# 가상 캐릭터의 동작 단순화 기법
## (Motion Simplification of Virtual Character)

안 정 현 † 오 승 우 † 원 광 연 ††
(Junghyun Ahn) (Seungwoo Oh) (Kwangyun Wohn)

**요 약** 컴퓨터 그래픽스 분야에 활용되는 단순화 기법(LoD)은 실시간 렌더링을 위해 메쉬(mesh)의 다각형 수를 줄이는 방법으로 널리 알려져 있다. 본 논문에서는 이러한 단순화 기법의 기본적인 개념이 가상 캐릭터의 동작 단순화에 적용된 새로운 방법론을 제안한다. 가상 캐릭터의 구성요소 중 하나인 관절은 동작을 생성하는 기본 단위로서 메쉬와 연계되어 캐릭터의 움직임을 생성한다. 동작 단순화의 기본과정은 이러한 캐릭터의 관절 수를 줄이고 남은 관절들의 움직임을 기존동작과 가장 유사하도록 최적화 하는 방법이다. 동작의 최적화를 위해 기존 동작과 단순화 된 동작 간의 오차를 측정하는 방법론을 제시하고, 최적화 문제를 빠르게 풀기 위해 동작 간 오차 식을 선형시스템으로 재구성한다. 본 논문에 제시한 동작 단순화 방법은 동작편집 또는 군중 애니메이션의 성능향상 등 캐릭터 애니메이션의 여러 분야에 유용하게 활용 될 수 있다.

**키워드** : 캐릭터 애니메이션, 군중 애니메이션, 스키닝 애니메이션, 단순화 기법

***Abstract*** The level-of-detail (LoD), which is a method of reducing polygons on mesh, is one of the most fundamental techniques in real-time rendering. In this paper, we propose a novel level-of-detail technique applied to the virtual character's motion (Motion LoD). The movement of a virtual character can be defined as the transformation of each joint and it's relation to the mesh. The basic idea of the proposed 'Motion LoD' method is to reduce number of joints in an articulated figure and minimize the error between original and simplified motion. For the motion optimization, we propose an error estimation method and a linear system reconstructed from this error estimation for a fast optimization. The proposed motion simplification method is effectively useful for motion editing and real-time crowd animation.

**Key words** : Character animation, Crowd animation, Skinning animation, Level-of-detail

## 1. 서 론

최근 들어 영화, 게임 또는 가상현실 어플리케이션에 자주 등장하는 군중 애니메이션에 대한 관심이 높아지면서, 가상 캐릭터를 움직이기 위한 관절 변환에 많은 계산량이 필요하게 되었다. 렌더링 속도의 향상을 위해 3차원 모델의 기하학 구조인 메쉬(mesh)를 단순화[1-3]하는 연구는 다수 이루어졌지만, 캐릭터 애니메이션의 성능향상을 위한 관절구조의 동작 단순화[4,5]에 대한 연구는 많지 않았다. 본 논문에서는 군중 애니메이션의 성능향상 외에도 서로 다른 관절구조에 같은 동작을 적용할 수 있게 하는 동작편집기법(motion retargeting)[6,7]과 복잡한 관절구조를 단순화하여 실시간에도 물리기반 시뮬레이션을 가능하게 하는 시뮬레이션 단순화(simulation LoD)[8,9] 기법에도 활용할 수 있는 동작 단순화(motion LoD) 기법에 대해 기술한다.

가상 캐릭터의 동작은 캐릭터 골격구조에 속한 관절과 연계된 메쉬의 변환을 통해 계산된다. 스키닝(skinning) 애니메이션은 이러한 가상 캐릭터의 동작생성에 가장 많이 사용되는 기술[10,11] 중 하나이다. 따라서, 본 논문에서는 스키닝 애니메이션의 기본개념을 활용하여 최적화 과정을 구성하였다.

최적화 과정에서는 관절구조의 단순화, 기존동작의 자

· 본 연구는 2005년도 정보통신부 정보통신연구진흥원지원 IT 혁신연구 개발(2005-S-095), 문화관광부지원 과학기술문화체험 가상현실기술개발(1-05-4005-001-2401-00-0008), 과학기술부지원 디지털 특수영상기술개발(M10329110001-03B4311-00110)사업의 연구비 지원으로 수행하였습니다.
† 학생회원 : 한국과학기술원 전자전산학과
chocchoggi@vr.kaist.ac.kr
redmong@vr.kaist.ac.kr
†† 종신회원 : 한국과학기술원 전자전산학과 교수
wohn@vr.kaist.ac.kr
논문접수 : 2006년 4월 27일
심사완료 : 2006년 7월 12일

세와 단순화된 동작의 자세 차이를 비교할 수 있는 오차공식의 정의 그리고 최적의 해를 빠르게 풀기 위한 선형시스템 구성의 세 가지 단계로 분류된다. 첫 단계인 관절구조 단순화에서는 골격구조의 상세도에 따라 관절이 제거되는 방법과 연계된 메쉬의 정보수정에 대해 기술한다. 두 번째 단계인 오차공식 정의에서는 스키닝 애니메이션 기법에 기반한 메쉬 변환을 오차공식에 어떻게 적용하였는지를 보여준다. 마지막 단계에서는 오차공식에서 선형시스템이 구성되는 과정과 최적의 해인 최소자승근사값(least-square approximation) [12-14]을 구하는 과정에 대해 기술한다.

골격구조의 단순화에 따라 다양한 골격구조의 같은 동작을 생성할 수 있으며, 이를 실시간 애니메이션 시스템의 성능향상을 목적으로 활용할 수 있다. 실시간 애니메이션에 적용하기 위해서는 동작의 상세도를 결정 할 수 있는 방법이 필요하다. 본 논문에서는 시각적인 질을 유지하면서 단순화 된 동작을 애니메이션 할 수 있도록 상세도를 결정하는 방법론에 대해 소개한다.

본 논문의 주된 목적은 실시간 군중 애니메이션을 위한 동작 단순화 기법의 제안이며 세부적으로 다음 사항들을 포함한다.

- 골격구조를 단순화 하는 방법 제안
- 서로 다른 구조의 동작을 최적화하는 방법 제안
- GPU를 활용한 스키닝 애니메이션에 직접 활용할 수 있는 단순화 방법 제안
- 실시간 환경에서 동작의 상세도 결정할 수 있는 방법 제안

본 논문의 구성은 다음과 같다. 1장의 서론에 이어 2장에서는 관련연구에 대해 기술하고, 3장에서 스키닝 애니메이션의 개념을 살펴본다. 4장에서는 본 논문의 핵심인 동작 최적화 과정을 설명하고, 5장에서는 실시간 환경에서 동작 상세도를 결정하는 방법에 대해 기술한다. 마지막으로 6, 7장에서는 실험결과와 결론을 도출한다.

