Building Efficient, Accurate Character Skins from Examples

Alex Mohr Michael Gleicher

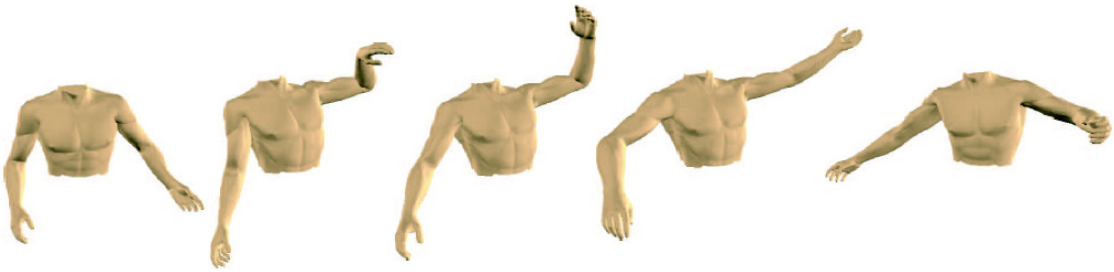

Figure 1: Several still frames of our character from Figure 9 in a new animation sequence.

☐ Abstract

◆ 좋은 캐릭터 애니메이션은 자세하고 세밀한 스킨 변형을 요구
◇ 주로 상용 애니메이션 패키지들에 의해서 제작됨
◦ 사용하기 편리하고 유용성이 높지만, 계산이 느리고 방대한 메모리를 요구
◦ 인터렉티브한 시스템에서 사용 불가

◇ 인터렉티브 시스템들은 계산이 빠르고, 메모리 효율적임
◦ 단, 제작하기 어렵고, 많은 단점("Candy-Wrapper"등)이 존재하는 캐릭터 생성

◆ 이 논문에서는 인터렉티브 시스템을 위해 자동화된 프레임워크를 제시
◇ 정적인 mesh와 쌍을 이루는 스켈레톤으로 구성된 Example의 모음을 이용
◦ 가장 원본 데이터에 근접
◦ 빠른 계산속도
◦ 메모리에 들어갈 정도로 Compact한 Size

1. Introduction

◆ 현실성 있게 하기 위해 캐릭터들은 그들이 움직이는 대로 그럴듯하게 변형되어야 함
◆ 전체 캐릭터 메시를 매 프레임마다 수작업 → 문제 해결, 그러나 실용적이지 않음
◆ 애니메이터들은 기본으로 깔려있는 계층적 스켈레톤을 조종
◇ 캐릭터 메시의 지오메트리는 그 기반이 되는 스켈레톤에 입혀짐.
◇ 스켈레톤이 변형되면 메시도 적절히 변형된다.
◇ 기반이 되는 스켈레톤에 "입혀지는" 모델 지오메트리 → "Skin"
◇ 이것은 골격 파라미터에서 변형 필드로 mapping하는 함수(function)

◆ 스킨을 만드는데 두 가지 기초 사항 - Authoring, Computation.
◇ Skin Authoring : 아티스트가 골격 움직임에 따른 스킨 지오메트리의 움직임을 묘사하기 위해서 어떻게 툴셋들을 사용할 것인지를 의미
◇ Skin Computation : 스켈레톤 구성에서 변형된 메시 지오메트리가 계산되는 방법을 의미

◆ Film과 같은 최고급(high-end) 어플리케이션에서는, 캐릭터의 비주얼한 충실도가 상당히 높음
◇ 아티스트들은 Skin Authoring에서 유용성과 컨트롤을 요구
◇ 상용 툴을 이용해서 세밀한 캐릭터를 만들어내는 데에 여러 가지 방법이 존재함
◦ 근육이나 힘줄과 같은 Skin Substructure를 모델링하는 방법 - [Scheepers - 1997]
◦ 컨트롤 포인트들을 골격 파라미터들과 연결시킴으로써 만들어지는 여러 변형들 또한 가능
◦ FFD 격자[Sederberg and Parry - 1986]나 Wires[Singh and Fiume - 1988] 사용
◦ 최신의 캐릭터들은 종종 이러한 기술들의 조합을 사용 - 캐릭터의 다른 부위들에 각기 다른 툴들이 적당한 경우도 존재

◆ 인터렉티브 시스템은 캐릭터가 빠른 계산과 메모리 사용량이 적을 것을 요구
◇ 캐릭터 계산 모델은 고정되어 있고, 아티스트들은 그것들을 직접 서포트할 수 있도록 그들의 툴셋을 제한해야 함
◇ 게임에서 가장 흔한 스킨 계산
◦ SSD (Skeleton Subspace Deformation)
◦ Enveloping
◦ Smooth Skinning
◦ Linear Blend Skinning
◇ 영향을 주는 조인트(관절부위)들과 블렌딩 가중치를 캐릭터의 각 정점에 할당
◇ 스킨은 조인트들의 로컬 좌표 프레임의 가중치 적용된 조합으로써 각 정점들을 변환
◇ 단점 → 제작이 어렵고 원하지 않는 변형 Artifact 들의 문제
◇ 그러나 널리 사용되는 이유
◦ 애니메이션 데이터의 크기에 상관없이 캐릭터 생성 가능
◦ 런타임에 실행 가능

◆ 인터렉티브 시스템 이전에 사용되었던 캐릭터 계산 메카니즘 → 메쉬 애니메이션
◇ 프레임당 하나씩 스테틱한 메쉬들로 저장
◇ 정적인 모델들은 직접 보여지거나 리얼 타임에 선형 보간되어서 디스플레이 됨
◇ 메시 애니메이션의 특징
◦ 아티스트들에게 캐릭터를 만들 때 필요로 하는 어떤 툴들도 사용할 수 있게 허가
◦ 스킨 Authoring과 Runtime Skin Computation을 분리시킴
◦ 그러나 애니메이션 시퀀스가 짧은 경우에만 적합
◇ 인터렉티브 어플리케이션에서는 대용량의 애니메이션을 사용 → 매 프레임 저장은 비효율적
◇ 이러한 기술은 런타임에 새로운 포즈들을 만들어내는 것이 불가능
◦ 캐릭터의 손을 문의 손잡이에 정확히 맞게 만든다거나
◦ 발 움직임을 계단에 정확히 맞추는 것 등이 불가능
◇ 이러한 제약 때문에 매시 애니메이션은 인기를 잃어가는 추세

