LLE 알고리즘을 사용한 얼굴 모션 데이터의 투영 및 실시간 표정제어

Realtime Facial Expression Control and Projection of Facial Motion Data using Locally Linear Embedding

김성호
상지대학교 컴퓨터정보공학부

Sung-Ho Kim(kimsh1204@sangji.ac.kr)

요약

본 논문은 얼굴 모션 캡쳐 데이터를 재사용하여 실시간 표정 제어 및 표정 애니메이션을 생성하기 위한 방법론을 기술한다. 이 방법의 핵심요소는 얼굴 표정들을 정의할 수 있는 표정상태 표현법을 정하고, 이를 LLE 알고리즘에 적용하여 표정들을 적당한 공간에 분포시키는 방법론과, 이 공간을 사용하여 실시간 표정 애니메이션 생성 및 표정제어를 수행하기 위한 사용자 인터페이스 기법이다. 본 논문에서는 약 2400개의 얼굴 표정 프레임 데이터를 이용하여 공간을 생성하고, 애니메이터가 이 공간을 자유롭게 항해할 때, 항해경로 상에 위치한 얼굴 표정 프레임 데이터들이 연속적으로 선택되어 하나의 애니메이션이 생성되거나 표정제어가 가능하도록 하였다. 약 2400개의 얼굴 표정 프레임 데이터들을 직관적인 공간상에 분포하기 위해서는 얼굴 표정 프레임 데이터로부터 얼굴 표정상태를 표현할 필요가 있고, 이를 위해서는 임의의 두 마커 사이의 거리들로 구성된 거리행렬 벡터를 이용한다. 직관적인 공간에서의 데이터 배치는 얼굴 표정상태벡터들의 집합을 LLE 알고리즘에 적용하고, 이로부터 2차원 평면에 균일하게 분포하였다. 본 논문에서는 애니메이터로 하여금 사용자 인터페이스를 사용하여 실시간으로 표정 애니메이션을 생성하거나 표정제어를 수행하도록 하였으며, 그 결과를 평가한다.

■ 중심어 : | 얼굴 표정 애니메이션 | 표정제어 | 표정상태벡터 | 거리행렬 | LLE |

Abstract

This paper describes methodology that enables animators to create the facial expression animations and to control the facial expressions in real-time by reusing motion capture datas. In order to achieve this, we fix a facial expression state expression method to express facial states based on facial motion data. In addition, by distributing facial expressions into intuitive space using LLE algorithm, it is possible to create the animations or to control the expressions in real-time from facial expression space using user interface. In this paper, approximately 2400 facial expression frames are used to generate facial expression space. In addition, by navigating facial expression space projected on the 2-dimensional plane, it is possible to create the animations or to control the expressions of 3-dimensional avatars in real-time by selecting a series of expressions from facial expression space. In order to distribute approximately 2400 facial expression data into intuitional space, there is need to represents the state of each expressions from facial expression frames. In order to achieve this, the distance matrix that presents the distances between pairs of feature points on the faces, is used. In order to distribute this datas, LLE algorithm is used for visualization in 2-dimensional plane. Animators are told to control facial expressions or to create animations when using the user interface of this system. This paper evaluates the results of the experiment.

■ keyword : | Facial Animation | Expression Control | Facial State Vector | Distance Matrix | LLE |

접수번호 : #061031-004
접수일자 : 2006년 10월 31일
심사완료일 : 2007년 01월 29일
교신저자 : 김성호, e-mail : kimsh1204@sangji.ac.kr

