조종 손잡이 자유도($y_i, i = 1...N$) 각각의 최대, 최소 가동범위 정보($y_i^l < y_i < y_i^u, l \text{ for } lower \text{ and } u \text{ for } upper$)를 알아낸다. 이 때, 사람 데이터에 대해서도 15개의 마커 각각의 동작 범위를 키넥트를 이용하여 캡처해둔다.

캐릭터에 대해서 M개의 주기적 키프레임 애니메이션을 손으로 만들어준다. 예를 들어 코끼리의 4족보행의 한 주기에 대해 키프레임 애니메이션을 만들어준다. 그 다음 M개의 주기적 캐릭터 동작과 각각 같은 의미를 지니는 사람의 동작을 키넥트를 이용하여 캡처한다. 즉, M 쌍의 사람-캐릭터 주기적 동작을 만들어준다. 그 후, 자유도 태깅(DOF tagging)작업을 수행한다. 자유도 태깅이란 캐릭터 조종손잡이의 자유도 각각에 대해서 사람 몸에 부착된 마커들 중 가장 의미적으로 가까운 마커들을 짝지어주는 작업이다. 자유도 태깅은 위의 M쌍의 대응 동작들을 바탕으로 이루어진다.

Type	**Amount of manual work**
Range of motion	for each active DOF
DOF tagging	for each control handle
Complex cyclical animation	M animations (usually 2-3)

표 3.2 조종할 캐릭터에 대해서 디자이너에게 요구되는 작업

제 4장 비주기적 동작을 위한 자세 매핑

이 장에서는 제 3장에서 획득한 캐릭터의 키프레임 애니메이션과 사용자 동작 데이터를 바탕으로 자세 특징 벡터 매핑이 학습된다. 자세 특징 벡터 매핑 학습과정은 Seol et al[1]의 연구를 참고하였다. 자세 특징 벡터 매핑은 $y = Ax + b$로 나타내어진다. 여기서 $y = \{y_i\}$란 조종 손잡이 자유도 값으로 이루어진 캐릭터의 특징 벡터이고 $x = \{x_j\}$는 표 3.1에서 보여 졌던 특징 값으로 이루어진 입력 자세 특징 벡터이다. A는 포즈 변환 매트릭스이고 b는 상쇄 벡터이다. β_{ij}를 특징 값 각각의 가중치라고 할 때, 각 출력 특징 값 y_i는 $y_i = \sum_j \beta_{ij}(a_{ij}x_j + b_{ij})$ 와 같이 선형식의 가중치 결합으로 정의된다. 가중치 값이 클수록 해당 특징 값이 캐릭터의 특징 값을 잘 표현하는 것이라고 할 수 있다.

다양한 입력 특징 값들 중에서 출력 특징 값을 가장 잘 설명하는 최상의 결합을 찾을 것이다. 각각의 자유도 y_i에 대해서 우리는 간단한 동작범위 애니메이션과 대응되는 사람의 동작 데이터를 제 3장에서 얻었다. 이 데이터들을 바탕으로 각 자유도에 대해서 자유도 태깅을 통해 선택된 마커들의 특징 값들인 모든 $\{x_j\}$에 대해 식 4.1을 계산한다. 식 4.1은 4가지 종류의 오차항의 합을 계산하고 있다. 각각 오차항의 의미와 계산 과정에 대해 아래 설명한다. 4가지 종류의 오차 항에 대한 계수는 여러 차례의 실험 결과 $w_1 = 10, w_2 = 2.5, w_3 = 1, w_4 = 10$으로 정한다.

$$E = w_1 E_{ij}^{MM} + w_2 E_{ij}^{AO} + w_3 E_{ij}^{SD} + w_4 E_{ij}^{ST} \tag{4.1}$$

E^{MM}은 최대·최솟값 매칭 오차(Min-max value matching)를 뜻한다. 이 오차 항은 입력 특징 벡터와 출력 특징 벡터의 최대·최소·평균값이 얼마나 잘 매칭 되는지를 측정한다. 자유도의 최대·최소·평균값인 y_i^l, y_i^u, y_i^r와 입력 특징의 최대·최소·평균값인 x_i^l, x_i^u, x_i^r를 기반으로 세 데이터 포인트$(x_j^l, y_i^l), (x_j^u, y_i^u), (x_j^r, y_i^r)$에 대한 선형 근사를 수행한다. 그 다음 식 4.2와 같이, 선형 근사를 통해 얻은 a_{ij} 와 b_{ij}값을 사용하여 자유도 y_i의 가동 범위로 정규화 해주면 E_{ij}^{MM}을 구할 수 있다.