◆ 이 논문에서는 예제들을 통해서 컴팩트하게 표현되고 계산이 빠른 캐릭터 스킨을 자동으로 생성해주는 방법을 제시함
◇ 아티스트들에게 인터렉티브 시스템에서의 계산 모델 환경에서 잘 동작하고, 퍼포먼스 요구치도 충족시키는 캐릭터를 어떠한 스킨 제작 툴로도 할 수 있게끔 함
◇ Linear Blend Skinning을 확장하는 프레임워크를 제공

1.1 System Overview

◆ 우리의 시스템에서 스킨을 만드는 것은 두 가지 커다란 과정을 포함
◇ 마야와 같은 애니메이션 패키지에서의 강체 캐릭터로부터 시작
① 캐릭터의 스킨 변형들을 몇몇 포즈들의 캐릭터 지오메트리를 알아봄으로써 샘플링
② 샘플된 데이터를 기반으로 우리의 Underlying 스키닝 모델의 파라미터들을 조절

◆ 우리의 Underlying 모델에 잘 맞는 캐릭터 스킨 변형의 예들을 얻어야 함
◇ 캐릭터를 익스트림 포즈를 포함해서 모든 조인트들을 충분히 움직일 수 있게끔 포즈를 취함
◇ 숙련된 애니메이터가 필요하지 않음
◦ 행동 자유의 정도만을 위한 것이지, 리얼한 동작을 위한 것이 아니기 때문
◇ 이 과정이 다 끝나면, 포즈들은 k번 샘플됨
◇ 샘플링은 유저의 관점에서 매우 쉽게 얻어짐
◦ 유저는 단지 우리가 구현한 마야 스크립트를 Invoke하면 됨
◇ 샘플은 스켈레톤 구성과 그에 해당하는 변형된 스태틱 메시 형태의 스킨 지오메트리로 구성
◇ 이러한 짝지어진 스켈레톤 구성과 스태틱 메시를 "예(Example)"라고 함

◆ 과정 OverView
① 각 정점에 영향을 미치는 조인트들의 셋을 결정 (조인트 추가 과정 포함)
② Bilinear Least Squares 문제를 풀어서 Underlying 스키닝 모델의 파라미터들을 맞춤

◆ 우리가 사용하는 스키닝 모델은 Standard Linear Blend Skinning 모델의 확장판
◇ 확장은 이미 존재하는 조인트들에 연관이 있는 추가 조인트들을 더해주는 것
◇ 새로운 조인트들은 Standard Linear Blend Skinning 모델보다 더욱 풍부한 변형을 캡춰
◇ 엑스트라 조인트를 자동으로 추가하게끔 구성
◇ 유저에게 특정한 엑스트라 조인트 셋을 적절히 조절할 수도 있게끔 함

2. Related Work

◆ 캐릭터 스킨 변형은 캐릭터 애니메이션에서의 기초
◇ [Catmull - 1972] : 처음으로 스켈레톤-driven 기술을 소개
◇ [Burtnyk and Wein - 1976] : 2D 스켈레탈 Bilinear 변형 방법 소개
◇ [Magnenat-Thalmann - 1988] : 초창기 3D 스켈레톤 driven 기술 소개

◆ 최근에는 간단한 스킨에서 시작해서 스킨과 예제들과의 오류를 수정하기 위해 얼마 되지 않는 데이터 보간을 이용하는 방법이 소개됨.
◇ Pose Space Deformation - [Lewis - 2000]
◇ Shape by Example - [Sloan - 2001]
◇ EigenSkin - [Kry - 2002]
◇ 이 세 논문은 리니어 블렌드 스킨의 수정을 위해 Radial Basis 보간을 사용함

◆ 또다른 최근 작업은 이러한 기술을 Range scan data에 적용 - [Allen - 2002]
◇ 예제를 사용한다는 점에서 우리의 것과 비슷
◇ 스킨을 오직 스켈레톤 구조에 의존하는 것이 아닌, 추상적인 파라미터들로 핸들링 가능
◇ 이러한 기술들은 인터렉티브 캐릭터에는 적당하지 않음
◦ 그들은 아주 큰 용량의 예제 데이터 저장을 요구하기 때문
◇ 반면에 우리의 기술은 Fitting 프로세스 뒤에는 모든 예제 데이터를 버린다
◦ 런타임 구조의 사이즈가 입력값(예제)에 비례하지 않음

◆ 다른 저자들은 물리적 시뮬레이션을 적용시킴
◇ 특히 secondary animation에 - [James and Pai - 2002]
◇ 우리의 방법은 직접적으로 보조 변형을 캡춰할 수 없지만 DyRT - [James and Pai - 2002] 는 보조 애니메이션 제작에 적용될 수 있음

◆ 최근의 모델 스키닝 수정(Fitting)에 관련된 작업
◇ Multi-Weight Enveloping - [Wang and Phillips - 2002] or MWE
◦ 우리의 방법에 가장 비슷
◦ MWE는 Linear Blend Skinning을 확장
◦ 각 정점에 영향을 주는 조인트에 하나의 가중치를 주는 것 대신에, 영향을 주는 조인트들의 Transformation Matrix의 coefficient에 하나의 가중치를 할당하는 것
◦ 입력되는 예제들을 이용해서 Linear Least-Square 문제를 풀어서 가중치들을 계산
◦ 우리의 그것과 비슷하지만, 그들은 근본적인 면에서 다름
◦ MWE는 Vertex Weight를 더 많이 더해주고, 반면에 우리는 Joint들을 더해줌