I. 서 론

컴퓨터 애니메이션 기법을 기반으로 하고 있는 각종 멀티미디어 분야 중에서, 특히 인간의 얼굴 표정 애니메이션을 제작하거나 3차원 아바타의 표정제어를 수행하기 위한 과정은 매우 많은 수작업을 비롯하여 시간적, 경제적으로 매우 많은 투자를 해야 하는 어려움이 존재하고 있다. 또한 최근까지 컴퓨터 애니메이션 분야를 연구하는 전문가들의 수많은 연구 결과[1-3][7-10]에도 불구하고 아직까지 부족한 부분이 많아 지금도 다양한 방법으로 연구들이 진행되고 있다. 그만큼 인간의 얼굴 표정을 실제 인간처럼 자연스럽게 표현한다는 것이 매우 어렵다는 것이다. 최근에는 모션 캡쳐를 사용한 캐릭터 애니메이션이 컴퓨터 애니메이션 분야에서 각광받으면서 배우의 얼굴 모션을 캡쳐하여 3차원 캐릭터 애니메이션에 적용하고 있다. 캡쳐한 데이터를 사용하는 방법으로 크게 두 가지가 제시되었다. 첫째는 배우의 모션 데이터를 새로운 모델에 재적용하는 모션 리타겟팅 기법[2][3]이다. 둘째는 배우의 동작을 가능한 한 많이 캡쳐하여 이 동작 속에 들어있는 자세들을 획득, 자세 데이터베이스를 만든 후, 애니메이터가 특정 자세들을 선택, 연결하여 새로운 동작을 생성하는 기법이다. 이 기법도 몸동작에 대한 것이 먼저 제시[3]되었는데, 아직 얼굴 표정에 대한 것은 제시된 것이 없다.

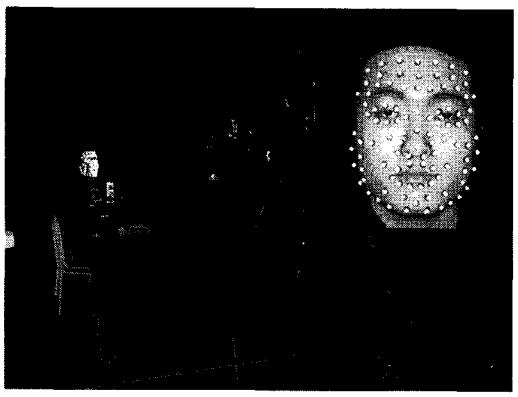


그림 1. 얼굴 모션 캡쳐 장면 및 배우의 얼굴에 부착한 마커 100개 부착 위치

그러므로 본 논문에서는 다량의 얼굴 모션 캡쳐 데이터를 직관적인 공간에 분포시키고, 애니메이터가 적당한 공간을 항해하면서 원하는 얼굴 표정들을 실시간으로 선택하여, 표정 애니메이션을 생성하거나 표정제어를 수행하기 위한 방법을 기술한다. 모션 데이터는 얼굴 표정을 전문적으로 연출하는 배우의 도움을 받아 [그림 1]과 같이 광학식 모션 캡쳐 시스템을 사용하여 얼굴 표정을 캡쳐한다.

표정을 캡쳐할 때, 배우는 [그림 1]과 같이 얼굴 주 근육 부분에 작은 반사 마커 100개를 부착한다. 그런 다음 배우로 하여금 수개의 얼굴 모션을 연출하게 하고, 초당 60 프레임으로 캡쳐한다. 본 논문에서 사용한 얼굴 모션 데이터는 배우의 즉각적인 연출에 의해서 캡쳐한 데이터이므로, [그림 2]와 같이 표정의 종류에 따라 프레임의 수가 상이하다.

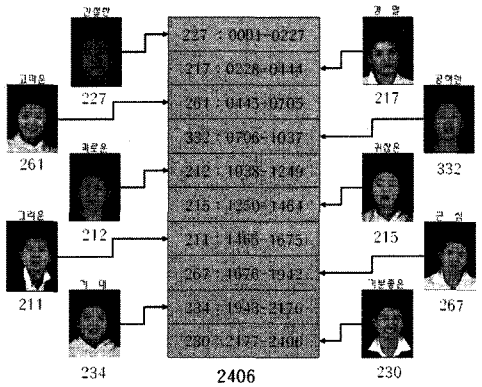


그림 2. 모션 캡쳐한 표정들 및 각 프레임 수 : 각 얼굴 표정 이미지에 나타난 숫자들은 각 표정별 프레임 수, 가운데 축적된 숫자들은 프레임 데이터를 합한 수