## 2. 관련 연구

캐릭터 애니메이션의 성능향상을 위한 연구분야는 크게 두 가지로 나뉜다. 첫 번째 분야는 캐릭터의 동작들을 각 카메라 방향에 대해 샘플링 하여 애니메이션 중에 관절변환 대신 이미지로 대체하는 이미지기반 기술(image-based technique)[15,16]이다. 캐릭터 관절을 매 프레임 변환하는 대신 저장된 이미지로 대체 하는 과정을 반복하기 때문에 다수의 캐릭터가 존재할 경우 애니메이션의 성능향상을 극대화 시킬 수 있다. 하지만, 모든 카메라 방향에 대해 샘플링 하는 것이 불가능하기 때문에 동작의 정확성이 떨어지고 동작 하나를 샘플링 하는데 메모리가 많이 필요할 뿐만 아니라 사용자의 요구에 맞게 동작을 제어하는 것이 불가능하기 때문에 애니메이션 속도 향상 이외에 많은 단점들이 존재한다. 이러한 문제점들을 보완하기 위해 일반적으로 활용되는 애니메이션 기술과 이미지기반 기술을 융합한 사례[17]가 최근 발표되었으나 두 가지 방법으로 생성된 동작의 시각적 차이가 크기 때문에 장면의 사실성(scene reality)이 떨어진다. 또한, 이전에 발생했던 문제들이 보완되었지만 여전히 존재한다. 이러한 단점을 극복하기 위해 이미지기반 기술을 배제하고 점차적으로 동작의 상세도를 줄여 나갈 수 있는 동작 단순화 기법의 필요성이 대두된다.

두 번째 분야는 골격구조 단순화에 초점을 맞춘다. 골격구조 단순화는 물리기반 시뮬레이션의 성능향상을 목적으로 주로 연구되었다. 물리기반 동작 생성 시스템에 빠른 동작융합(motion blending)을 위해 수작업으로 골격구조를 단순화[18]하여 성능을 향상시킨 사례가 있다. 최근에는 시뮬레이션 단순화에 활용할 수 있는 새로운 방법들[8,9]이 제안되고 복잡한 골격구조를 빠른 속도로 시뮬레이션 하였다. 본 논문은 물리적 시뮬레이션보다 모션캡처(motion capture)[19,20]된 동작을 빠른 속도로 애니메이션 할 수 있는 골격구조 단순화 방법에 대해 초점을 맞춘다. 모션캡처된 동작의 속도향상을 위한 관련연구로는 유사한 관절의 움직임을 추출하는 JPC(joint posture clustering)[5] 방법이 가장 대표적이다. JPC 방법은 캐릭터의 골격구조 내에 존재하는 관절들의 매 프레임 변환이 애니메이션 성능을 저하시키는 주요 원인 중의 하나로 보고, 이를 해결하기 위해 각 관절들의 변환 값을 군집화(clustering)하여 각 관절의 유사한 자세들을 추출하여 관절 변환을 최소화 한다. 이 경우 동작이 분석된 상황에 따라 관절들의 계층구조가 동적으로 변하기 때문에 계층구조를 재구성하는데 드는 시간 비용이 존재한다. 본 논문에서는 이러한 계층구조의 재구성이 필요 없고 최적화 과정을 통해 기존 동작과의 유사성을 잃지 않는 동작 단순화 기법을 제안한다.

## 3. 스키닝 애니메이션(Skinning Animation)

동작 단순화는 소수의 관절로 구성된 골격구조의 동작을 다수의 관절로 구성된 기존 골격구조의 동작에 최대한 유사하도록 만드는 방법이다. 가장 유사한 동작을 만들기 위해서 다양한 방법으로 접근이 가능한데, 본 논문에서는 스키닝 애니메이션에 의해 계산되는 메쉬의 점(mesh vertex)들을 비교하여 그 차이를 최소화 하는 방법을 택했다. 이는 메쉬의 점들이 시각적으로 그려지는 다각형(polygon)들과 직접적으로 관련되어 있고 캐릭터 애니메이션을 위해 스키닝 방법이 가장 널리 활용되기 때문이다.

캐릭터 동작(motion)은 자세(posture)들의 연속으로 이루어진다. 전체 동작 중 프레임 $t$의 자세는 $(\mathbf{v}^t, \mathbf{M}^t)$의 한 쌍으로 정의된다. 여기서 $\mathbf{v}^t$는 스키닝 애니메이션에 의해 프레임 $t$일 때 변환된 $p$개 점들의 집합 $\{\mathbf{v}^t_1, \mathbf{v}^t_2, \cdots, \mathbf{v}^t_p\}$이고, $\mathbf{M}^t$는 프레임 $t$일 때 $n$개 관절의 변환 행렬들의 집합 $\{\mathbf{M}^t_1, \mathbf{M}^t_2, \cdots, \mathbf{M}^t_n\}$이다. 또한, 스키닝 애니메이션을 위해 초기자세(rest pose)가 필요한데, 이는 $(\mathbf{v}^0, \mathbf{M}^0)$로 정의된다. 따라서 총 $f$개의 프레임이 존재하는 동작을 $(\mathbf{v}^0, \mathbf{M}^0)$, $(\mathbf{v}^1, \mathbf{M}^1)$, $\cdots$, $(\mathbf{v}^f, \mathbf{M}^f)$과 같은 자세들의 연속으로 정의할 수 있다.

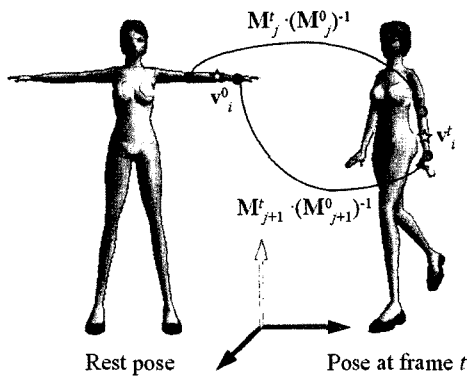



그림 1 스키닝 애니메이션 과정(초기자세) $\mathbf{v}^0_i$: $i$번째 점, $\mathbf{M}^0_j$: $j$번째 관절의 변환행렬(프레임 $t$ 자세) $\mathbf{v}^t_i$: $i$번째 점, $\mathbf{M}^t_j$: $j$번째 관절의 변환행렬

그림 1에 도시한 행렬 $\mathbf{M}^t_j \cdot (\mathbf{M}^0_j)^{-1}$은 $j$번째 관절의 스키닝 행렬이며, 하드웨어 가속을 필요로 하는 캐릭터 애니메이션에서 GPU에 직접 넘겨줄 수 있는 행렬이다. 따라서, 프레임 $t$일 때 $j$번째 관절의 스키닝 행렬을 $\mathbf{K}^t_j$로 재정의하면, 메쉬의 $i$번째 점을 변형하는 스키닝 공식은 식 (1)과 같다.

$$\mathbf{v}^t_i = \sum_j^J w_{ij} \mathbf{K}^t_j \mathbf{v}^0_i \tag{1}$$

집합 $J$는 $i$번째 점과 연관된 관절 인덱스의 집합이고, 변수 $w_{ij}$는 $i$번째 점이 $j$번째 관절과 연관된 가중치 값이다.