◆ MWE는 Vertex마다 많은 수의 가중치를 사용 (영향을 주는 조인트마다 12개)
◇ 이것은 Least-Square 솔루션에서 Rank Deficient Matrices를 유발할 가능성 존재
◇ 매트릭스의 coefficient들이 Highly Correlated 되어있기 때문
◇ Overfitting을 유발 가능성 존재

◆ 우리의 방법 → 정점에 대한 가중치의 수는 여전히 작다. (영향을 주는 조인트마다 1개)
◇ 추가적인 조인트들은 이미 존재하는 조인트와 명백히 다르게 디자인
◇ Overfitting을 막기 위한 특별한 조치가 필요 없음
◇ 5.2에 설명된 것처럼 작은 양의 Overfitting이 일어나는 것을 발견해내고 다룰 수 있음
◇ MWE는 우리의 방법처럼 그래픽 하드웨어에 의해 쉽게 가속되기 어려움
◇ 우리의 방식은 이미 존재하는 소프트웨어 기반구조에 별다른 변화 없이 사용될 수 있음

3. Linear Blend Skining

◆ 전통적인 인터렉티브한 스키닝 모델
◇ SSD (Skeleton Subspace Deformaton)
◇ Smooth Skinning - Called by MAYA
◇ Linear Blend Skinning - 우리가 부를 이름
◇ 이러한 기술들은 인터렉티브 어플리케이션들에 널리 사용됨

◆ 계층적 뼈대 구조를 정적인 캐릭터 모델(전형적으로 neutral한 포즈) 안에 놓음으로써 작동
◇ 이런 초기 캐릭터 포즈를 "Dress Pose"라 함
◇ 각 정점들은 각 영향에 따라 영향을 미치는 조인트와 블렌딩 가중치가 할당됨
◇ 특정 포즈에서의 변형 계산은 각 영향 조인트들에 의한 강체 변형을 포함
◇ 블렌딩 가중치는 이러한 강체 변환 위치들을 합치기 위해 쓰임
◇ 스켈레톤 구조 c에서의 변형된 정점 위치 $\overline{V_c}$는 다음과 같이 계산된다.

<Equation ①>

$$\bar{\mathbf{v}}_c = \sum_{i=1}^{n} w_i M_{i,c} M_{i,d}^{-1} \mathbf{v}_d$$

◦ w_i : 가중치
◦ V_d : 정점 V의 Dress Pose 위치
◦ $M_{i,c}$: c라는 구조에서의 i번째 조인트에 관련된 변형 매트릭스
◦ $M_{i,d}^{-1}$: i번째 영향에 관련된 Dress Pose 매트릭스의 역행렬
◦ $M_{i,d}^{-1} V_d$: i번째 영향의 Local Coordinate Frame에서의 V_d의위치

◆ 이런 알고리즘은 복잡한 변형은 표현하지 못하며, “Candy-Wrapper” 현상과 같은 단점 존재
◇ “Candy-Wrapper” - 손목 관절이나 구부러지는 조인트 등에서 일어나는 현상 (그림 참조)

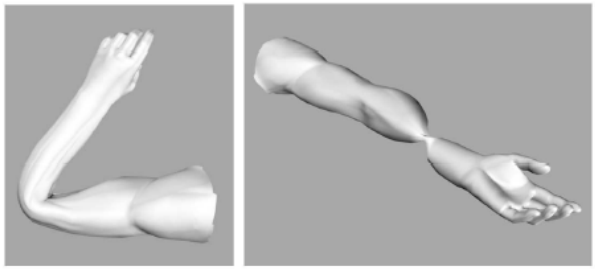

Figure 2: Common problems with linear blend skinning: the bent arm on the left demonstrates shrinkage around bent joints such as the elbow and knee while the twisted wrist on the right demonstrates the “candy-wrapper” collapse effect. These artifacts are caused by blending dissimilar transformations.

◆ 이러한 단점들은 정점들이 선형 보간된 매트릭스들에 의해 변형되기 때문
◇ 만약 보간된 매트릭스가 비슷하지 않다면(예:180도 회전), 보간된 변형은 변질되어 버려서 지오메트리는 반드시 Collapse됨
◇ 이런 단점에도 불구, 매우 빠르고 상업 어플리케이션에 널리 적용되기 때문에 여전히 유명함

4. Extending Linear Blend Skinning

◆ Linear Blend Skinning 모델은 <그림 3>과 같이 변형을 캡춰하기에 충분하지 않음

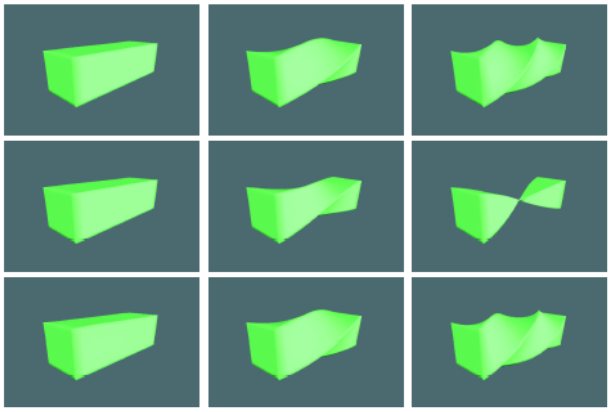

Figure 3: Top Row: Three examples of a twisting box driven by a nonlinear deformer. Middle Row: Solved linear blend skin using one joint. Bottom Row: Our result with just one additional joint that half interpolates the twist rotation.

