모션 프레임의 효율적인 단순화를 위한 점진적 기법

고인영[1O] 김일수[1] 하종성[2] 유관희[1]
[1]충북대학교 정보산업공학과 및 컴퓨터교육과
[2]우석대학교 게임콘텐츠학과
khyoo@chungbuk.ac.kr, jsha@woosuk.ac.kr

A Progressive Method for the Efficient Simplification of Motion Frames

In-Young Ko[1O] Il-Soo Kim[1] Jong-Sung Ha[2] Kwan-Hee Yoo[1]
[1]Dept. of IIE and Dept of Computer Education, Chungbuk National University
[2]Dept. of Game and Contents, Woosuk University

요 약

본 논문에서는 연속된 모션 프레임의 효과적인 단순화를 매우 효율적으로 수행시킬 수 있는 점진적 모션 (progressive motion) 기법을 제안하고 실험 결과를 설명한다. 제안된 점진적 모션 기법에서는 효과적인 단순화를 위하여 인접한 두 모션간의 차이를 측정하는 수식을 정의하고 모션 LOD (level of detail)를 효율적으로 처리하기 위하여 모션 차이가 최소인 두 모션을 차례대로 병합하고 역으로 병합된 모션을 분해하는 기법을 사용하였다. 이 방법은 모션 편집 등에서 매우 빠른 속도로 모션 LOD를 제공하며 임의의 캐릭터의 실시간 시뮬레이션에서 효과적인 단순화 기법으로 적용될 수 있는 잠재력을 가지고 있다.

1. 서론

최근 들어 3차원 게임 및 애니메이션에서는 캐릭터의 모션을 시뮬레이션하기 위하여 모션 캡쳐 장비 또는 모션 생성 소프트웨어로 만들어진 모션데이터가 다양하게 사용되고 있다. 그러나 캐릭터의 자연스러운 움직임을 위하여 모션데이터를 세밀하게 생성할수록 그래픽스 렌더링 시간, 데이터의 전송 대역폭 및 저장 공간 등 시스템 자원에 과부하가 걸리므로 실시간의 모션 시뮬레이션이 불가능하게 될 수도 있다. 예를 들어 초당 30 프레임을 지원해야 하는 30 분짜리 캐릭터 모션을 만들기 위해서는 30분*60초*30프레임 = 54,000 모션이 필요하므로 각 프레임마다 많은 캐릭터의 각 모션으로 렌더링하는데 한계가 있다. 따라서 카메라 시점에서 먼 캐릭터는 모션을 단순화시키는 방법을 사용하게 되는데 단순하게 균일한 시간 단위로 일정한 개수의 연속된 프레임들을 하나의 모션으로 단순화시키는 방법은 원래의 모션에 비하여 부드럽지 않다. 따라서 수백 수천 명으로 구성된 군중 모션의 실시간 시뮬레이션[1]에서는 모션의 자연스러움을 가급적 잃지 않고 모션데이타를 단순화시키는 모션 LOD 기능이 필수적이라 할 수 있다.

본 논문에서는 궁극적으로는 캐릭터의 실시간 시뮬레이션에서 모션의 효과적인 LOD 기능을 매우 효율적인 시간에 제공하기 위한 목적을 가지는 점진적 모션 기법을 제안한다. 여기서 효과적이란 단순화 과정에서 손실이 적어 부드러운 모션을 연출하고 효율적이란 렌더링 과정에서 빠르게 연속적(continuous) LOD를 계산할 수 있음을 의미한다. 이 기법은 3차원 모델을 나타내는 메쉬의 단순화를 위한 점진적 메쉬 기법[2]을 응용한 것으로 모션 캡쳐 혹은 모션 생성 소프트웨어에 의해 만들어진 캐릭터 모션 집합이 있다고 가정할 때 모션병합(motion collapse)과 모션분해(motion split)라는 두 연산을 통해 모션 LOD를 제공한다. 즉 캐릭터의 모션들을 병합하여 새로운 모션을 만들면서 전체 모션 집합을 줄여 나가고 역으로 병합된 모션을 분해하여 보다 세밀한 모션의 집합으로 표현할 수 있는 것이다. 본 논문에서는 모션의 병합시 보다 자연스러운 모션을 생성하는 효과적

인 모션 모핑(morphing) 방법을 제시하며 모션병합과 모션분해를 모션 편집과 실시간 애니메이션에 적용하는 방법을 논의한다.