## 4. 동작 최적화(Motion Optimization)

앞서 기술하였듯이, 서로 다른 골격구조의 두 동작을 최대한 유사하게 만드는 것이 동작 최적화 방법의 목적이다. 최적화 과정은 차례대로 한 프레임씩 자세를 최적화하여 최종적으로 최적화된 전체 동작을 생성한다. 따라서, 본 장에서는 동작 최적화의 기본 단위인 자세 최적화(posture optimization) 방법에 대해 집중적으로 설명한다. 우선, 관절 수를 줄이는 골격구조 단순화의 의미를 시작으로 자세 간 오차공식과 선형시스템 설계에 대해 기술한 후 결과를 도시한다.

### 4.1 골격구조 단순화(Skeletal Simplification)

동작을 단순화 하기 위해 첫 단계로 동작을 의미하는 기본 단위 중 하나인 관절 수를 줄인다. 실제로 캐릭터의 관절 수는 애니메이션의 성능을 저하시키는 원인 중 하나가 된다. 여기서 주목해야 할 점은 관절을 제거해도 기존 동작의 특징들을 최대로 유지할 수 있어야 한다는 것이다. 이러한 관절제거 우선순위의 특성을 살펴보면 아래와 같다.

**계층구조상의 위치(Hierarchical factor)**: 하위계층의 관절일수록 그 관절의 움직임에 영향을 받는 관절이 적기 때문에 하위 관절의 우선 제거가 전체 동작의 특성을 유지 시킬 수 있다.

**회전변화(Rotational factor)**: 상위계층의 관절이라도 그 관절의 회전변화가 적은 경우 하위 관절들이 영향을 적게 받기 때문에 관절 제거 우선순위가 높아진다.

본 논문에서는 관절 제거의 우선순위를 결정하기 위해 위의 두 가지 중요 사항을 고려한 JPC방법의 자세 간 오차측정 방법[5]을 적용하였다. 자세 간 오차의 합이 가장 적은 관절부터 차례대로 우선순위 리스트에 포함시켰다. 걷는 동작에 대한 오차측정 결과 관절 제거의 우선순위는 그림 2와 같다. 그림에서 보듯이, 상위계층에 있는 척추관절들(spine2, spine3)이 회전변화가 적은 경우 관절 제거의 높은 우선순위로 올라갈 수 있고 상대적으로 하위계층에 있는 팔 관절들(LUpperArm, RUpperArm)이 회전변화가 많은 경우 관절 제거의 낮은 우선순위로 내려갈 수 있다.

최적화 과정에서는 제거할 관절의 개수를 결정하고 우선순위 리스트에 따라 관절을 제거한 후 남은 관절들의 모든 변환행렬을 재구성한 선형시스템으로 계산한다. 단, 집합 $J$와 가중치 $w$에 남아있는 제거된 관절의 정보는 제거 되지 않은 부모 관절로 수정된다.

### 4.2 자세(Posture) 최적화를 위한 오차 추정

자세 최적화를 선형시스템으로 재구성하기에 앞서 프레임 $t$일 때 기존 동작의 자세와 단순화된 골격구조의 자세 간의 오차를 계산할 수 있는 방법이 필요하다. 본 절에서는 이러한 자세 간의 오차를 계산하는 방법에 대해 기술한다. 프레임 $t$일 때의 자세 간 오차는 식 (2)와 같이 정의된다.

$$E(t) = \alpha E_p(t) + \beta E_n(t) \tag{2}$$

자세 간 오차인 $E(t)$는 스키닝 애니메이션으로부터 계산된 점들의 위치 값에 대한 오차(positional error) $E_p(t)$와 노말 값에 대한 오차(normal error) $E_n(t)$의 합으로 계산된다. 상수인 $\Rightarrow$와 $\Leftarrow$는 각 오차에 대한 가중치 값으로 $\Rightarrow$는 위치 오차의 최대 값의 역수이고 $\Leftarrow$는

| Priority | Joint Name | Hierarchical | Rotational | Total Error |
|---|---|---|---|---|
| 1 | RToe0Nub | 0.000 | 0.000 | 0.000 |
| 2 | HeadNub | 0.000 | 0.000 | 0.000 |
| 3 | LFinger0Nub | 0.000 | 0.000 | 0.000 |
| 4 | LFinger1Nub | 0.000 | 0.000 | 0.000 |
| 5 | RFinger0Nub | 0.000 | 0.000 | 0.000 |
| 6 | RFinger1Nub | 0.000 | 0.000 | 0.000 |
| 7 | LToe0Nub | 0.000 | 0.000 | 0.000 |
| 8 | LFinger01 | 0.022 | 0.000 | 0.000 |
| 9 | RFinger01 | 0.022 | 0.000 | 0.000 |
| 10 | LFinger11 | 0.040 | 0.000 | 0.000 |
| 11 | RFinger11 | 0.040 | 0.000 | 0.000 |
| 12 | LFinger0 | 0.044 | 0.000 | 0.000 |
| 13 | LFinger2 | 0.073 | 0.000 | 0.000 |
| 14 | RToe0 | 0.083 | 0.000 | 0.000 |
| 15 | RFinger0 | 0.044 | 0.000 | 0.000 |
| 16 | RFinger2 | 0.073 | 0.000 | 0.000 |
| 17 | LToe0 | 0.083 | 0.000 | 0.000 |
| 18 | Spine2 | 2.451 | 0.000 | 0.000 |
| 19 | Spine3 | 2.375 | 0.000 | 0.000 |
| 20 | Head | 0.230 | 0.150 | 0.034 |
| 21 | LHand | 0.272 | 0.133 | 0.036 |
| 22 | Neck | 0.338 | 0.196 | 0.066 |
| 23 | LFoot | 0.253 | 0.474 | 0.120 |
| 24 | RHand | 0.272 | 0.550 | 0.150 |
| 25 | RClavicle | 0.849 | 0.179 | 0.152 |
| 26 | RFoot | 0.253 | 0.655 | 0.166 |
| 27 | RForearm | 0.535 | 0.685 | 0.366 |
| 28 | LClavicle | 0.849 | 0.495 | 0.420 |
| 29 | LForearm | 0.535 | 0.880 | 0.471 |
| 30 | RCalf | 0.781 | 0.650 | 0.507 |
| 31 | Spine1 | 2.532 | 0.229 | 0.579 |
| 32 | LCalf | 0.788 | 0.756 | 0.595 |
| 33 | LThigh | 1.186 | 0.504 | 0.598 |
| 34 | RThigh | 1.180 | 0.595 | 0.702 |
| 35 | Spine | 5.242 | 0.144 | 0.753 |
| 36 | LUpperArm | 0.748 | 2.034 | 1.521 |
| 37 | RUpperArm | 0.748 | 2.343 | 1.751 |
| 38 | Pelvis | 5.341 | 2.824 | 15.083 |

그림 2 걷는 동작(walking motion)에 대한 관절제거 우선순위 리스트 계산 결과; 최종 우선순위는 두 특성값의 곱인 'Total Error' 값으로 결정된다.