◆ 이러한 문제는 트위스트가 180도 가까이 되었기 때문
◇ 선형 보간된 변형은 그들 자신의 모양보다 더욱 다르게 붕괴되는 경향이 있다.
◇ 관절 조인트들, 무릎이나 팔꿈치 등에서 볼륨의 loss 역시 관찰됨 (<그림 2> 참조)

◆ 너무 Dissimilar한 블렌딩 변형들을 피함으로써 이런 문제를 해결
◇ Collapse 없이 적당히 보간되는 변형을 추가
◇ 손목 트위스팅의 경우, 회전 각을 보간하면서 Collapse되지 않는 추가적인 조인트를 추가

◆ 더욱 일반적으로, 변형 효과들을 적당히 변형시키는 조인트들을 추가시킴으로써 어떠한 변형 효과도 얻어낼 수 있음을 확인
◇ 근육 부풀어 오름의 경우, 근육이 부풀어야할 때는 위쪽으로 스케일링되는 조인트를 추가
◇ 손목의 경우 변형을 잘 표현할 수 있도록 잘 움직이고 스케일링 되는 몇 개의 조인트를 추가

◆ 그러나 너무 많은 추가 조인트들을 더하는 것은 실용적이지 않음
◇ 많은 수의 조인트를 더하는 것은 퍼포먼스에 심각한 영향을 미칠 수 있음
◇ 만일 입력 예제들에서 이러한 변형들을 찾아낸다고 해도, 모든 포즈들에서 이러한 변형들과 스켈레톤 파라미터들 사이에 작용하는 일반적인 관계를 결정하는 방법은 명확하지 않음
◇ 이러한 Relationship에 대한 지식 없이도, 우리의 스키마는 오직 입력 프레임들을 재생산할 수만 있지, 새로운 포즈들에 대해서는 동작하지 않을 수도 있음

◇ 대신에, 우리는 전통적인 Linear Blend Skinning 모델을 간단히 Original 스켈레톤 파라미터들에 연관되어 있는 비교적 작은 수의 조인트를 추가함으로써 확장하고 그들을 이용해 Fitting함
◇ Standard Linear Blend Skinning에서 실패한 곳에서 examining하고, 우리가 캡춰하고자 하는 추가적인 캐릭터 변형도 examining해서 추가적인 조인트들을 고름
◇ 이 확장된 스켈레톤을 이용해서 스키닝 모델의 파라미터들을 수정
◇ 정점들이 변형들의 가중치된 합을 고르기 때문에, 만일 추가된 조인트의 어떠한 선형 스케일링이라도 이익이 된다면, 그 조인트는 사용됨

◆ 우리의 방법이 캐릭터에 관한 관찰을 기반으로 해서 더 좋은 변형과 추가적인 조인트들을 얻어내는 프레임워크라는 것을 강조
◇ 다른 변형의 다른 캐릭터들은 다른 추가적인 조인트들을 요구할지도 모름
◇ 그러나 한번 이러한 조인트들이 결정되면, 우리의 Fitting 알고리즘을 사용해서 Skin들이 해결될 수도 있음

4.1 Additional Joints

◆ 지오메트리의 Collapsing 문제를 해결하기 위해 우리의 시스템은 자동으로 적당히 Collapsing 없이 회전을 보간하는 조인트들을 더해줌
◇ 이것은 드레스 포즈와 연관된 조인트의 회전을 조사해봄으로써 할 수 있음
◇ 새로운 조인트를 같은 포지션에 위치한 이 회전의 Halfway Spherical Linear Interpolation으로 계산함
◇ 보간 파라미터들을 똑같이 배분한 더욱 많은 조인트들이 더해진다면, 모양은 더욱 좋겠지만, 우리의 실험 결과 단 하나의 보간된 회전만으로도 충분하다는 것을 알 수 있음

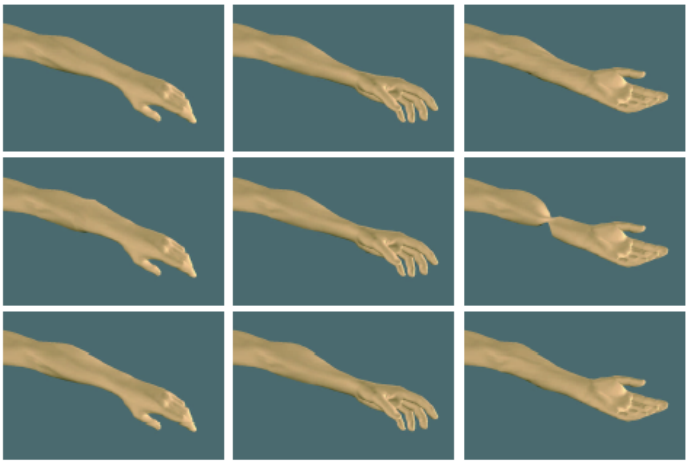

Figure 4: Top Row: Original examples of a twisting wrist. Middle Row: Linear blend skin approximation. Bottom Row: Our result using one additional joint.

◆ <그림 4>는 단지 하나의 보간된 회전 조인트를 추가함으로써 얻어진 손목 경우의 향상된 점을 보여줌 (<그림 5>는 구부러진 팔꿈치의 경우)

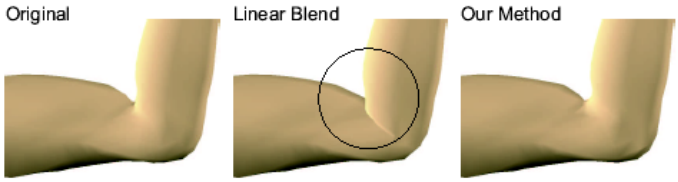

Figure 5: Linear blend skinning alone is incapable of capturing correct creasing around elbows. At the left is an example of a bent elbow. In the middle is the linear blend skin approximation. notice the interpenetration. In contrast, our method avoids the interpenetration.