2. 관련 연구 비교

캐릭터 모션데이타를 단순화시키는 모션 LOD에 관한 연구는 다른 그래픽스 연구들에 비해 그리 많지 않은 편이다. 캐릭터 모션을 실시간으로 시뮬레이션하는 과정에서 카메라 시점에 따라 어떻게 단순화할 것인지에 대한 연구로는 Ercegovac 등[3], Ahn 등[4] 및 Redon 등[5]이 있다. 이 방법들은 캐릭터를 구성하는 뼈대(skeleton)를 단순화하고 그 모션을 새로 얻음으로써 결과적으로 모션 데이터를 줄이는 기법을 사용하고 있다. [3]에서는 캐릭터의 관절 운동을 기하적연산 또는 역운동학(inverse kinematics)으로 재설정(retargeting)하여 뼈대를 단순화시킨다. [4]는 연속적 프레임에서 유사한 자세(posture)를 가지는 관절을 클러스터링하여 뼈대를 단순화시키는 방법이고 [5]는 관절을 전진 동역학(forward dynamics)로 표현하여 원하는 자유도만큼 뼈대를 간소화시키는 방법으로 비교적 간단 방법인 [3]보다는 원래 모션과의 유사성을 더 높일 수 있는 방법들이다.

한편 다양한 모션 편집에서 사용될 수 있는 Lee 등[6]의 다해상도(multiresolution) 기법이 있다. 이 기법은 모션의 프레임수를 줄이기 위하여 연속된 모션의 패턴을 광역적으로 필터링하여 줄어든 프레임의 모션을 다시 설정하는 기법으로 모션의 축약(reduction), 확장(expansion), 생성(construction), 변형(modification), 혼합(blending), 찢어붙이기(stitching)와 같은 모션의 다양한 편집 작업을 가능하게 한다.

[3-5]은 실시간 애니메이션에 주요 목적이 있으며 하나의 프레임안에서 뼈대를 단순화시켜 축약된 동작을 얻는 공간적 (spatial) LOD 기법이라고 할 수 있고, [6]은 오프라인의 편집작업에서 프레임수를 줄이되 줄어든 모든 프레임에서 캐릭터의 모션의 부드러운 모핑에 주요 목적이 있는 시간적(temporal) LOD 기법이라 할 수 있다. 즉 두 가지 방식은 서로 주요 목적이 다르므로 [3-5]은 실시간 애니메이션에는 적합하지만 [6]에 비하여 단순화된 모션이 부자연스러운 반면, [6]은 모션 프레임 간의 변화를 광역적으로 필터링하기 때문에 모션의 부드러운 모핑이라는 측면에서 매우 좋은 결과를 얻을 수 있지만 많은 캐릭터의 실시간 애니메이션에서 연속적 모션 LOD를 제공하기에는 한계가 있다고 볼 수 있다.

본 논문에서 제시하는 점진적 모션은 [6]과 같이 연속된 모션에서 동작을 모핑하여 동작이 변하는 프레임 수를 줄이는 기법에 속한다. 따라서 이 방법은 다양한 모션 편집에서 모션 LOD를 제공하는데 응용될 수 있는데 모션의 부드러움이라는 효과적인 면에서 광역적 필터링을 사용하는 [6]보다는 뒤지지만 국부적으로 연속된 두 프레임의 모션을 병합하여 모핑하므로 시간의 효율성 면에서는 매우 빠른 연산으로 모션 LOD를 구현할 수 있다는 장점이 있다. 특히 이 방법은 지금까지 뼈대를 단순화하여 그 뼈대의 모션을 새로 계산하는 공간적 모션 LOD 기법으로만 접근되고 있는 실시간 애니메이션을 시간적 모션 LOD 기법으로도 접근할 수 있게 하여 공간적 모션 LOD와 함께 보다 효과적이면서 효율적인 모션 LOD를 제공할 수 있는 잠재력이 있다.