노말 오차 최대 값의 역수이다. 위치오차와 노말오차는 아래 그림 3에 도시된 **v**, **p**, **n**, **o** 네 종류의 벡터들 차이의 합으로 정의된다.

위치 오차 $E_p(t)$는 기본적으로 프레임 $t$일 때 기존 동작 자세의 메쉬 점 위치($\mathbf{v}^t_i$)들과 단순화된 골격구조의 관절 변환 후 메쉬 점 위치($\mathbf{v}'^t_i$)들 차이의 합이다. 이는 식 (3)과 같이 표현된다.

$$E_p(t)=\sum_{i=1}^{p}A_i\left\|\mathbf{v}^t_i-\mathbf{v}'^t_i\right\|^2+\sum_{j=1}^{m}l_j\left\|\mathbf{p}^t_j-\mathbf{p}'^t_j\right\|^2 \quad (3)$$

위치 벡터 $\mathbf{p}^t_j$와 $\mathbf{p}'^t_j$는 메쉬의 점의 위치가 아닌 관절의 위치로써 메쉬 점의 위치와 같은 형태로 위치 오차를 계산한다. 상수 $A_i$는 위치 차이 가중치로 $i$번째 점

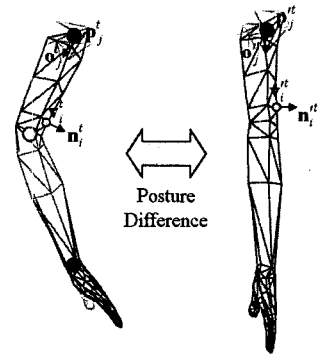



그림 3 같은 프레임 $t$에서 3개의 관절로 이루어진 기존 동작 자세와 1개의 관절로 이루어진 단순화 된 동작 자세의 차이로 **v**는 메쉬 점의 위치, **p**는 관절의 위치, **n**은 메쉬 점의 노말, **o**는 관절의 방향을 나타낸다. 또한, 단순화 된 자세의 **v'**, **p'**, **n'**, **o'** 는 기존 자세에서 정의된 **v**, **p**, **n**, **o** 에 일대일 대응되는 벡터들이다.

과 연결된 다각형 면적의 합이고, $l_j$는 관절의 위치 차이 가중치로 $j$번째 관절 밑에 있는 최하위 계층 관절(leaf node)들까지의 거리의 합이다. 상수 $p$와 $m$은 각각 메쉬 점의 개수와 단순화된 골격구조의 관절 개수를 나타낸다.

노말 오차 $E_n(t)$는 기본적으로 위치 오차와 유사하게 정의된다. 단, 다른 점은 메쉬 각 점들의 위치벡터 대신 노말벡터($\mathbf{n}^t_i$, $\mathbf{n}'^t_i$)의 차이를 오차계산의 척도로 활용한다. 이는 식 (4)와 같이 표현된다.

$$E_n(t)=\sum_{i=1}^{p}A_i\left\|\mathbf{n}^t_i-\mathbf{n}'^t_i\right\|^2+\sum_{j=1}^{m}l_j\left\|\mathbf{o}^t_j-\mathbf{o}'^t_j\right\|^2 \quad (4)$$

벡터 $\mathbf{o}^t_j$는 기존 동작이 프레임 $t$일 때 $j$번째 관절의 y축 방향 벡터이고 $\mathbf{o}'^t_j$는 단순화된 동작이 프레임 $t$일 때 $j$번째 관절의 y축 방향 벡터이다. 프레임 $t$에서의 자세 최적화는 자세 오차 $E(t)$를 최소화 하는 $\mathbf{v}'^t_i$, $\mathbf{p}'^t_j$, $\mathbf{n}'^t_i$, $\mathbf{o}'^t_j$ 벡터를 구하는 것이다.

### 4.3 선형시스템(Linear System) 설계

자세 최적화를 풀기 위해서는 관절의 개수가 다른 골격구조의 관절 변환 행렬을 구해야 한다. 따라서, 자세 오차 $E(t)$에서 미지수로 규정된 $\mathbf{v}'^t_i$, $\mathbf{p}'^t_j$, $\mathbf{n}'^t_i$, $\mathbf{o}'^t_j$ 값을 관절의 변환 행렬이 있는 식으로 변형해야 한다. 이는 앞서 3장에서 기술한 스키닝의 애니메이션의 식 (1)을 통해 식 (5)와 같이 표현된다.

$$\mathbf{v}'^t_i=\sum_{j}^{J'}w'_{ij}\mathbf{K}'^t_j\mathbf{v}'^0_i \qquad \mathbf{p}'^t_j=\mathbf{K}'^t_j\mathbf{p}'^0_j$$

$$\mathbf{n}'^t_i=\sum_{j}^{J'}w'_{ij}\mathbf{K}'^t_j\mathbf{n}'^0_i \qquad \mathbf{o}'^t_j=\mathbf{K}'^t_j\mathbf{o}'^0_j \quad (5)$$

위 식에서 단순화된 캐릭터의 초기자세(rest pose)와 연관된 벡터 $\mathbf{v'}^0_i$, $\mathbf{p'}^0_j$, $\mathbf{n'}^0_i$, $\mathbf{o'}^0_j$는 기존 골격구조의 초기자세와 차이가 없으므로 기존 골격구조의 초기자세 벡터인 $\mathbf{v}^0_i$, $\mathbf{p}^0_j$, $\mathbf{n}^0_i$, $\mathbf{o}^0_j$와 같은 값을 갖는다. 또한, 앞서 4장 1절에서 기술하였듯이, 관절이 단순화 된 후 집합 $J$와 가중치 $w$에 남아있는 제거된 관절의 정보는 모두 제거 되지 않은 부모 관절로 바뀌며, $J'$과 $w'$으로 수정된다. 따라서, 최종적으로 프레임 $t$일 때 단순화된 골격구조의 스키닝 애니메이션 행렬들의 집합인 $\mathbf{K'}^t$만 구하면 된다. 행렬들의 집합 $\mathbf{K'}^t$의 모든 원소들을 1열로 나열한 벡터를 $\mathbf{k'}^t$로 정의한다.