◆ Linear Blend Skinning에서 구현하기 어려운 또 하나의 예
→ 근육과 힘줄 등에 의한 부풀어 오름과 움푹 패임 현상
◇ 캐릭터 관찰 결과, 근육과 힘줄 등에 의한 변형은 단지 조인트들의 각도에 연관되어 있음
◇ Ex) 이두박근은 팔꿈치 근처가 확장되었을 때에는 작고, 최대로 구부러졌을 때 가장 큼
◇ 이러한 효과들을 캡춰하기 위해서 특정 조인트들의 각도에 따라서 스케일 Up 되고 Down 되는 조인트 하나를 추가

◆ 다음과 같은 방식으로 스케일링 조인트들을 더해감
① Original 스켈레톤으로부터 새로운 조인트의 스케일링 파라미터를 Drive할 조인트를 찾음
② Driver가 선택되면, 2개의 조인트 Set을 추가
③ 첫 번째 Set → Driver의 “Upstream”
◦ Driver와 그의 부모를 연결시키는 뼈의 중앙에 위치
④ 두 번째 Set → Driver의 “Downstream"
◦ Driver와 그의 자식을 연결시키는 뼈의 중앙에 위치

◆ 모든 "Upstream" 조인트들은 모두 같은 방식으로 구성 (<그림 6> 참조)

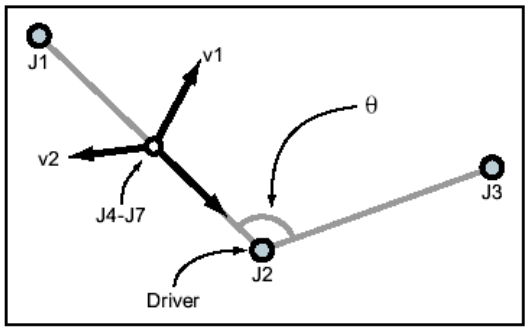

Figure 6: Our method adds extra joints to characters to help better approximate deformations. Here $J4$ through $J7$ are automatically added upstream joints that scale depending on the angle θ. As θ decreases, $J4$ scales up in the direction $v1$ which is orthogonal to the bone connecting $J1$ and $J2$. Meanwhile, $J5$ scales up in the direction $v2$, orthogonal to both the bone and $v1$. $J6$ and $J7$ operate similarly, but scale down as θ increases rather than up. Downstream joints are very similar except that these joints are positioned on the bone from $J2$ to $J3$.

◆ 우리는 4개의 upstream 조인트를 사용했음
◇ 그 중 2개는 Scale Up 되고, 이에 상응하는 짝은 Scale Down 됨
◇ 이러한 조인트들의 스케일 파라미터들은 Driver와 부모, 또는 자식을 연결하는 각도를 기반으로 설정됨
◇ 만일 Driver가 자식을 여럿 가지고 있다면, 각도를 결정할 때에는 Driver와 그 자식들을 연결하는 뼈들의 합을 나타내는 벡터가 쓰임

◆ Scale Up 되는 조인트에 대해 Scale 파라미터 s는

$$s = 1 + \frac{k}{2}\left(\frac{\mathbf{b}_1 \cdot \mathbf{b}_2}{\|\mathbf{b}_1\|\,\|\mathbf{b}_2\|} + 1\right)$$

b_1, b_2 : Driver 조인트의 각도를 재기 위한 Bone Vector들
k : b1과 b2사이의 각이 0도일 때의 최대 scale factor

◆ Scale Down되는 경우에는 Scale 파라미터가 단지 s^{-1}이 됨
◆ k값은 유저에 의해서 선택될 수도 있지만, 우리의 예에서는 8이 잘 작동했음
◇ 정점들은 새로운 조인트들의 어떠한 Scaling도 택할 수 있으므로 큰 값도 상관없음
◇ Ex) 정점이 사실 8이 아니라 2로 곱해지는 조인트가 필요하다 → 가중치를 1/4로 설정

5. Fitting the Skinning Model

◆ 위의 과정을 마친 후, 스키닝 모델을 예제들과 잘 맞추기 위해 Fitting 프로시저 수행
◇ Fitting 과정의 입력은 "예제"들
◇ 예제들은 단지 스켈레톤들과 연결된 정적인 캐릭터 메시
◇ 정적인 메시는 스켈레톤 구조가 바뀜에 따라서 변형됨

◆ Linear Blend Skinning 계산 → <Equation 1>
◇ 이러한 스키닝 모델을 검사하는데, 오직 Mi만이 미리 계산됨
◇ 이것들은 캐릭터에 있는 모든 조인트들과 연관된 코디네이트 프레임들
◇ 다시 말하면, 각 정점마다, 영향 미치는 조인트의 Set과, 영향 가중치와(w_i), 드레스 포즈의 정점 위치(V_d)를 선택할 수 있다는 것
◇ 우리는 이 정보들을 예제들과 잘 들어맞게끔 하는 것들로 맞추고자 함

5.1 Finding Influence Sets

◆ 영향 셋을 우선 설정하는 이유
◇ 이상적으로, 영향 셋은 가중치 계산 프로시저 과정에서 자연스럽게 나와야 함
◇ 연관이 없는 조인트들은 자연스레 가중치가 0이 될 것임
◇ 그러나 이것은 실제로 일어나지 않음 → 우리의 샘플링이 Exhaustive하지 않기 때문

◇ 정점에 더 많은 조인트가 영향을 미치면, 스킨 만들어내는데 시간은 더 느려짐
◇ 요즘의 하드웨어들은 오직 제한된 정점 명령어들만 지원
◇ 따라서 우리는 효과가 좋은 작은 셋을 골라야 함

◆ 최근의 조사에서, 영향 셋 결정은 유저에게 맡겨짐
- [Lewis - 2000], [Wang - 2002], [Sloan - 2001]
◇ 이 작업은 각 조인트에 영향을 미치는 지역을 "Painting"함으로써 수행
◇ 우리의 시스템은 자동으로 이러한 영향 셋을 휴리스틱 알고리즘을 사용함으로써 결정