$$E_{ij}^{MM} = \frac{1}{y_i^u - y_i^l}(\| y_i^r - (a_{ij}x_j^r + b_{ij})\|^2 + \| y_i^l - (a_{ij}x_j^l + b_{ij})\|^2 + \| y_i^u - (a_{ij}x_j^u + b_{ij})\|^2) \tag{4.2}$$

E^{AO}는 절대 방위 오차(Absolute orientation)을 뜻한다. 사람들은 어떤 캐릭터의 동작을 흉내 낼 때, 세계 좌표계 내에서 캐릭터가 움직이는 방향과 비슷하게 움직이는 경향이 있다. E^{AO}은 이러한 경향성을 반영하기 위한 오차 항으로, 입력 특징 값과 자유도의 동작 방향 유사도를 측정했다. 식 4.3에서 $\hat{y_i}$는 자유도 y_i값에 의해 움직이는 캐릭터 신체 부위의 운동 방위이고 $\hat{x_j}$는 해당 입력 특징 값의 세계 좌표 방향이다.

$$E_{ij}^{AO} = 1 - |\hat{x_j} \bullet \hat{y_i}| \tag{4.3}$$

E^{SD}는 표준편차(Standard deviation) 오차 항이다. 가장 큰 동작 변화량을 가진 특징 값일수록 사용자의 동작 의도를 효과적으로 반영하기 때문에 각 특징 값들의 표준편차는 입력-출력 매칭에 있어서 중요한 요소이다. 식 4.4에서 σ_j는 x_j의 표준편차를 뜻한다.

$$E_{ij}^{SD} = \exp(-\sigma_j) \tag{4.4}$$

E^{ST}는 동작 변이 오차 항(Smooth transitions)이다. 이 항은 주기적 동작에 연관된 자유도에 한해서 직접 특징 매핑에서 만들어지는 동작과 디자인해놓은 동작간의 자연스러운 변이 정도를 측정한다. 두 동작간의 유사성은 직접 특징 매핑의 결과가 캐릭터의 자연스러운 동작을 나타낼 확률이 높다는 것을 말하기 때문에 중요한 정보이다. 식 4.5에서 M은 주기적 동작의 프레임 개수이고 a_{ij}와 b_{ij}는 식 4.2에서 사용한 값이고 c_{ij}와 d_{ij}는 주기적 동작 내 있는 M개의 데이터 $\{x_j^k, y_j^k\}$에 선형 근사를 적용함으로써 얻어낸 계수이다.

$$E_{ij}^{ST} = \frac{\sum_{k=1}^{M} (\|y_i^k - (a_{ij}x_j^k + b_{ij})\|^2 + \|y_i^k - (c_{ij}x_j^k + d_{ij})\|^2)}{(y_i^u - y_i^l)M} \tag{4.5}$$

각각의 자유도 y_i에 대해서 E값을 최소화하는 특징 값들을 선택한다. 이때 최소 오차 값 $\hat{E}$를 가지는 특징 값을 x^{y_i}라고 하면 오차 값이 $(1+\epsilon)\hat{E}$보다 작은 모든 특징 값들$\{x_j^{y_i}\}$을 선택해준다. ϵ값은 여러 차례의 실험을 통해 0.2로 사용한다. 선택된 특징 값$\{x_j^{y_i}\}$에

대한 최적 계수 $\{\beta_{ij}\}$는 식 4.6에 의해 입력-출력 데이터간의 매칭 오차를 최소화시키는 방향으로 정해진다. 이 때 사용되는 데이터는 E^{MM} 계산 시 사용된 최대·최소·평균 자세 데이터 쌍과 E^{ST} 계산 시 사용된 주기적 동작 데이터 쌍이다. $\{\alpha_k\}$는 각 데이터 쌍의 중요도를 나타내는 계수이다. 평균 자세 데이터의 경우 $\alpha_k = 1$, 최대·최소 데이터의 경우 $\alpha_k = 0.3$, 주기적 동작 데이터의 경우 $\alpha_k = 0.05$로 설정한다. 식 4.6은 2차 계획법(Quadratic Programming)을 이용해서 최적화시켜준다.

$$\min \| \sum_k \alpha_k (y_k - \sum_j \beta_{ij} (a_{ij} x_j^{y_i} + b_{ij})) \|^2 \; s.t. \; \sum_j \beta_{ij} = 1, \beta_{ij} \geq 0 \qquad (4.6)$$

위의 계산 과정을 거쳐서 우리는 자세 매핑을 위한 매핑 매트릭스 A와 상쇄 벡터 b를 구해낸다. 계산된 자세 특징 벡터 매핑은 실시간 처리 단계에서 비주기적 동작을 생성할 때 사용된다.