관절 변환행렬의 최적화된 원소 벡터 $\mathbf{k'}^t$를 구하기 위한 선형시스템은 식 (6)과 같이 설계된다.

$$\begin{bmatrix} \alpha A_1\mathbf{V}_{11} & \cdots & \alpha A_1\mathbf{V}_{1m} \\ \vdots & & \vdots \\ \alpha A_p\mathbf{V}_{p1} & \cdots & \alpha A_p\mathbf{V}_{pm} \\ \alpha l_1\mathbf{P}_1 & \cdots & \mathbf{0} \\ \vdots & & \vdots \\ \mathbf{0} & \cdots & \alpha l_m\mathbf{P}_m \\ \beta A_1\mathbf{N}_{11} & \cdots & \beta A_1\mathbf{N}_{1m} \\ \vdots & & \vdots \\ \beta A_p\mathbf{N}_{p1} & \cdots & \beta A_p\mathbf{N}_{pm} \\ \beta l_1\mathbf{O}_1 & \cdots & \mathbf{0} \\ \vdots & & \vdots \\ \mathbf{0} & \cdots & \beta l_m\mathbf{O}_m \end{bmatrix} \cdot \begin{bmatrix} \mathbf{k}'^t_1 \\ \vdots \\ \mathbf{k}'^t_m \end{bmatrix} = \begin{bmatrix} \alpha A_1\mathbf{v}^t_1 \\ \vdots \\ \alpha A_p\mathbf{v}^t_p \\ \alpha l_1\mathbf{p}^t_1 \\ \vdots \\ \alpha l_m\mathbf{p}^t_m \\ \beta A_1\mathbf{n}^t_1 \\ \vdots \\ \beta A_p\mathbf{n}^t_p \\ \beta l_1\mathbf{o}^t_1 \\ \vdots \\ \beta l_m\mathbf{o}^t_m \end{bmatrix} \tag{6}$$

선형시스템의 좌 항은 단순화된 골격구조를 통해 계산된 자세 $\mathbf{v'}^t_i$, $\mathbf{p'}^t_j$, $\mathbf{n'}^t_i$, $\mathbf{o'}^t_j$를 표현하고 우 항은 기존 동작 $t$프레임에서의 자세 $\mathbf{v}^t_i$, $\mathbf{p}^t_j$, $\mathbf{n}^t_i$, $\mathbf{o}^t_j$를 나타낸다. 이 선형시스템은 각 벡터들의 차이를 최소자승근사값을 푸는 형태(least-square form)로 표현한 것이다.

좌 항의 단순화된 골격구조의 자세 중 $\mathbf{k'}^t$를 추출하고 남은, 행렬 $\mathbf{V}_{ij}$, $\mathbf{P}_j$, $\mathbf{N}_{ij}$, $\mathbf{O}_j$는 모두 3×12 행렬로서 아래 식 (7)과 같은 1×4 행렬(벡터) 블록을 대각선 원소(block diagonal)로 가지도록 정의된다. 이 외의 다른 원소들은 전부 0이다.

$$\begin{matrix} w'_{ij}\left[(\mathbf{v}'^0_i)^{\mathrm{T}}\ 1\right] & \left[(\mathbf{p}'^0_j)^{\mathrm{T}}\ 1\right] \\ w'_{ij}\left[(\mathbf{n}'^0_i)^{\mathrm{T}}\ 0\right] & \left[(\mathbf{o}'^0_j)^{\mathrm{T}}\ 0\right] \end{matrix} \tag{7}$$

메쉬 점의 위치와 노말의 경우 $i$번째 메쉬 점과 $j$번째 관절이 서로 관계가 없는 경우 $w'_{ij}$의 값이 0이 되므로 $\mathbf{V}_{ij}$와 $\mathbf{N}_{ij}$는 모든 원소가 0인 행렬이 된다. 또한, $\mathbf{N}_{ij}$와 $\mathbf{O}_j$의 경우 기본적인 스키닝 애니메이션에서 행렬의 회전 부분(rotational part)만을 추출하여 계산하기 때문에 마지막 원소가 1 대신 0의 값을 가지게 된다. 행렬 $\mathbf{V}_{ij}$는 $w'_{ij}$ 값이 0이 아닐 경우 $\mathbf{v}_x{}^t_i$, $\mathbf{v}_y{}^t_i$ 그리고 $\mathbf{v}_z{}^t_i$의 3개 식을 생성한다. 하나의 관절행렬 $\mathbf{k'}^t_j$는 상수인 마지막 행을 제외하고 12개의 원소로 구성되므로 하나의 관절에 연관된 메쉬의 점이 4개 이상 존재해야 한다. 하지만, 관절의 위치 식인 $\mathbf{p}_x{}^t_j$, $\mathbf{p}_y{}^t_j$ 그리고 $\mathbf{p}_z{}^t_j$의 3개 식도 같은 $j$번째 관절행렬에 대한 식이므로 한 관절에 대해 메쉬의 점이 3개 이상 존재하면 된다.

### 4.4 회전행렬(Rotational Matrix) 추출

식 (6)에서 구한 단순화된 골격구조의 관절행렬들은 최적화하는 과정에서 회전(rotational), 밀림(shearing), 비율(scaling) 등의 효과를 동시에 포함시키게 된다. 하지만, 밀림 그리고 비율 효과는 메쉬 점들의 거리상 오차는 줄일 수 있어도 전체적인 메쉬 형태를 변형시키기 때문에 회전행렬만 남도록 관절행렬을 분해시키는 방법이 필요하다. 이를 위해, 본 논문에서는 극 분해(polar decomposition) 방법[21]을 적용하였다. 선형 시스템을 푼 결과행렬 $m$개 중 하나를 $\mathbf{K}$로 정의했을 때 행렬 $\mathbf{K}$의 회전효과만을 남기기 위해 아래 식 (8)을 활용한다. 행렬 $\mathbf{R}$은 구하고자 하는 회전행렬이고 행렬 $\mathbf{S}$는 밀림 그리고 비율효과를 가진 행렬이다.

$$\begin{aligned} \mathbf{K} &= \mathbf{R}\cdot\mathbf{S} \\ \mathbf{K} &= \mathbf{R}\cdot\left(\sqrt{\mathbf{K}^T\cdot\mathbf{K}}\right) \\ \mathbf{R} &= \mathbf{K}\cdot\left(\sqrt{\mathbf{K}^T\cdot\mathbf{K}}\right)^{-1} \end{aligned} \tag{8}$$

### 4.5 자세 최적화(Posture Optimization) 결과

본 장에서 기술한 자세오차를 최소화하는 자세 최적화 결과는 아래 그림 4와 같다. 그림에서 보듯이 관절의 개수를 반 이상 줄여도 실제 동작의 자세와 시각적으로 차이가 없음을 확인할 수 있다. 좌측 열의 VP방법은 위치오차만을 적용한 결과로서 단순화 율이 높아질수록 캐릭터의 기본형태가 변형됨을 확인할 수 있다. 중간 열의 VPNO방법은 위치오차와 노말오차를 동시에 적용한 결과로서 단순화 율이 높을 때 캐릭터의 기본 형태를 VP방법보다 좋게 유지할 수 있다. 마지막으로 우측 열의 VPNO+PD방법은 극 분해를 최적화 결과에 적용하여 기본형태를 VPNO방법보다 좋게 유지할 수 있다. 이에 대한 자세한 실험 결과는 6장에서 상세히 다루도록 한다.