물론 모션 캡쳐를 할 때 표정마다 동일한 시간(3초 혹은 4초 등)으로 고정시켜놓고 캡쳐한 얼굴표정 데이터를 실험에 사용하는 것도 크게 문제 될 것은 없다. 그럴 경우에는 매우 유사하거나 거의 같은 얼굴표정에 속하는 프레임들의 수가 작아짐으로 특별히 밀집된 곳이 작아진다는 점은 있다. 그러나 본 실험을 위해 연출한 배우는 판토마임 전문가로서 모든 표정마다 감정을

충분히 표현하도록 요구를 하였으며, 배우 또한 그렇게 하기 위해 최대한 노력하였기 때문에 굳이 시간을 고정시킬 필요는 없었다. 본 논문에서는 모션 데이터로부터 실험에 불필요한 요소를 모두 제거한 3차원 위치 값으로만 구성된 10개의 모션 데이터들을 그림 2와 같이 순서대로 합체하여 2400여 프레임을 가진 하나의 연속된 모션 파일을 구성한다. 한 개의 마커는 3차원 좌표 값으로 표현되므로 100개의 마커위치로 표현되는 한, 표정은 300개의 측정치를 갖는다. 얼굴 표정 데이터는 3차원 공간상의 위치 값으로만 구성되어져 있기 때문에, 얼굴 표정들 사이의 유사성을 수치적으로 표현하기가 어렵다. 얼굴 표정들 사이의 유사성은 300개의 측정치를 2차원이나 3차원과 같은 저차원 평면상에 분포시킬 경우, 얼굴 표정들 사이의 거리 값으로 구분하기 때문에 필요하다. 그러므로 다양한 얼굴 표정들을 잘 구분할 수 있도록 하기 위해서 2장과 같이 마커들 사이의 거리를 이용한 벡터행렬로 변경하고, 3장과 같이 Locally Linear Embedding(이하 'LLE') 투영 기법[4,5]으로 2차원에 투영시킨다.

혹자는, 원래 캡쳐한 2400여개의 프레임들은 10가지의 대표적인 표정들을 나타내는 단계적인 프레임들의 총합이며 따라서 이들 표정이 모두 무표정에서 출발하여 각각의 표정들로 변하는 과정을 나타내는 frame sequence들이라면, 중앙에 무표정을 놓고 이를 출발점으로 하여 햇살처럼 퍼지는 10개의 방사선을 따라 각각의 표정에 해당되는 프레임들을 차례로 나열하여 애니메이터가 수동 조작할 수 있게 할 수 있지 않느냐고 반문할 수도 있다. 그렇게 되면 굳이 본 논문에서처럼 복잡한 LLE 알고리즘을 사용하지 않고도 2차원 평면에 표정상태들을 분포시키고 이를 이용한 표정제어가 가능하다고 판단할 수 있다. 그러나 그것은 본 논문의 목적과는 매우 거리가 멀며, 매우 빈약한 결과를 가져올 수 있다. 왜냐하면, 혹자가 말한 방식대로 하게 되면, 고차원 데이터를 2차원 공간에 분포시키는 복잡한 과정은 필요가 없을 것이지만, 실험에 사용한 10개의 모션 데이터에 한정된 표정 제어가 될 것이기 때문이다. 또한 아무리 애니메이터가 2차원 공간에 분포된 표정상태들을 연속적으로 잘 선택할 수 있다고 하여도 2차원 공간에 작은 점들로 분포된 표정상태 상호간의 유사도를 일일이 알 수 없기 때문에 매우 부자연스러운 표정의 변화를 보여줄 것이다. 특히 무표정상태를 보여줄 필요가 없이 다양한 표정의 변화를 연속적으로 보여줄 경우에는 더욱더 그러하다. 그러므로 본 논문에서는 고차원 데이터를 저차원 Embedding 벡터로 인접 데이터를 보전하면서 매핑하는 LLE 알고리즘을 사용한다.