제 5장 주기적 동작을 위한 동작 매핑

5.1 사람 동작 분류기 생성

동작 매핑의 첫 번째 단계에서는 서포트 벡터 머신을 기반으로 한 동작 분류기를 생성한다. 동작 센서를 통해 입력된 사람 동작이 데이터 구성 단계에서 만들어놓은 여러 종류의 주기적 동작들 중 어느 동작에 해당하는지 구분하기 위한 분류기이다.

먼저 주기적 동작들을 각각의 동작 이름과 함께 기본 동작 단위인 L 프레임으로 나누어 준다. 예를 들어 기본 동작 단위를 10 프레임이라고 했을 때, 30 프레임의 걷는 동작은 '걷기' 이름과 함께 3개의 단위 동작으로 나누어진다. 그 다음, 단위 동작마다 각각의 동작 특징 벡터를 만든다. 동작 특징 벡터는 앞서 제 4장에서 선택된 자세 특징 벡터를 각 프레임마다 만들어준 뒤, 단위 동작 단위로 합친 것이다. 여러 차례의 실험 결과, 동작 특징 벡터 내의 여러 가지 특징 값들 중 몸통 기반 위치, 속도, 가속도, 이 세 가지 특징 값만으로도 효과적으로 동작을 분류 가능함을 알아냈다. 동작 분류기는 계산 로드를 줄이기 위해 위의 세 특징 값만을 기반으로 한다. 제 3장에서 저장된 M가지 종류의 주기적 동작을 나타내는 M개의 동작 이름 외에 1개의 이름이 분류기에 추가된다. 입력된 사용자 동작이 미리 저장된 주기적 동작들 중 어느 하나와도 매칭 되지 않을 때, 이를 비주기적이라 판단하고 따로 분류하기 위한 이름이다. 비주기적 이름을 가진 데이터는 배경 데이터라고 부르고 기존 M개의

주기적 동작들에 노이즈를 추가함으로써 생성해낸다. 각각의 동작 이름을 가진 데이터들을 이용하여 멀티 클래스 서포트 벡터 머신 (multi-class SVM) 분류기를 학습한다. 배경 데이터를 생성하는 과정에서 노이즈 추가가 랜덤으로 이루어지기 때문에, 주기적 동작 데이터 값과 큰 차이가 나지 않는 경우가 생겨 분류가 잘 이루어지지 않을 꺼라 생각할 수 있다. 하지만 데이터의 차원수가 높아 원본 데이터와 비슷한 값을 생성해낼 확률이 매우 낮고 서포트 벡터 머신은 노이즈에 강하기 때문에 생성된 동작 분류기는 효과적으로 작동한다.

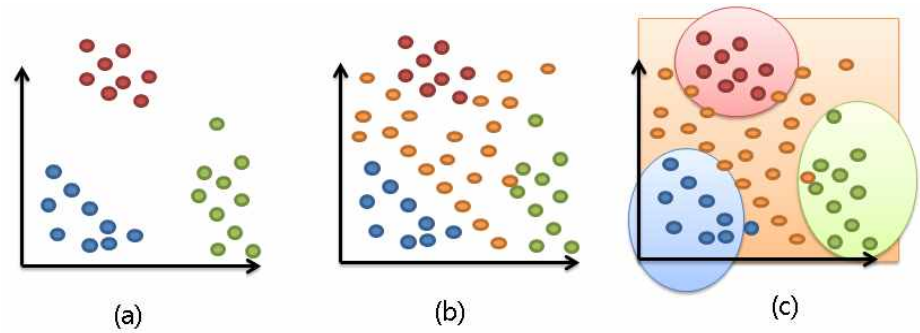

그림 5.1 (a) 저장된 주기적 동작 데이터 (b) 무작위 노이즈를 더한 배경 데이터를 추가 (c) SVM 학습 결과

5.2 캐릭터 동작 그래프 생성

동작 매핑의 두 번째 단계에서는 캐릭터에 대한 동작 그래프를 생성한다. 동작 그래프란 주어진 동작 데이터 이외의 다양한 동작들을 생성할 수 있게 도와주는 일종의 그래프이다. 기존의 동작

그래프는 동작 데이터 내 각각의 프레임들을 꼭짓점으로 가지고 자연스럽게 이어질 수 있는 프레임들 사이에만 이동 경로를 가진다. 자연스럽게 이어지는지 여부는 각 프레임 내 저장된 자세의 유사성으로 계산한다. 따라서 동작 그래프를 시간에 따라 탐색해 나가면 자연스럽게 이어지는 새로운 동작 흐름이 나오게 된다. 이 연구에서 사용할 동작 그래프는 앞서 설명한 기존의 동작 그래프에 변형을 가한다. 변경한 사항은 동작 그래프내의 각 꼭짓점을 프레임 단위 대신 동작 단위로 한 점이다. 여전히 모서리는 자연스럽게 이어지는 꼭짓점 간에만 존재한다.