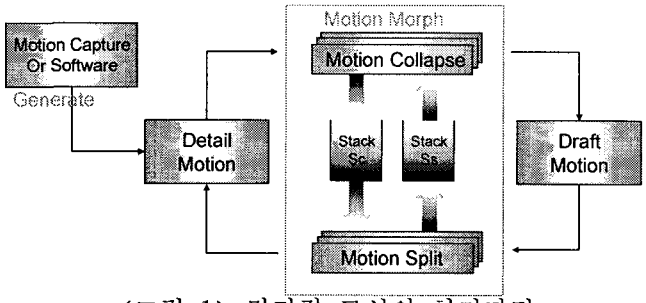
3. 제안된 점진적 모션

본 논문에서 제안하는 점진적 모션의 기본 개념은 3차원 모델을 나타내는 메쉬의 간략화에 사용되는 Hoppe 등[2]의 점진적 메쉬(progressive mesh)에 기반을 두고 있다. 점진적 메쉬는 메쉬의 구성 요소인 에지(edge)의 병합, 교환 및 분해를 통하여 메쉬를 단순화시키는 방법이다. 주어진 메쉬의 에지 병합을 위해 에지를 구성하는 두 정점간의 거리와 메쉬에서의 에지의 역할 등을 고려한 에너지 함수를 정의한다. 주어진 메쉬를 구성하는 모든 에지에 대해 에너지값을 구한 후 최소값을 갖는 에지부터 차례로 병합을 수행하면서 메쉬를 단순화하였으며 단순화된 메쉬를 복원하기 위해 병합된 에지를 역순으로 분해하였다. 본 논문에서는 메쉬 단순화 기법의 주요 개념을 연속된 캐릭터 모션의 단순화에 적용한다.

3.1. 점진적 모션의 개요

본 논문에서 제안된 점진적 모션은 캡쳐된 모션데이터 혹은 소프트웨어에 의해 만들어진 모션데이터를 입력값으로 받는다. 주어진 모션 데이터의 집합을 세부 모션

(detail motion)으로 정의하였을 때 더 이상 주어진 모션만큼 세부적인 모션이 요구되지 않을 경우 모션병합을 수행하여 주어진 모션들보다 단순화된 모션(draft motion)으로 만들고, 세부적인 모션이 요구되면 모션분해를 거쳐 원래 모션으로 복원한다. 이렇게 모션의 점진적 단순화와 세밀화를 처리하는 전반적인 과정을 <그림 1>에 기술하였다.



<그림 1> 점진적 모션의 처리과정

3.2. 모션병합 및 분해

n 개의 프레임으로 구성된 캐릭터의 모션을 고려해 보자. 설명을 쉽게 하기 위해 i 번째 프레임에서 관절의 계층으로 표현된 캐릭터 모션 상태를 X_i라 표기한다. X_i는 캐릭터 루트 위치 벡터 P_{i0}와 가중치 루트 회전값 Q_{i0}, 그리고 루트로부터 계층적으로 이루어진 모든 몸체 k에 대해 그의 부모에 관한 가중치 회전값 Q_{ik}으로 표기하며 식으로 표현하면 식 (1)과 같다.

$$X_i = [P_{io}, Q_{io}, Q_{i1} \cdots Q_{i,m}] \quad \text{(식 1)}$$

여기서 m은 캐릭터를 구성하는 뼈대 개수

따라서 n 개의 프레임으로 구성된 캐릭터의 모든 모션 M^n은 식 (2)과 같이 표현된다.

$$M^n = [X_0, X_1 \cdots X_n]^T \quad \text{(식 2)}$$

M^n을 캐릭터의 초기 모션데이터라 할 때 이 상태에서 임의의 연속적인 두 개의 모션을 하나의 모션으로 병합하는 모션병합에 만들어진 새로운 모션데이타를 $M^{n-1} = [X_0', X_1' \cdots X_{n-1}']^T$으로 표기한다. 모션병합 $mcol_i$은 단순화 방법에 의하여 선택된 모션 X_k와 X_{k+1}을 병합하여 새로운 모션 X_k'를 생성한다. 초기 모션 상태 M^n로부터 n 번의 연속적인 모션병합 과정을 적용함으로써 가장 단순한 모션 상태 M^0이 생성된다.

$$M^n \xrightarrow{mcol_{n-1}} M^{n-2} \xrightarrow{mcol_{n-2}} \cdots \xrightarrow{mcol_1} M^1 \xrightarrow{mcol_0} M^0$$

위와 같은 모션병합 과정에서 가장 중요한 요소는 어떤 기준으로 병합될 두 모션을 선택하는가이다. 본 논문에서는 특정 모션 상태 X_i에서 다른 모션 상태 X_j로 이동하기 위해 소요되는 비용함수 $D(X_i, X_j)$를 식 (3)과 같이 정의하여 사용하였다.