◆ 캐릭터 스킨에 있는 정점들은 대체로 특정 조인트에 대해 Rigid 하게 변환 된다는 걸 발견
◇ Ex) 하박(아래 팔)위의 점은 단순히 하박을 따라감
◇ 대부분의 캐릭터에서 스킨들은 그들이 바운드된 조인트들에 의해 가장 크게 영향을 받음
◇ 이두박근 위의 포인트가 팔이 움직임에 따라 정확히 Rigid 하지 않다고 해도 (근육의 부풀어 오름 현상 때문에) 이러한 포인트들은 여전히 상박에 가장 영향을 많이 받기 때문에 그것에 의해서 영향을 받아야만 함
◇ 정점이 모든 예제들의 모든 조인트들과 얼마나 Rigid하게 변형되는지를 조사하고, 가장 Rigid 하게 변형되는 조인트들을 영향 셋으로 사용

◆ 각 정점에 대해서, 조인트에 대한 Rigid 스코어는 다음과 같이 계산

각 예제들에 대해서, 예제의 로컬 코디네이트 포지션은

$$M_{i,e}^{-1} V_e$$

$M_{i,e}$: e번째 예제의 i번째 조인트와 연관된 코디네이트 프레임

V_e : 정점의 e번째 예제의 Global 좌표

◇ 모든 예제들에서의 로컬 코디네이트 포지션들은 Point Cloud를 구성한다. (<그림 7> 참조)

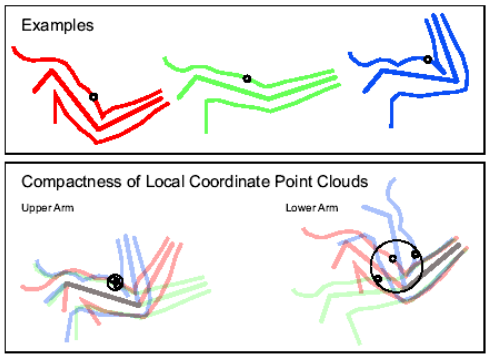

Figure 7: Top: A set of three examples of a deforming arm mesh with a bulging bicep. A particular point on the arm mesh is highlighted in each example. Bottom Left: Each example rotated and aligned so that the upper arm bones coincide. The highlighted points form a cloud in the local coordinate frame of the upper arm. Even though the bicep bulges significantly, this cloud is compact. Lower Right: A similar point cloud but relative to the forearm. This cloud is far less compact than the former, making the forearm a poorer choice for an influence.

◇ 포인트 구름이 더욱 간결할수록 정점의 관계가 더욱 Rigid 함
◇ 포인트 구름의 간결함을 그 구름의 지름을 계산함으로써 연산

◆ 우리는 각 포인트와 모든 다른 포인트를 비교할 수 있는 $O(n^2)$ 알고리즘을 발견
◆ [Malandain - 2002] 논문에 있는 알고리즘을 사용하면 $O(nlogn)$ 시간에 계산 가능

◆ 정점에 대해 모든 조인트들에 대한 간결함이 모두 계산되면, 가장 작은 k값이 그 정점의 영향 셋으로 선택됨
◇ 일정한 임계치를 두고 영향 셋을 선택하는 것이 좋을까? → 문제점 발견
◇ Rigid 정도치가 커질수록, 그들은 더욱 의미가 없어지기 때문에, 좋은 임계치를 설정하는 것이 명확하지 않음
◇ Ex) 왼쪽 어깨 위의 한 점이 오른쪽 발보다 왼쪽 발에 더욱 영향을 많이 받는 경우가 생길 수 있는데, 하지만 두 경우 다 영향을 미치지는 않음
◇ 큰 Rigid 스코어는 특정한 의미가 없기 때문에, 의미 있는 임계치를 설정하기가 불가능

◆ 영향 셋들은, 유저가 원한다면 k값을 선택할 수 있게 하도록 해야 함
◇ 우리의 경험상, 캐릭터의 복잡도에 따라서 3에서 8개의 영향들이 잘 작동함

5.2 Solving for Weights and Vertices

◆ 영향 셋들이 결정되면, 가중치(w_i)와 드레스 포즈 정점 위치(V_d)만 남음
◆ 모든 예제에 대해 스킨과 예제들 사이의 차이를 가장 적게 하는 정점들과 가중치를 얻고자 함

$$\min \left\| \sum_{i=1}^{n} \bar{\mathbf{v}}_{e_i} - \mathbf{v}_{e_i} \right\|^2$$

V_{e_i} : i번째 예제에서 입력 정점 위치

$\overline{V}_{e_i}$: i번째 예제의 구조에서 스키닝 모델에 의해 계산된 변형된 정점

$$\bar{\mathbf{v}}_e = \sum_{i=1}^{n} w_i \mathbf{M}_{i,e} \mathbf{M}_{i,d}^{-1} \mathbf{v}_d$$

◆ 이 문제는 가중치와 정점에서 Bilinear 함
◆ 우리는 최적화를 위해 대안적 방법을 사용
① 우선 첫 번째 Variable을 Fix하고, Linear Least-Square Problem을 풀어서 Second를 찾음
② 그 뒤에 두 번째를 Fix하고, 첫 번째를 위해 Linear Least-Square Problem을 계산
③ 이러한 과정을 Converges(수렴하다) 할 때까지 반복
◦ 이 기술은 널리 사용된다. - [Freeman - 1997]
④ 가중치 계산을 시작하는데, 우리는 좋은 추측이 없지만, 초기 드레스 포즈의 정점이 이상적이라는 것은 알고 있음
⑤ 그 다음, 우리는 수정된 가중치를 Hold해놓고 정점 위치를 계산
◦ 이러한 과정은 전형적으로 한번이나 두 번의 반복 뒤엔 수렴함