이러한 변형 동작 그래프를 만드는 과정을 살펴보자. 먼저 M가지 종류의 주기적 동작들을 단위 동작으로 나누어 각각 1개의 꼭짓점으로 지정해준다. 같은 종류의 동작 내에서 시간 순으로 꼭짓점간의 이동 화살표를 만든다. 동작 내 마지막 꼭짓점은 다시 첫 번째 꼭짓점을 향한 이동 화살표를 가짐으로써 순환하게 만들어 준다. 그 다음 다른 종류 동작 간의 이동 경로를 생성해야한다. 같은 종류 동작 내 이동과는 다르게, 다른 종류 동작 간에는 동작이 비슷하지 않아 꼭짓점 사이를 이동시 자세 차이가 크게 날 수 있다. 이는 사용자로 하여금 캐릭터의 동작이 자연스럽다고 느끼지 못하고 동작을 끊겨보이게 한다. 이를 위해 합성 꼭짓점을 추가한다. 먼저 두 꼭짓점 내의 가장 포즈가 가까운 두 프레임을 찾는다. 식 5.1에 나오는 거리 값$distance_{ij}$이 프레임 i와 프레임 j의 유사도 정도를 나타내준다. p_{ik}, p_{jk}는 각각 프레임 i와 j의 k번째 자유도 값이다. v_{ik}, v_{jk}는 각각 자유도의 속도 값이다.

$$distance_{ij} = \sum_{k=1}^{N} ((p_{ik} - p_{jk})^2 + (v_{ik} - v_{jk})^2) \quad (5.1)$$

출발하는 꼭짓점에서 선택된 프레임을 i, 도착하는 꼭짓점에서 선택된 프레임을 j 라고 하자. 단위 동작을 L 프레임으로 정의했을 때, $L+i-j$ 값에 따라 두 가지 시나리오로 나뉜다. 왜냐하면 $L+i-j$값이 동작간의 총 이동 시간을 결정하기 때문이다. 프레임 i와 j간의 자연스러운 이동 동작은 최대 $\frac{L}{2}$프레임정도가 필요하다. 출발 동작에서 도착 동작까지 걸리는 총 프레임 수는 출발 꼭짓점에서 i프레임, 자연스러운 이동 동작 $\frac{L}{2}$ 프레임 , 그다음 도착 꼭짓점에서 $L-j$프레임을 합하여 $\frac{L}{2}+L+i-j$ 프레임이 걸린다. 따라서 $L+i-j$값이 L 보다 클 경우 동작간의 이동시간이 $2L$ 정도 걸리게 되어 전체 시스템의 실시간성이 떨어지게 된다.

$L+i-j$ 값에 따른 각각의 시나리오를 살펴보자. $L+i-j$값이 L보다 작을 경우, 프레임 i와 프레임 j를 자연스럽게 이어주는 합성 동작을 추가한다. 새로운 합성 꼭짓점은 출발 꼭짓점으로부터 이동 화살표를 받고, 도착 꼭짓점을 향한 이동 화살표를 가진다. 합성 꼭짓점 내 프레임 개수는 프레임 i와 프레임 j의 거리 값에 따라 달라진다. L이 10일 때, 부드러운 동작을 생성하기 위해 평균적으로 5 프레임 정도가 필요하다. $L+i-j$ 값이 L 보다 큰 경우, 출발하는 꼭짓점의 $1...i$ frame과 도착하는 꼭짓점의 $j...L$ 프레임을 합쳐 새로운 합성 꼭짓점을 생성해낸다. L이 10일 때, 평균적으로 10 ~ 15 프레임 정도가 필요하다. 새로운 합성 꼭짓점은 출발 꼭짓점이 받았던 모든 이동 화살표를 똑같이 받고, 도착 꼭짓점이 가지는 모든 이

동 화살표를 똑같이 가진다.