$$D(X_i, X_j) = w_t \times T(X_i, X_j) + w_r \times R(X_i, X_j) \text{(식 3)}$$

여기서 w_t와 w_r은 가중치

$$T(X_i, X_j) = \| P_{i0} - P_{j0} \|$$

$$R(X_i, X_j) =$$

$$\frac{1}{m}\sum_{k=0}^{m} MIN(\| log(Q_{ik}^{-1} Q_{ik}) \| , \| log(Q_{ik}^{-1}(-Q_{ik})) \|)$$

분명 두 모션의 병합을 위해서는 식 (3)의 $D(X_i, X_j)$ 값이 최소화되는 i 프레임과 j 프레임을 선택하여야 한다. 그러나 i 프레임 측면에서 모션의 특성을 분석해보면 $D(X_i, X_j)$을 최소로 하는 j 프레임은 $(i+1)$ 프레임임을 알 수 있다. 그러므로 두 모션의 병합을 위해서는 먼저 식 (4)와 같이 최소의 비용값를 가지는 i 번째 프레임을 선택하여야 한다.

$$MIN_{i=0}^{n-1} D(X_i, X_{i+1}) \quad \text{(식 4)}$$

식 (4)로부터 선택된 i 번째 프레임은 $(i+1)$ 번째 프레임과 병합이 이루어져 그 결과가 i번째 프레임에 저장

된다.

모션분해 명령 $msplit_i$은 두 모션병합의 역 과정이므로 모션 병합에서와 같이 특정한 한 프레임을 연속된 두 프레임으로 분해한다. 따라서 모션병합의 최종 상태인 M^0로부터 n 번의 모션분해 과정을 거치면 모션의 초기 상태인 M^n가 얻어진다. 그 과정을 나타내면 다음과 같다.

$$M^0 \xrightarrow{msplit_1} M^1 \xrightarrow{msplit_0} \cdots \xrightarrow{msplit_{n-2}} M^{n-1} \xrightarrow{msplit_{n-1}} M^n$$

모션병합과 모션분해가 서로 역 과정이므로 이를 효율적으로 처리하기 위해 본 논문에서는 병합 모션과 분해 모션을 각각 저장하는 두 개의 스택 S_c와 S_s를 사용하였다(<그림 1>참조). 모션병합이 일어날 때마다 두 프레임을 차례대로 S_c에 저장한다. 모션분해 명령이 요구되면 가장 최근에 병합된 두 모션을 위치를 찾은 후에 그 위치에 병합되어 저장된 모션 상태가 S_s에 저장되고 S_c에 저장된 두 모션이 그 위치에 있는 모션을 대치한다.

3.3. 모션 모핑

식 (4)로부터 병합될 연속된 두 프레임이 선택되었다고 하자. 이들 두 모션과 이들 모션의 앞뒤 모션을 이용하여 새로운 모션을 생성하여야 한다. 분명 이 작업에서 가장 중요하게 고려되어야 할 사항이 어떻게 하면 부드러운 모션을 새롭게 생성하는가이다. 이를 위해 본 논문에서는 두 가지 변환, 즉 캐릭터의 루트 위치 구하기와 각 조인트 부분에서의 회전각 구하기를 처리한다.

먼저 캐릭터의 두 모션 X_i와 X_{i+1}을 병합하여 새로운 모션 상태 X_i'를 생성한다고 가정해 보자. 즉 $X_i' = [P_{i0}', Q_{i0}', Q_{i1}' \cdots Q_{im}']^T$을 구해보자. 이를 위해 먼저 캐릭터의 루트 위치인 P_{i0}'을 구해야 한다. 본 논문에서는 네 개의 모션 상태 X_{i-1}, X_i, X_{i+1}, X_{i+2}의 각 루트의 위치 정보 P_{i-10}, P_{i0}, P_{i+10}, P_{i+20}을 보간하여 생성하는 Catmull-Rom 스플라인 커브를 이용하여 P_{i0}'을 구하였다[7]. Catmull-Rom 스플라인 커브를 이용하여 P_{i0}'을 구하는 구체적인 식은 식 (5)와 같다.

$$P'_{i0} = P(u) = P_{i-10}(-su^3 + 2su^2 - su) + P_{i0}((2-s)u^3 + (s-3)u^2 + 1) + P_{i+10}((s-2)u^3 + (3-2s)u^2 + su) + P_{i+20}(su^3 - su^2)$$

(식 5)

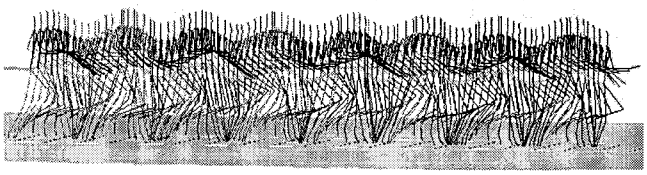
본 논문에서는 식 (5)의 s와 u으로 각각 0.5를 주었다.