## 5. 동작 상세도 결정

앞서 4장에서 제시한 관절제거 우선순위 리스트와 자세최적화 방법으로 하나의 동작에 대해 다양한 레벨의 동작들을 생성할 수 있다. 실제 가상환경에서 적절한 레벨의 동작이 선택되기 위해서는 렌더링시 영상에 맺히는 캐릭터의 크기를 계산해야 한다. 이 외에도 캐릭터의 움직임 속도나 카메라와 캐릭터 간의 거리 그리고 사용자의 눈이 화면상에 바라보고 있는 위치 등 여러 가지 방법들이 존재한다. 하지만, 실시간에 레벨을 결정하는

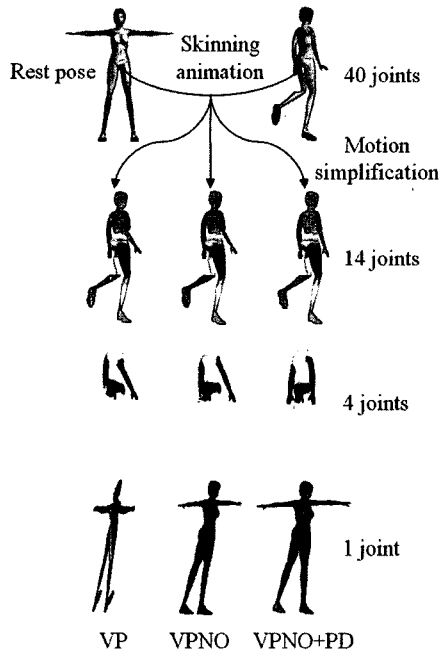
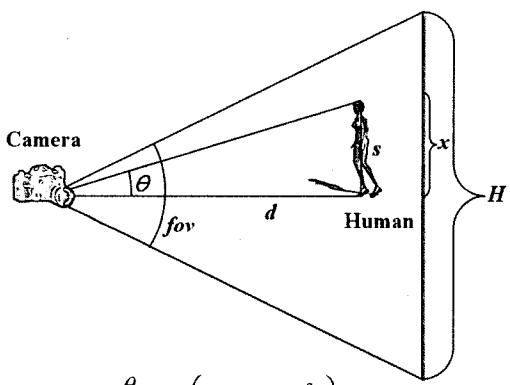



그림 4 총 40개 관절로 구성된 캐릭터의 각 관절 수에 대한 자세 최적화 결과로 VP는 위치오차만을 적용한 결과, VPNO는 위치오차와 노말오차를 동시에 적용한 결과 그리고 VPNO+PD는 위치, 노말오차를 이용한 최적화 후에 극 분해 방법을 수행한 결과이다.




$$x = \frac{\theta}{fov} H \quad \left(\theta = \arctan(\frac{s}{d})\right)$$

***fov***: Field of view
***d***: Distance from camera to human
***s***: Size of human (3D distance)
***H***: Screen height (pixel distance)
***x***: Size of human (pixel distance)

그림 5 실시간 애니메이션 환경에서 동작의 상세도를 결정하는 방법

계산량을 최소화하기 위해, 본 논문에서는 레벨 결정의 척도로서 가장 중요한, 영상에 맺히는 캐릭터의 크기만을 활용하였다. 캐릭터의 크기를 계산하는 방법은 그림 5와 같이 정의된다.

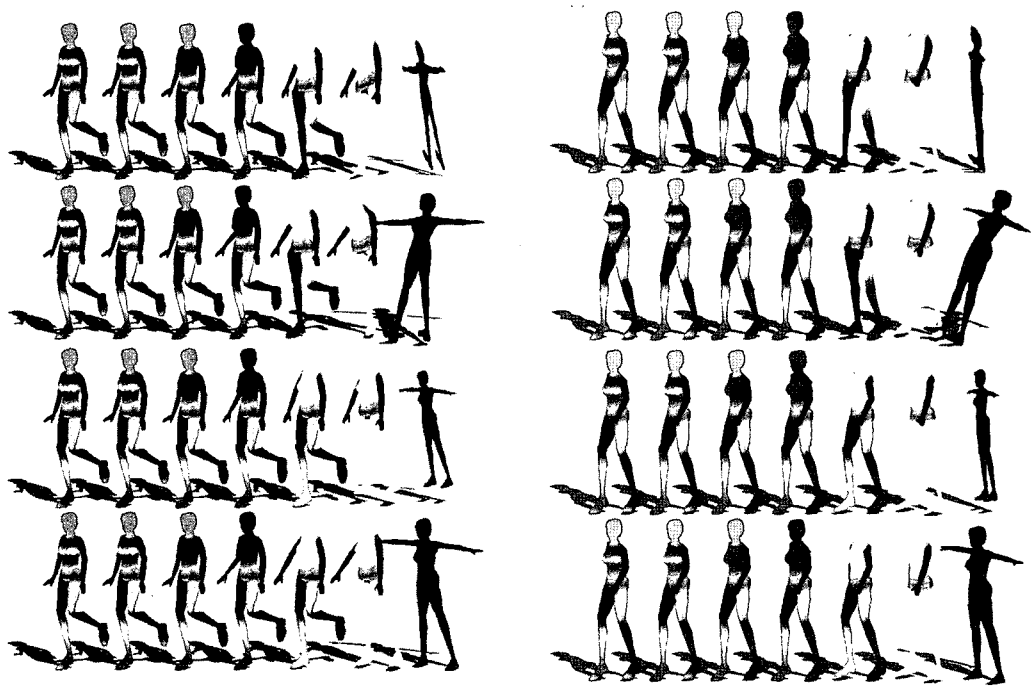
그림에서 보듯이 렌더링 영상에 맺히는 캐릭터의 픽셀 크기 $x$값을 토대로 동작의 상세도 레벨을 결정하며, $x$를 구하기 위해 실시간에 계산되는 값은 카메라와 캐릭터 간의 거리 $d$ 뿐이다. 나머지 변수들은 이미 렌더링 초기화 단계 또는 그 이전에 결정되는 상수들이다.

## 6. 실험결과

동작 단순화 기법으로 생성된 다양한 레벨의 동작은 그림 6과 같다. 걷는 동작의 두 가지 자세에 대해 VP, VP+PD, VPNO, VPNO+PD의 네 가지 자세 최적화 방법을 적용하였다. VP는 위치오차만을 적용한 최적화, PD는 극 분해 방법을 사용했는지 여부 그리고 VPNO는 위치, 노말오차를 동시에 사용한 최적화 방법이다. 그림에서 보듯이 단순화된 골격구조에 관절이 적게 남아 있을수록 기존 자세와 많은 차이를 보이게 된다. 또한, 어떤 최적화 방법을 활용해도 관절이 50% 이상 단순화 되도록 기존동작의 자세와 비슷한 자세를 유지하는 것을 확인할 수 있다. 마지막으로 극 분해 방법을 각 최적화된 결과에 수행하였을 경우 관절행렬의 밀림, 비율효과를 제거하여 전체적인 메쉬의 형태를 보존할 수 있다.

동작 단순화 기법의 효용성을 살펴보기 위해 크게 두 가지 실험을 수행하였다. 첫째로, 기존동작과 단순화된 동작 간의 자세오차가 각 동작 레벨에 따라 어느 정도 발생하는지에 대한 단순화의 정확성 측면을 실험하였고, 둘째로 동작 상세도 결정법을 적용한 실시간 환경에 성능향상이 어느 정도인지에 대한 애니메이션 신속성의 측면을 살펴봤다. 먼저 동작단순화의 정확성을 판단하기 위한 자세오차 그래프는 그림 7과 같이 도시된다.