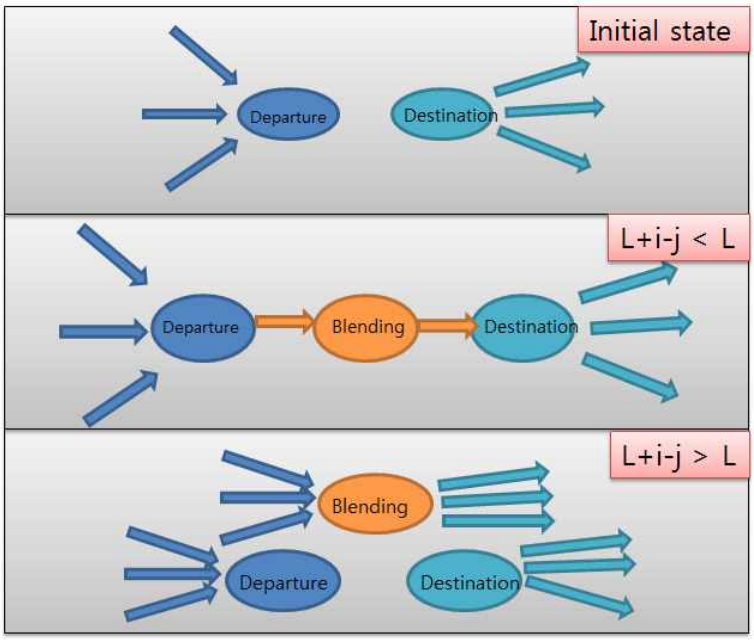

그림 5.2 합성 꼭짓점을 추가하는 두 가지 시나리오의 간단한 작동 방식

마지막으로 완성된 동작 그래프의 각 꼭짓점마다 M개의 확률 값을 저장해둔다. 확률 값이란 해당 꼭짓점에 저장된 동작이 앞서 정의한 M개의 주기적 동작 각각과 얼마나 유사한지를 말해준다. 예를 들어, 걷는 동작에서 파생된 꼭짓점이라면 걷기 확률 값은 100이고 나머지 확률 값은 0으로 저장된다. M개의 주기적 동작에서 직접 파생된 꼭짓점들에 자신의 주기적 동작에는 100, 그 외에는 0을 확률 값으로 넣어준다. 그 다음 주기적 꼭짓점들을 이어주는 합성 꼭짓점들에는 양 옆 꼭짓점들의 확률 평균값을 넣어준다. 이렇게 확

률이 저장된 캐릭터에 대한 동작 그래프를 만들어낸다.

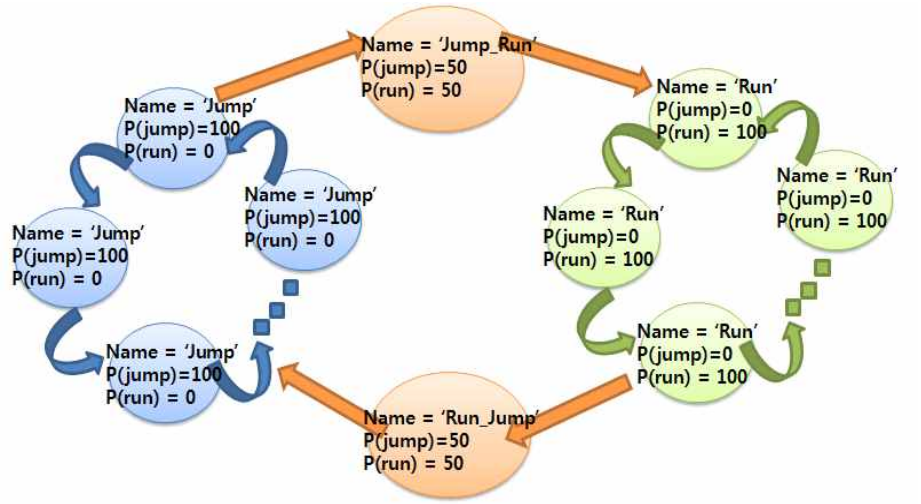

그림 5.3 간단한 동작 그래프 예시. 주황색 꼭짓점이 합성 꼭짓점이다. 각각의 꼭짓점은 5~15 프레임의 동작 정보와 동작 이름, 동작 확률을 가지고 있다.

제 6장 실시간 처리 단계

이제 실시간 처리 단계를 살펴보자. 마이크로소프트 키넥트와 같은 동작 센서 앞에서 사용자는 일련의 동작을 실시간으로 수행한다. 동작 센서를 통해 입력된 일련의 동작들을 기본 동작 단위로 끊어 앞서 정의한 동작 특징 벡터를 만들어낸다. 벡터 값을 앞서 전처리 단계에서 만들어둔 사람 동작 분류기에 넣어주면 결과 값으로 가장 유사한 주기적 동작 이름이 나온다. 만약 입력된 사용자 동작이 어떠한 주기적 동작과도 유사하지 않는 경우, 비주기적 동작 이름이 나오게 된다.