또한 캐릭터의 두 모션이 병합되어 새로운 모션을 생성할 때 캐릭터의 각 연결 부위에서의 새로운 회전각을 구해야 한다. 이를 위해 본 논문에서는 Shoemake[8]가 제시하여 쿼터니언(quaternion)의 보간법으로 널리 사용되는 Slerp을 식 (6)과 같이 이용하였다.

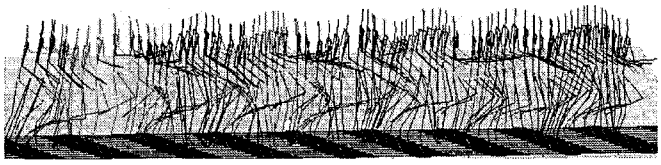
$$Q_{ij}' = Slerp(Q_{ij}, Q_{i+1j}, 0.5), j = 0 \cdots m \quad \text{(식 6)}$$

4. 구현 및 평가

본 논문에서 제안한 점진적 모션 기법은 PC 환경에서 Visual C++ 6.0을 사용하여 구현하였다. 3차원 그래픽스 라이브러리로 산업 표준인 OpenGL을 사용하였다. <그림 2>는 총 101 프레임으로 구성된 뛰는 모션이 병합되는 과정을 각각의 프레임에 따라 어떻게 실행되는지 보여주고 있다. 전체 모션의 병합순서를 (b)~(j) 로 나타내고 있으며 각각 10 프레임 단위로 병합의 모습을 보여주고 있다. 각 병합 단계별 결과를 살펴보면 캐릭터 루트의 전이 및 각 조인트 부분에서의 회전변화가 그리 심하지 않는 부위에서 모션병합이 이루어짐을 알 수 있다. 앞에서 설명한 바와 같이 모션의 분해는 모션병합의 역과정이다. 따라서 <그림 2>의 (j)에서 10 프레임을 추가하면 (i)가 되며, (i)에 또 10 프레임을 추가하면 (h)가 된다. 이러한 작업을 반복적으로 적용하면 초기 모션의 집합을 나타내는 <그림 2>의 (a)가 생성된다. 본 실험에서 식 (3)의 가중치는 $w_r = 0.5$, $w_t = 0.5$로 하였다.