◆ 우리는 많은 예제에서부터 작은 수의 가중치를 계산
◇ 따라서 입력 데이터가 잘 샘플되어 있다면 우리의 시스템은 잘 동작함
◇ 또한 Overfitting 문제도 없음
◇ 따라서 우리는 특별한 예방을 할 필요가 없음

◆ 명확성을 위해, Least Square를 통해 계산하는 행렬을 Block Form으로 제시함

$$T_{i,e} = M_{i,e} M_{i,d}^{-1}$$

◆ 결과로 나온 가중치가 Affine이라는 것을 확실히 하기 위해

$$w_1 = 1 - \sum_{i=2}^{n} w_i$$

◆ 그리고 w_2부터 w_n까지 계산

$$\begin{bmatrix} (T_{2,e_1} - T_{1,e_1})\mathbf{v}_d & \cdots & (T_{n,e_1} - T_{1,e_1})\mathbf{v}_d \\ \vdots & \ddots & \vdots \\ (T_{2,e_k} - T_{1,e_k})\mathbf{v}_d & \cdots & (T_{n,e_k} - T_{1,e_k})\mathbf{v}_d \end{bmatrix} \begin{bmatrix} w_2 \\ w_3 \\ \vdots \\ w_n \end{bmatrix} = \begin{bmatrix} \mathbf{v}_{e_1} - T_{1,e_1}\mathbf{v}_d \\ \vdots \\ \mathbf{v}_{e_k} - T_{1,e_k}\mathbf{v}_d \end{bmatrix}$$

◆ 정점 위치를 계산하기 위해 사용된 매트릭스는 다음과 같다

$$\begin{bmatrix} \sum_{i=1}^{n} w_i T_{i,e_1} \\ \vdots \\ \sum_{i=1}^{n} w_i T_{i,e_k} \end{bmatrix} [\mathbf{v}_d] = \begin{bmatrix} \mathbf{v}_{e_1} \\ \vdots \\ \mathbf{v}_{e_k} \end{bmatrix}$$

◆ 호모지니어스 코디네이트를 다루기 위해 $\sum_{i=1}^{n} w_i T_{i,e_k}$ 매트릭스의 Translation 부분은 우변의 V_e 에서 Subtract 되었음

◆ 우리는 Least-Square Problem을 Singular Value Decomposition을 이용해서 계산
◇ 매트릭스가 Overfitting을 유발하는 Rank Deficient일 때 그것을 검사할 수 있음
◇ 가장 큰 싱글러 값과 가장 작은 것의 비율을 비교함으로써 검사
◇ 복구하기 위해, 이러한 Singular 값들을 0으로 하고, Fitting 과정을 다시 시작함
◇ Overfitting이 문제라면, [Wang - 2002]의 논문에 쓰였던 예방법들이 사용될 수 있음
◇ 그러나 이 논문에서의 어떠한 예제에서도 싱글러 값이 0으로 되지 않았음

5.3 Handling Normals

◆ 노말 벡터들도 잘 계산되는 것이 아주 중요 → 정확한 조명 계산을 위해
◇ 노말은 정점마다 계산되는 것을 가정한다.

$$\bar{\mathbf{n}}_c = \frac{(\sum w_i M_{i,c} M_{i,d}^{-1})^{-T} \mathbf{n}_d}{\left\| (\sum w_i M_{i,c} M_{i,d}^{-1})^{-T} \mathbf{n}_d \right\|}$$

◆ 이 식은 부드러운 표면의 지역적 이웃들에 유효함
◇ → 우리의 경우에는 부드러운 표면을 가지고 있지 않음
◇ 대신, 우리는 독립적으로 계산된 하나의 포인트가 있음
◇ 위의 노말 계산 식은 블렌드된 변환들이 Pure Rotation이 아닐 때 문제를 초래할 수 있음

◆ 인터렉티브 시스템들은 전형적으로 노말 계산을 다음과 같이 수행함

<Equation ②>

$$\bar{\mathbf{n}}_c = \frac{\sum w_i M_{i,c}^{-T} M_{i,d}^{T} \mathbf{n}_d}{\left\| \sum w_i M_{i,c}^{-T} M_{i,d}^{T} \mathbf{n}_d \right\|}$$

◇ 이유 → 각 스텝에서 회전과 스케일링은 특별한 Inverse 폼을 가지고 있기 때문에 조인트 매트릭스와 그들의 역행렬을 계산하는데 종종 더 빠르기 때문
◇ 따라서 일반적인 Inversion 방법을 사용할 이유를 경감시킴
◇ [<EigenSkin>: Kry-2002] 에서 노말들은 두 번째 스키닝 문제로 다루며, 독립적으로 계산
◇ 우리의 시스템에서, <Equation ②> 시스템에 사용된 모델을 사용했고, 최적화 과정에 노말도 포함시켰음

6. Results

◆ 일반적으로 비디오 게임이나 다른 인터렉티브한 어플리케이션에 사용된 간단한 Linear Blend Skinning 모델은 아주 빠르고 간결함
◇ 그러나 High Quality의 변형은 캡춰하지 못함
◇ 우리의 Linear Blend 모델을 확장한 프레임워크는 효율성을 유지하면서도 더욱 재미난 변형들을 캡춰하는 것을 가능하게 함

◆ 터무니 없는 Linear Blend Skinning의 단점들을 우리의 방식으로 해결함

◆ Linear Blend Skinning의 문제를 해결함과 동시에 추가적인 요소들도 캡춰 가능
◇ <그림 8>은 어떻게 우리가 캐릭터의 팔에서 이두박근과 삼두박근의 부풀어 오름 현상을 캡춰할 수 있었는지를 보여줌

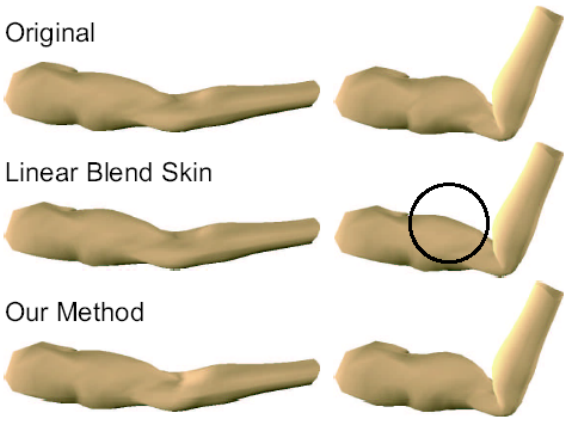

Figure 8: Top: Examples of a muscular arm flexing. Middle: Linear blend skin approximation. Note the lack of bicep bulge. Bottom: Results using our method.