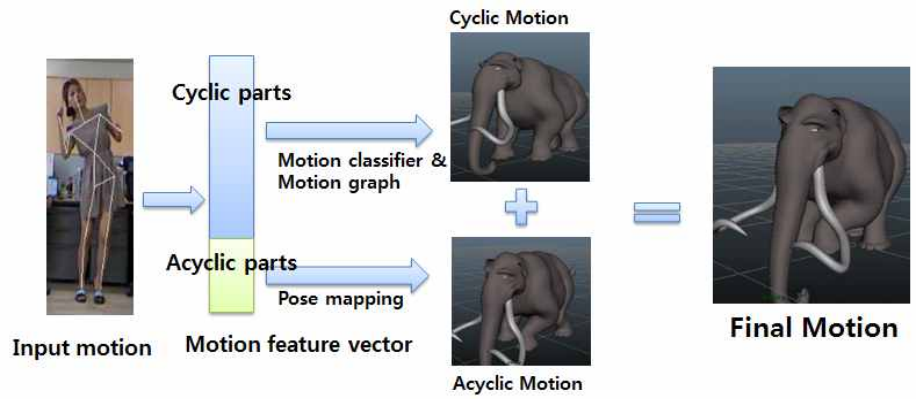

그림 6.1 자세 매핑과 동작 매핑을 통해 최종 동작이 추출된다.

결과 값으로 나온 동작 이름을 바탕으로 캐릭터의 동작 그래프 내 현재 꼭짓점에서 이동 가능한 꼭짓점들을 살핀다. 현재 꼭짓점에서 이동 가능한 꼭짓점들 중에 동작 분류기 결과 값에 해당하는 주기적 동작의 확률 값이 가장 높은 꼭짓점으로 이동한다. 입

력된 사용자 동작 정보를 바탕으로 캐릭터의 동작 그래프에서 의미가 맞는 방향으로 탐색해 나가면 원하는 주기적 동작이 결과로 나오게 된다. 여기까지 캐릭터 고유의 동작 패턴을 지키기 위한 주기적 동작에 대한 처리가 끝났다. 동작 분류기에서 비주기적 동작으로 분류된 경우 동작 그래프 내 이동 없이 다음 단계로 넘어간다.

두 번째로 주기적 동작과 관련 없는 캐릭터의 부위에 대해서 전처리 단계에서 만든 자세 매핑 함수를 이용해 비주기적 동작을 만들어낸다. 동작 그래프 탐색을 통해 얻어낸 주기적 동작을 살펴보면 특정 부위만 동작을 반복 수행함을 알 수 있다. 따라서 주기적 동작으로부터 어떠한 동작 정보도 부여받지 못한 부위들이 수행할 동작을 정해주어야 한다. 또한 동작 분류기에서 비주기적 동작으로 분류된 경우, 모든 부위에 대해서 어떠한 동작 정보도 받지 못했기 때문에 캐릭터의 모든 부위가 자세 매핑의 대상이 된다. 입력된 사람 동작의 프레임 단위 즉 자세 단위로 자세 매핑 함수에 넣어준 뒤, 계산된 캐릭터 자세를 이어주면 캐릭터의 비주기적 동작이 완성된다.

이렇게 만들어진 캐릭터의 주기적 동작과 비주기적 동작을 합쳐주면 최종 동작이 나온다. 최종 동작은 캐릭터의 고유 동작 패턴을 유지하면서 사용자가 의도한 동작을 실시간으로 수행해준다.

제 7장 결과

앞서 제안한 시스템은 사람의 동작을 이용하여 비인간형 캐릭터의 동작을 조종하는데 성공하였다. 캐릭터와 사람의 신체구조와 동작 패턴이 다름에도 이를 모두 유지하는 매핑이 이루어졌다. 그림 7.1에서는 사용자가 주기적 동작인 걷기를 수행하는 모습을 시간 순으로 보여준다. 코끼리의 4족보행은 사람의 2족보행과 동작의 싱크가 맞지 않는다. 그림 7.1에서 보이듯이, 이 알고리즘을 통해 코끼리는 사람의 보행 주기와 상관없이 자신의 보행주기를 유지한 채 걷고 있다.

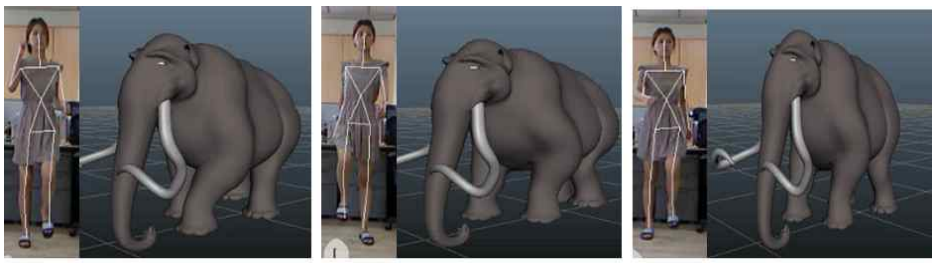

그림 7.1 (a) t=0 일 때 사용자는 오른팔과 왼발을 들고 걷는 동작을 시작한다. (b) t= 3일 때 사용자는 왼팔과 오른발을 들고 걷는 동작의 절반을 수행했다. (c) t=6일 때 사용자는 이족보행의 한 주기를 완성했다.