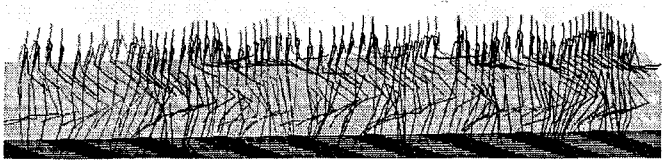


(a) 101 프레임

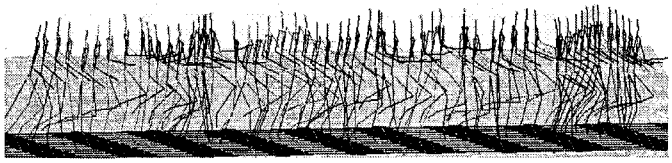


(b) 90 프레임

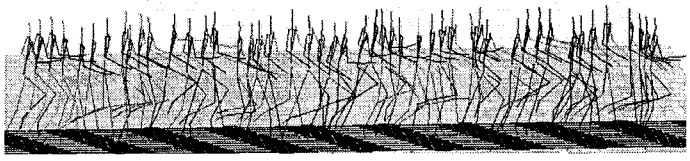
(c) 80 프레임



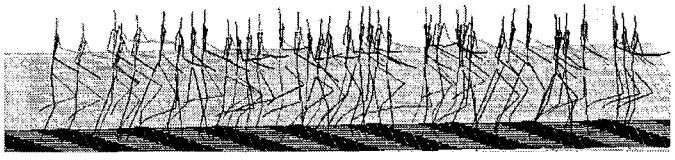
(d) 70 프레임



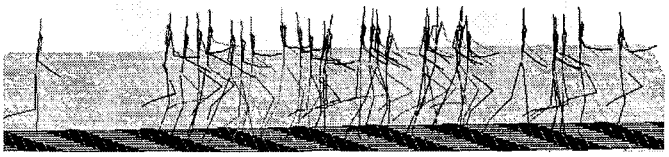
(e) 60 프레임



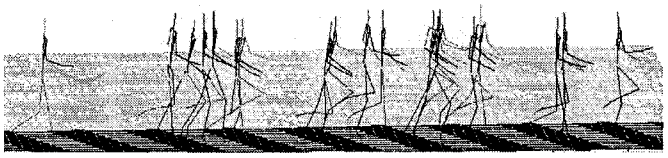
(f) 50 프레임



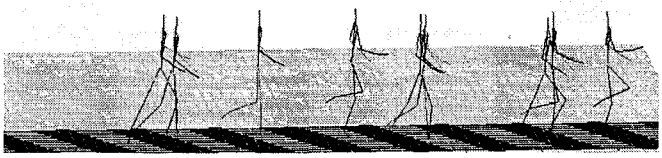
(g) 40프레임



(h) 30프레임



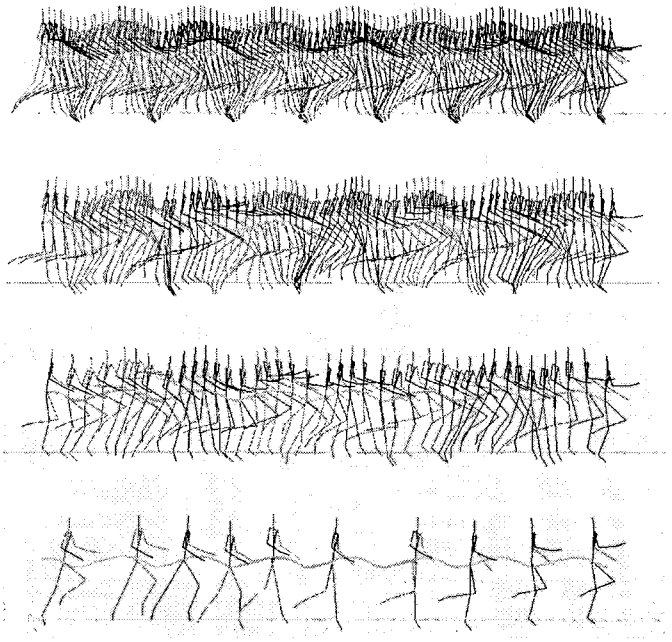
(i) 20 프레임



(j) 10프레임

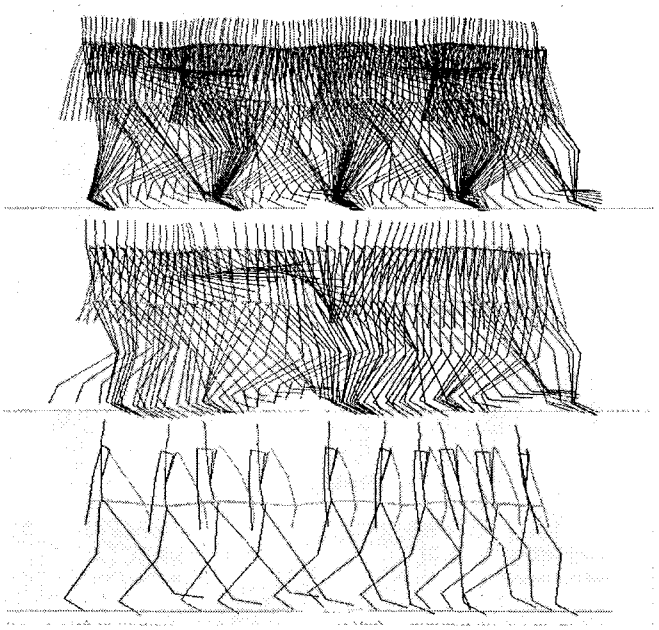
<그림 2> 총 101프레임의 달리기 모션에 적용

<그림 2>에서 보는 바와 같이 40 프레임의 (g)까지는 각 부분 모션이 부드럽게 병합되어 모션의 끊김이 없이 병합된 모션을 보여주지만 40 프레임 이하의 모션의 처음 시작 부분에 병합이 심하게 일어나 연속적인 모션의 표현시 부드러운 모션을 나타내주지 못할 뿐 아니라 계속 프레임을 줄여 병합하게 되면 모션의 이질감은 더욱 커져 자연스러운 모션의 표현이 이루어지지 않는다. 