실험결과에서 보듯이 관절이 상당 수 제거되어도 자세 오차의 평균은 거의 0에 가깝다. 이는 관절의 수가 10개 이상 남아 있을 경우 어떤 단순화 방법을 적용해도 기존동작의 특징을 잃지 않은 단순화 된 동작의 생성이 가능함을 의미한다. 관절의 단순화 율이 높은 경우 자세 오차의 차이가 발생하는데, 위치오차와 노말오차를 모두 적용한 VPNO 방법이 위치오차만을 적용한 VP방법보다 평균오차가 적게 나타남을 알 수 있다. 점선으로 표시된 각 단순화 방법에 극 분해를 적용한 경우 시각적으로는 메쉬의 기존 형태를 유지하기 때문에 보다 사실적으로 보이지만, 자세오차의 측면에서 평균오차가 증가한다.



(a) 자세 1: 위에서부터 VP, VP+PD, VPNO, VPNO+PD (b) 자세 2: 위에서부터 VP, VP+PD, VPNO, VPNO+PD

그림 6 동작 단순화 기법으로 구성한 다양한 레벨의 동작

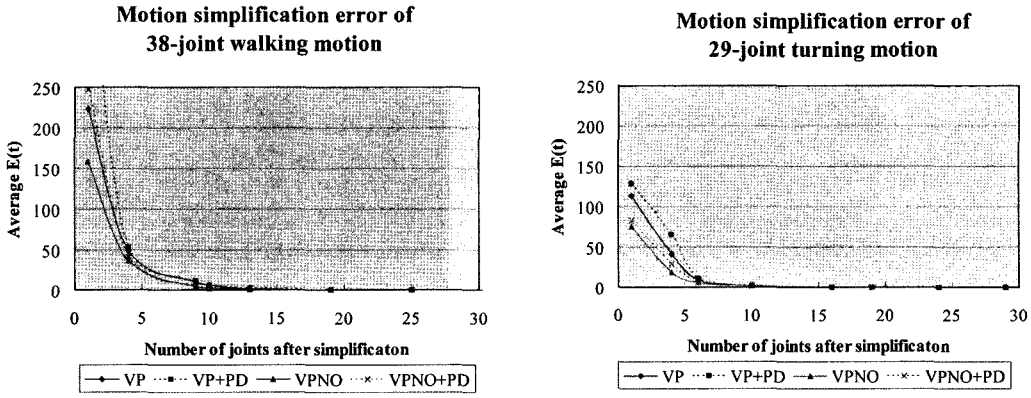



그림 7 동작 단순화 레벨 별 평균 자세오차 그래프로 평균 $E(t)$는 동작 전체 프레임에 대해 식 (2)의 평균 결과이며 38개의 관절로 구성된 걷기동작과 29개의 관절로 구성된 돌기동작에 대해 각각 실험한 결과이다($\alpha$=1.43, $\beta$=0.53).

동작단순화의 속도향상 측면은 실시간 애니메이션의 속도를 비교함으로써 실험하였다. 이를 위해 그림 8과 같이 1000개의 가상 캐릭터를 동작단순화 기법을 적용하였을 때와 하지 않았을 때를 구분하여 애니메이션 속도를 비교하였다. 실험결과, 동작단순화를 적용할 경우 3~4배 정도의 애니메이션 성능향상이 있음을 확인할 수 있다. 이 실험의 경우 동작단순화의 성능향상을 확인하기 위해 캐릭터 메쉬의 단순화는 배제하였다. 동작 단순화에 메쉬 단순화까지 적용할 경우 보다 향상된 애니메이션 속도를 기대할 수 있다. 마지막으로 성능향상 실험에 사용한 애니메이션 장면은 그림 9와 같다. 장면의 사실성을 위해 다양한 의상을 착용한 캐릭터[22]들을 애니메이션 환경에 적용하였다. 각 캐릭터당 29~38개의 관절들 그리고 2000~2200개의 다각형으로 구성되었다. 실험에 사용한 하드웨어는 P4 3.0GHz, 2GB RAM 그리고 GeForce 7800 GTX이다.

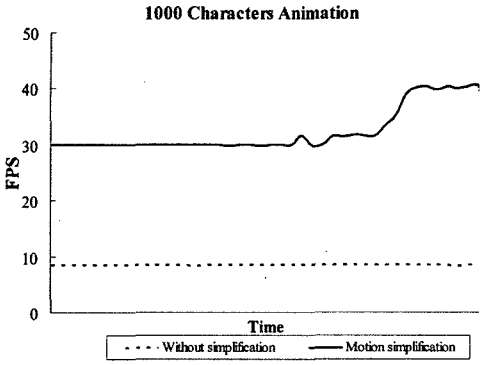


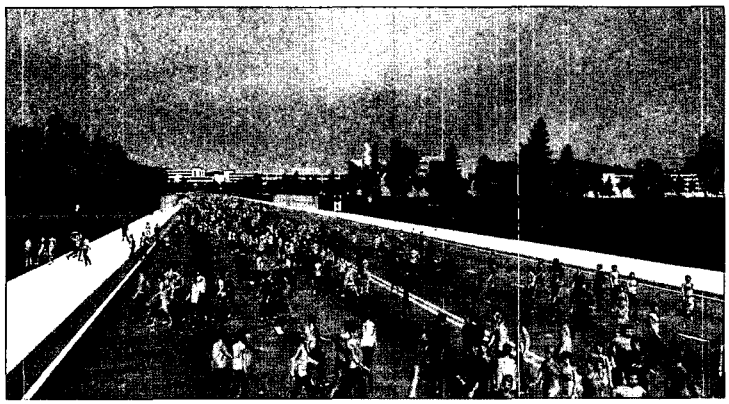

그림 8 총 1000개 캐릭터의 애니메이션 속도실험으로 점선은 동작단순화를 활용하지 않은 애니메이션 결과, 실선은 동작단순화를 적용한 애니메이션 속도 결과이다.

## 7. 결론 및 향후 연구

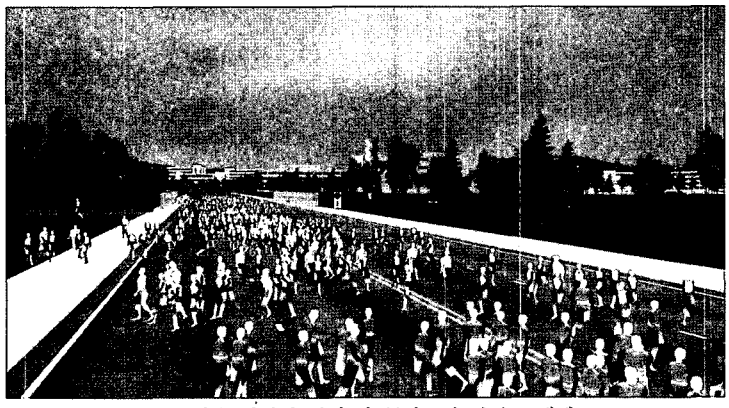
본 논문에서는 관절구조로 이루어진 가상캐릭터의 동작 단순화 방법을 소개하였다. 동작 단순화를 위해 우선 관절의 개수를 줄이고, 남은 관절들의 움직임을 기존 동작과 최대한 유사하게 유지하기 위한 자세오차를 정의하였다. 또한, 자세오차를 최소화 하기 위해 선형 시스템을 설계하여 최적의 관절 움직임을 생성하였다. 실험결과 제시된 동작 단순화 기법은 기존 동작과의 오차를 최소화 하였으며, 실시간 군중 애니메이션의 성능을 성공적으로 향상 시켰다.