그림 7.2에서는 사용자가 비주기적 동작을 수행하는 모습과 그에 대응되는 코끼리의 비주기적 동작 수행 모습을 확인할 수 있다. 동작 분류기가 입력된 사용자의 동작이 비주기적임을 알아내고 입력된 동작을 바탕으로 자세 매핑을 성공적으로 수행했음을 알 수

있다.

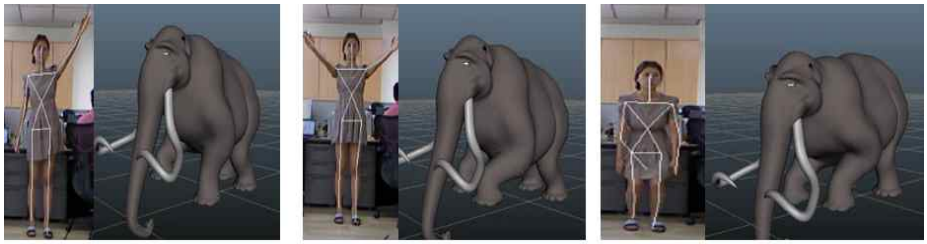

그림 7.2 (a) 팔 한쪽을 들어 올리는 비주기 동작 (b) 양 팔을 들어 올리는 비주기 동작 © 무릎을 굽혀 자세를 낮추는 비주기 동작

마지막으로 사용자가 주기적 동작과 비주기적 동작을 동시에 수행할 경우를 살펴보자. 캐릭터는 주기적 동작에 관련된 신체부위는 동작 매핑을 통해, 비주기적 동작에 관련된 신체부위는 자세 매핑을 통해 동작 데이터를 얻는다. 그림 7.3에서 캐릭터의 주기적 동작과 비주기적 동작이 자연스럽게 조화되어 수행되고 있음을 볼 수 있다. 또한 디자이너가 미리 만들어놓은 동작 외의 다양한 동작이 생성 가능함 또한 확인 할 수 있다.

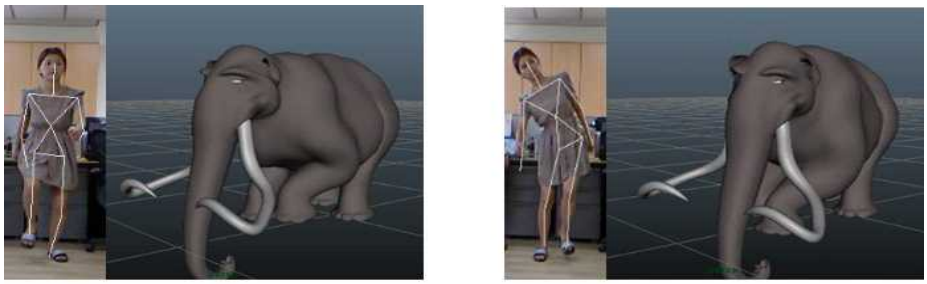

그림 7.3 (a) 주기적 동작인 걷기와 비주기적 동작인 무릎 굽히기를 동시에 수행하였다. (b) 주기적 동작인 걷기와 비주기적 동작인 몸통 흔들기를 동시에 수행하였다.

앞서 제 2 장에서 비인간형 캐릭터의 실시간 조종 문제에 관련된 최신 연구 2종류를 살펴보았다. 표 7.1에서는 해당 연구들과의 성능 비교를 보여준다. 우리의 연구결과는 캐릭터 고유의 패턴을 유지함과 동시에 다양한 동작을 생성 가능하여 이전 연구들의 한계점을 해결하였다.

	(a)	(b)	Ours
Human-Human	Y	Y	Y
Human-Character	Y	Y	Y
Real-time	Y	Y	Y
Natural Character Style	N	Y	Y
Various Motion	Y	N	Y

표 7.1 이전 연구들과 비교 (a) [Seol et al. 2013] (b) [Helge et al. 2014]

8장 논의 및 향후 연구

지금까지 신체구조와 동작 패턴이 사람과 다른 캐릭터를 사람의 동작을 이용해 실시간으로 조종하는 알고리즘을 제안했다. 이를 위해 먼저 자세 매핑 함수와 사람 동작 분류기, 캐릭터 동작 그래프를 생성하였다. 그 결과 실시간으로 동작 매핑과 자세 매핑이 함께 이루어져 캐릭터 고유의 동작 패턴을 유지하면서 다양한 동작을 생성해낼 수 있었다.