이 문제를 해결하기 위해서는 어떠한 모션을 병합할 것인지 정하는 가장 중요한 식 (3)에서 가중치값을 적절하게 조절하게 되면 처음에 실험한 결과와는 다른 좋은 결과를 얻을 수 있다. <그림 3>에서 기존에 주었던 가중치를 각각 $w_r = 0.5, w_t = 0.5$와 $w_r = 0.1, w_t = 0.9$로 바꾸었을 때의 모션의 변화를 볼 수 있다. <그림 2>와는 다르게 40 프레임 이하에서도 병합과정이 일부분에서 집중적으로 일어나지 않고 모션의 병합후에도 전체적인 모션이 고르게 잘 분포되어 있어 모션의 부드러움을 느낄 수 있다.



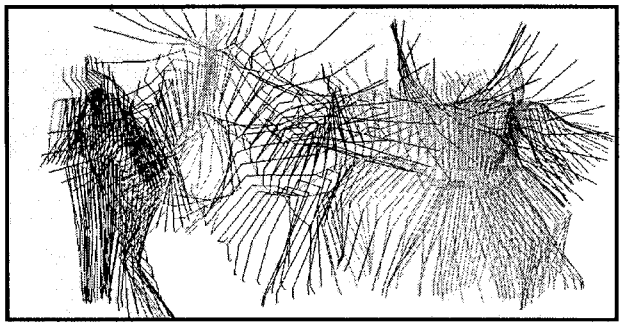
<그림 3> $w_r = 0.1, w_t = 0.9$의 실험 결과 (100, 70, 40, 10 프레임순으로 나열)

그 외의 다른 모션 데이터에도 본 논문에서 제안한 점진적 모션을 적용시켜 보았다. 먼저 모션의 큰 변화가 있는 뛰는 모션이 아닌 걷는 모션을 나타내는 <그림 4>와 같은 모션 프레임에 점진적 모션 기법을 적용하여 보았다. 이렇게 커다란 변화가 없는 모션에는 잘 적용되고 있음을 보여주고 있다. 하지만 <그림 5>에서와 같이 춤을 추는 모션의 프레임의 변화가 큰 모션에서는 문제점을 보이기도 했다. 모션의 동선에서도 표현되고 있지만 앞서 보였던 걷는 모션과 뛰는 모션에 비해 모션의 변화가 상당히 큰 모션데이터이다. 이런 모션 변화가 큰 데이터에서 가장 중요하게 고려해야 할 점은 연속된 모션 중 전체 모션을 대표해서 잘 표현할 수 있는 모션을 남기고 그와 유사한 모션들을 병합하는데 있다.

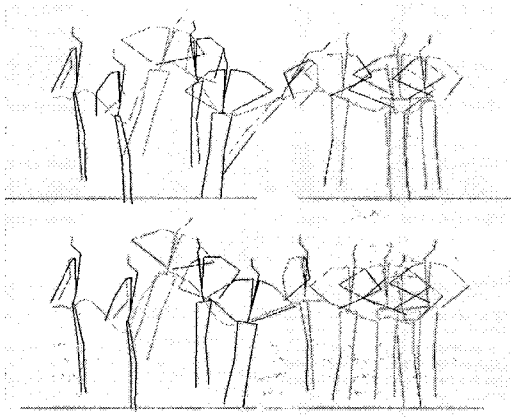


<그림 4> $w_r = 0.3, w_t = 0.7$의 실험결과 (100, 50, 10 프레임순으로 나열)