본 논문에 제시한 동작 단순화 기법을 보다 향상시키기 위해 다음과 같은 향후 연구가 필요하다.

**동작 분석(Motion Analysis)**: 앞서 6장 실험결과에서 보여진 대로 동작단순화 결과는 관절제거 후 남은 관절의 개수가 적을수록 캐릭터 메쉬의 초기자세에 종속적인 형태가 됨을 확인 할 수 있다. 자세오차를 더욱 줄이기 위해서는 최적화를 수행하기 전에 동작을 분석하여 전체 동작을 가장 잘 표현하는 대표자세를 초기자세와 대체할 경우 보다 좋은 단순화 결과를 생성할 수 있다.



(a) 단순화된 동작을 활용한 실시간 애니메이션



(b) 같은 장면에 대해 각 동작들의 상세도 레벨

그림 9 총 1000개의 가상 캐릭터를 애니메이션 한 장면으로서 투영된 캐릭터의 크기가 작을수록 관절의 개수가 적은 동작을 활용하고 투영된 캐릭터의 크기가 클수록 관절의 개수가 많은 동작을 활용한다.

**동작의 연속성(Time Coherence)**: 현재 동작최적화를 위해 동작의 기본 단위인 자세들로 분리하고 각 자세 별로 자세최적화를 수행한다. 자세 간 독립적으로 최적화 과정을 거치게 되므로 동작의 연속성이 고려되지 않는다. 보다 자연스러운 동작을 생성하기 위해서는 이러한 동작 시간의 연속성을 최적화에 고려해야 한다.

**메쉬 단순화(Mesh Simplification)**: 선형 시스템 설계 과정에서 캐릭터 메쉬의 점 개수는 전체 최적화 문제를 푸는 시간과 직결된다. 기존에 발표된 메쉬 단순화 기법을 동작 단순화 기법에 함께 적용할 경우 최적화 문제를 푸는 시간을 줄일 뿐만 아니라 전체 애니메이션의 속도 향상도 기대할 수 있다.

## 참 고 문 헌


[1] H. Hoppe, "Progressive Mesh," ACM SIGGRAPH 96, pp. 99-108, 1996.

[2] M. Garland and P. Heckbert, "Mesh Simplification with Quadric Error Metrics," ACM SIGGRAPH 97, pp. 209-216, 1997.

[3] H. Kim and K. Wohn, "Multiresolution Model Generation with Geometry and Texture," VSMM 2001.

[4] D. Carlson and J. Hodgins, "Simulation Levels of Detail for Real-time Animation," Graphics Interface 97, 1997.

[5] J. Ahn and K. Wohn, "Motion Level-of-Detail: A Simplification Method on Crowd Scene," CASA 2004.

[6] M. Gleicher, "Retargetting Motion to New Characters," ACM SIGGRAPH 98, pp.33-42, 1998.

[7] K. Choi and H. Ko, "Online Motion Retargetting," Journal of Visualization and Computer Animation 11(5): pp. 223-235, 2000.

[8] J. Beaudoin and J. Keyser, "Simulation Levels of Detail for Plant Motions," SCA 2004.

[9] S. Redon, N. Galoppo and M. Lin, "Adaptive Dynamics of Articulated Bodies," ACM SIGGRAPH 2005.

[10] J. Lewis, M. Cordner and N. Fong, "Pose Space Deformations: A Unified Approach to Shape Interpolation and Skeleton-Driven Deformation," ACM SIGGRAPH 2000.

[11] B. Allen, B. Curless and Z. Popovic, "Articulated Body Deformation From Range Scan Data," ACM Transaction on Graphics 21(3): pp.612-619, 2002.

[12] X. Wang and C. Phillips, "Multi-weight Enveloping: Least squares Approximation Techniques for Skin Animation," ACM S1GGRAPH Symposium on Computer Animation, pp.129-138, 2002.

[13] D. James and C. Twigg, "Skinning Mesh Animations," ACM SIGGRAPH 2005.

[14] M. Muller, B. Heidelberger, M. Teschner and M. Gross, "Meshless Deformations Based on Shape Matching," ACM SIGGRAPH 2005.

[15] A. Aubel, R. Boulic and D. Thalmann, "Real-time display of virtual humans: Level of details and impostors," IEEE Transactions on Circuits and Systems for Video Technology, Special Issue on 3D Video Technology, 10(2): 207-217, 2000.

[16] F. Tecchia, C. Loscos and Y. Chrysanthou, "Image Based Crowd Rendering," IEEE CG&A March/April, pp.36-43, 2002.

[17] S. Dobbyn, J. Hamill, K. O'Conor and C. O'Sullivan, "Geopostors: A real-time geometry / impostor crowd rendering system," I3D 2005.

[18] Z. Popovic and A. Witkin, "Physically based motion transformation," ACM SIGGRAPH 99, pp.11-20, 1999.

[19] J. Ahn, C. Sul, E. Park and K. Wohn, "Development of LAN-based Optical Motion Capture System," HCI 2000.

[20] J. Ahn, M. Jang and K. Wohn, "Trajectory Rectification of Marker using Confidence Model," KCGS Journal, 8(3):17-23, 2002.

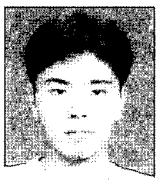
[21] K. Shoemake and T. Duff, "Matrix Animation and Polar Decomposition," Proceedings of Graphics Interface, pp.245-254, 1992.

[22] S. Oh, H. Kim, N. Magnenat-Thalmann and K. Wohn, "Generating unified model for dressed virtual humans," The Visual Computer 21(8): 522-531. 2005.



안 정 현

1998년 한국과학기술원 전산학과(학사) 2000년 한국과학기술원 전산학과(석사) 2006년 현재 한국과학기술원 전산학과 박사과정. 관심분야는 군중애니메이션, 모션캡처 시스템, 가상현실


오 승 우

2000년 한국과학기술원 전산학과(학사) 2002년 한국과학기술원 전산학과(석사) 2006년 현재 한국과학기술원 전산학과 박사과정. 관심분야는 실시간 의상애니메이션, 의상 디자인, 가상현실



원 광 연

1974년 서울대학교(학사). 1984년 Univ. of Maryland 전산학(박사). 1986년 Univ. of Pennsylvania 교수. 1991년 한국과학기술원 전산학과 교수. 2005년~현재 한국과학기술원 문화기술대학원 원장. 관심분야는 문화기술, 가상현실, 컴퓨터 비젼, 컴퓨터 그래픽스