◆ 특정 추가 조인트가 캐릭터의 모든 변형을 위해 충분하지 않을 수도 있지만, 우리의 방법을 이용하여 추가된 또 다른 조인트는 이러한 문제를 해결해줄 것임

◆ 우리의 시스템은 팔과 다리만이 아닌, 다른 부위에도 적용 가능 (<그림 9> 참조)
◇ 이 그림은 이 캐릭터가 새로운 포즈에서의 모습도 보여줌
◇ 우리의 스킨은 새로운 포즈에도 잘 일반화 됨

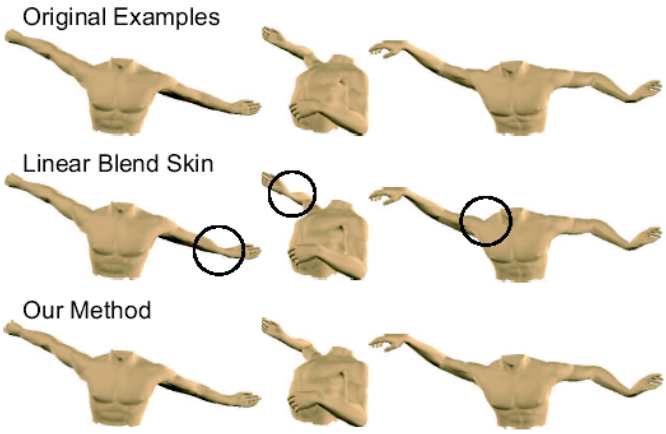

Figure 9: Top Row: Examples of an upper body rigged in Maya. Middle Row: Best linear blend skin. Note the circled problem areas. Bottom Row: Our results.

◆ 우리의 솔루션 과정은 일반적으로 아주 빠름
◇ 이 논문에 제시된 어떤 예제들도 요즘의 일반 컴퓨터에서 5분 이상도 걸리지 않았음
◇ 가장 느린 것 → 6000개의 정점을 가진 상체 모델, 50개의 예제, 5개의 정점당 영향 조인트
◇ 정점당 연산 시간은 “영향”의 개수와 “예제”들의 개수에 의존적
◇ 각 정점은 독립적으로 계산되기 때문에 우리의 알고리즘은 쉽게 병렬화 가능

6.1 Applications

◆ 예제들을 통한 캠팩트하게 표현되고 계산이 빠르고 높은 퀄리티의 스킨 근사는 매우 유용함

◆ 현재 시스템들의 단점들
① 요즘의 많은 인터렉티브 시스템들은 오직 Linear Blend Skinned 캐릭터만을 지원
② "Candy-Wrapper"와 같은 변형 문제들
③ 스킨들을 Authoring하기도 어려움
④ 블렌딩 가중치와 영향 셋들을 설정하는 것도 스킨 Author가 해야 함
⑤ 애니메이션 시스템은 어떠한 직관적이고 유용한 변형 프리미티브도 제공하지 않을지도 모름

◆ 우리의 시스템을 이용함으로써 얻는 장점들
① 캐릭터 Author들은 캐릭터를 만들 때 자기가 좋아하는 어떠한 툴들도 사용 가능
② 시스템이 요구하는 것은 예제 Set들 뿐임
③ Author가 영향 셋들이나 가중치들을 하나하나 설정하지 않아도 됨
④ 우리의 스킨들이 Linear Blend Skins와 같은 방식으로 계산되기 때문에 그래픽 하드웨어의 가속을 받을 수 있음

◆ 하나의 스켈레톤에 붙어있던 캐릭터를 다른 기반 스켈레톤으로 Mapping하는 것이 가능함
◇ 이것을 “Skin Retargeting”이라고 부름
◇ 특정 인터렉티브 시스템이 특정한 스켈레톤을 가진 캐릭터를 요구할 때 유용
◇ Ex) 비디오 게임이 특정한 스켈레톤 토폴로지를 지닌 캐릭터를 위한 최적화된 엔진을 가지고 있는 경우.
◇ 보통의 경우에는, 만일 캐릭터가 다른 스켈레톤을 가지고 생성되었다면, 그 캐릭터는 새로운 스켈레톤에서 작동하려면 수동으로 Re-Rigged되어야 함
◇ 우리의 시스템을 이용하면 이것은 좀더 쉽게 구현될 수 있음
◇ 새로운 스켈레톤과 상응하는 포즈들과 짝지어진 오리지날 스켈레톤에 의해 변형된 예제 메시들을 익스포트하면 됨
◇ 적당한 영향 셋과 블렌딩 가중치를 자동으로 계산

◆ High-End 애니메이션 분야
◇ High-End 캐릭터들은 종종 인터렉티브하게 계산되지 않는 복잡한 변형을 가지고 있음
◇ 따라서 애니메이터들은 전형적으로 캐릭터의 대략적인 모습만을 보여주는 자세하지 않은 버전으로 작업을 한다.
◇ 우리의 시스템을 이용하면 변형된 캐릭터에 더욱 비슷한 인터렉티브 캐릭터 제작이 가능해짐