<그림 5>의 춤추는 모션에 본 논문에서 제안하고 있는 점진적 모션을 적용했을 때의 결과가 <그림 6>에 나타나 있다. <그림 6>는 모션의 변화가 큰 춤추는 모션 데이터를 10 프레임까지 병합했을 경우 결과물을 보여주고 있다. 일반적으로 보통의 모션에서는 문제가 발생하지 않았으나 모션 모핑 과정에서의 중간값을 찾는 보간 특성상 큰 모션의 경우 점점 모션의 크기가 감소되어 <그림 6>과 같은 최종적인 결과에서는 큰 모션은 거의 사라져 버리는 현상이 나타났다. 이와 관련된 부분은 앞으로 연구할 시점 의존과 연관지어 사용자의 시점에서 멀어졌을 경우 보이는 가장 최소 프레임이라고 생각했을 경우 그리 커다란 문제로 여겨지지 않는다. 그러나 아직까지 최적의 가중치 부여와 시점 의존에 관련된 부분, 그리고 큰 모션이 점차 작은 모션으로 바뀌는 문제점을 해결하기 위해 다른 관점에서 모핑에 대한 연구가 필요하다.



<그림 5> 모션의 변화가 큰 춤추는 모션



<그림 6> $w_r = 0.1, w_t = 0.9$의 10 프레임(위)과
$w_r = 0.5, w_t = 0.5$의 10 프레임(아래)

5. 결론

본 논문에서는 Hoppe[2]가 제안한 점진적 메쉬 기술을 캐릭터의 모션에 응용한 점진적 모션 기법을 제시하고 구현하여 실험하였다. 실험된 결과로 이 기법은 프레임의 병합시 부드러운 모션을 생성하는 효과적인 모션 LOD를 효율적으로 제공할 수 있음을 보여주고 있다. 제시된 방법은 매우 효율적인 시간으로 프레임수를 줄여야 하는 오프라인의 모션 편집에 직접 적용이 가능하다. 또한 이 기법은 많은 캐릭터의 실시간 애니메이션에서 연속된 동작의 모핑으로 결국 시점이 멀어지는 캐릭터는 동작 변화가 없는 연속 프레임수가 늘어나게 할 수 있으므로 이를 렌더링 시간에 활용하여 렌더링 시간을 줄일 수 있는 가능성을 가지고 있다. 즉 시점에 따른 연속적(continuous) LOD를 제공하기 위하여 하나의 프레임 안에서 뼈대 단순화에 기반한 기존의 공간적 모션 LOD 기법들과는 다른 새로운 연구의 접근 방법을 생각해볼 수 있는 것이다. 따라서 많은 캐릭터를 필요로 하는 3차원 게임 또는 실시간 애니메이션에 점진적 모션을 적용하기 위하여 다른 시간적 모션 LOD를 가지는 캐릭터들, 즉 시점으로부터 거리가 달라 불변 모션의 연속된 프레임수가 서로 다른 캐릭터들이 있을 때 그 불변성을 렌더링 계산시간의 감소에 활용하는 기법을 연구 구현하고 있다. 이 결과의 시간적 모션 LOD 적용 후 기존의 공간적 모션 LOD를 적용하면 실시간 애니메이션에서 더 큰 효율성을 기대할 수 있다. 또한 병합에 의한 모션 단순화에서 큰 모션이 점차 작은 모션으로 바뀌는 문제점을 해결하는 효과적인 모션의 모핑 방법을, 그리고 최적의 병합 모션을 찾기 위하여 두 모션간의 차이를 정량화하는 메트릭(metric)의 개발에 대해서도 더 연구하고 실험할 필요가 있다.

참고문헌

[1] D. Kim, H. Kim, and S.Y. Shin, Event Driven Crowd Simulation using Example Motions, Technical Report, KAIST, 2002.

[2] H. Hoppe, "Progressive Meshes," Computer Graphics (Proc. of SIGGRAPH 95), pp. 99-108, 1995.

[3] V. Ercegovac and N. Polyzotis, Motion Level of Detail, CS838 Final Report, University of Wisconsin, 2001.

[4] J. Ahn and K. Wohn, "Motion level of detail: a simplification method on crowd scene," CASA 2004, Amsterdam, July 2004.

[5] S. Redon, N. Galoppo, and M.C. Lin, "Adaptive Dynamics of Articulated Bodies," ACM Transection of Graphics, 2005.

[6] J. Lee and S.Y. Shin, "Multiresolution Motion Analysis with Applications," The international workshop on Human Modeling and Animation, Seoul, pp. 131-143, June 2000.

[7] D. Hearn and M.P. Baker, Computer Graphics, Second Edition, Prentice Hall, International Editions, 1994.

[8] K. Shoemake, "Animating rotation with quaternion curves," Computer Graphics (Proc. of SIGGRAPH 85), pp. 245-254, 1985.