다만 전처리 단계에서 자세 매핑 함수를 생성하는데 시간이 오래 걸리기 때문에 이를 개선할 필요가 있다. 현 알고리즘에서는 여러 종류의 오차 항들을 정의한 뒤에 전체 오차 합을 최소화하는 방향으로 최적화함으로써 매핑 함수를 구했다. 이러한 방법대신 주어진 사람 동작-캐릭터 동작 데이터 쌍을 이용하여 서포트 벡터 회귀법(Support Vector Regression) 학습으로 매핑 함수를 구해보는 것이다. 개선된 방법을 통하면 시스템 생성 속도 면에서만 향상되는 것이 아니라 시스템 구현의 복잡도도 낮아질 것이라 예상한다.

이 알고리즘에서의 동작 특징 벡터는 자세 특징 벡터를 기본 동작 단위로 묶어준 것에 불과하다. 만약 동작 단위에서만이 뽑아낼 수 있는 특징 값들이 있다면, 더욱 다양한 동작 매핑이 가능할 것이다. 유용한 동작 특징 값들을 이용하여 새로운 주기적 동작까지 합성할 수 있다면, 더욱 다양한 캐릭터 조종 시스템이 될 것이다.

참고문헌

[1] Yeongho Seol, Carol O'Sullivan, and Jehee Lee, Creature Features: Online motion puppetry for non-human characters, ACM SIGGRAPH / Eurographics Symposium on Computer Animation, 2013, pages 213-222, July 2013.

[2] Rhodin Helge, Tompkin James, Kim Kwang In, Varanasi Kiran, Seidel Hans-Peter and Theobalt Christian, Interactive Motion Mapping for Real-time Character Control, Computer Graphics Forum (Proceedings Eurographics), vol. 33, no. 2, 2014.

[3] SHIRATORI, T., AND HODGINS, J. K. 2008. Accelerometer-based user interfaces for the control of a physically simulated character. ACM Trans. Graph. 27, 5, 123:1 - 123:9.

[4] LIU, H., WEI, X., CHAI, J., HA, I., AND RHEE, T. 2011. Realtime human motion control with a small number of inertial sensors. In Symposium on Interactive 3D Graphics and Games, 133 - 140.

[5] LOCKWOOD, N., AND SINGH, K. 2012. Finger walking: Motion editing with contact-based hand performance. In Proceedings of the 2012 ACM SIGGRAPH/Eurographics

symposium on Computer animation.

[6] SHOTTON, J., FITZGIBBON, A., COOK, M., SHARP, T., FINOCCHIO, M., MOORE, R., KIPMAN, A., AND BLAKE, A. 2011. Real-time human pose recognition in parts from single depth images. In Proceedings of the 2011 IEEE Conference on Computer Vision and Pattern Recognition, 1297 - 1304.

[7] WEISE, T., BOUAZIZ, S., LI, H., AND PAULY, M. 2011. Realtime performance-based facial animation. ACM Trans. Graph. 30, 4 (Aug.), 77:1 - 77:10.

[8] CHEN, J., IZADI, S., AND FITZGIBBON, A. 2012. Kinetre: animating the world with the human body. In ACM Symposium on User Interface Software and Technology.

Abstract

Online Motion Puppetry for Non-human Characters

Minji Yoon

School of Electrical and Computer Science Engineering

College of Engineering

The Graduate School

Seoul National University

As motion sensors give higher performance with lower price, various applications using 3d motion data captured from motion sensors have been introduced. One of the applications is a puppetry problem meaning that when a person produces certain actions in front of a motion sensor, non-human characters in the monitor produce actions which have the same meaning. To solve this problem, there have been many approaches. Those solutions successfully control non-human characters accomplishing the performer's intention. However they do not maintain the characters' original motion patterns. For instance, an elephant which usually moves in a quadruped manner walks with human's biped manner in those solutions. In this research, we want to

puppet non-human characters while maintaining their original motion patterns. We select features from a motion unit and map them to characters' motions, which is contrary to previous researches which learn mappings based on a pose unit. Using this algorithm, motion mappings between human and non-human characters can be done while maintaining both the semantics and original patterns of motions.

Keywords : Puppetry, Non-human character, Real-time, Animation, Motion

Student Number : 2012-23226