실시간 표정 애니메이션을 생성하거나 표정제어를 수행하기 위해서는 4장과 같이 본 논문에서 개발한 사용자 인터페이스를 사용하여 애니메이터로 하여금 투영된 2차원 공간을 항해하게 하고, 항해경로의 각 점에 해당되는 모션 데이터를 3차원 얼굴 모델에게 실시간으로 적용하여 디스플레이 하도록 한다.

II. 얼굴 표정상태 표현법

표정공간을 생성하기 위해서는 각각의 얼굴 표정상태를 수치적으로 표현해야 한다. 표정상태는 얼굴에 부착된 마커들의 위치에 의해서 결정된다. 표정상태의 표현은 표정들 간의 상대적인 거리 관계를 잘 표현하는 것이어야 하고, 이로써 얼굴 표정을 구분할 수 있어야 한다. 표정의 상태를 표현하는 가장 간단한 방법은 마커들의 위치들로 이루어진 상태벡터를 이용하는 것이다. 본 논문에서 사용한 얼굴 모션 프레임 데이터는 100개의 마커를 사용하고, 한 마커는 3차원 좌표를 가지기 때문에, 표정상태벡터는 300개의 측정치가 된다. 이런 식으로 표현된 표정상태벡터를 '위치벡터'라고 하자.

본 논문에서는 표정상태를 표현할 때, 위치벡터를 사용하지 않는다. 대신 위치벡터의 임의의 두 마커간의 거리를 표현하는 '거리행렬'을 이용하여 표정상태를 표현한다. 왜냐하면 거리행렬이 위치벡터보다 두 표정이 서로 다르다는 것을 시각적으로 정확하게 비교할 수 있기 때문이다. 따라서 이 정보를 명시적으로 표현하는 거리행렬 방식이 더 좋은 표정상태 표현방식이다.

거리행렬은 각 프레임 데이터인 위치벡터를 기반으로 [그림 3]과 같이 구하는데, 하나의 위치벡터에서 100개 마커들 상호간의 거리는 100×100 행렬의 upper

또는 lower triangle로 표현할 수 있으며, 이를 편의상 일렬로 나열하여 4950개의 튜플을 생성한다. 본 논문에서는 거리행렬 방법으로 구해진 4950개의 튜플을 얼굴 표정상태로 표현하고 편의상 4950차원으로 구성된 표정상태벡터라고 한다.

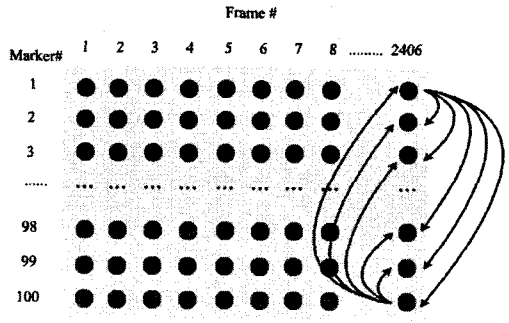


그림 3. 거리행렬 구하는 방법. 하나의 위치벡터에서 임의의 두 마커간의 거리를 구함으로써 한 표정 당 4950개의 튜플을 생성하고, 이를 하나의 얼굴 표정상태벡터로 사용

III. LLE(Locally Linear Embedding)

앞에서 생성한 거리행렬 방식의 표정상태 공간은 4950차원 공간이다. 따라서 이 고차원 공간을 애니메이터가 항해하면서 원하는 표정을 일일이 선택하는 것은 불가능하다. 그러므로 원래 고차원 표정공간의 구조를 근사적으로 표현하는 저차원 즉, 2차원 또는 3차원으로 축소된 공간을 구하여 이 공간을 항해하는 방법을 사용한다. 본 논문에서는 2차원 공간을 생성하며, 이를 위해 LLE 알고리즘[4][5][11]을 사용한다. LLE 알고리즘은 고차원 데이터를 저차원 Embedding 벡터로 인접 데이터를 보전하면서 매핑하는 것이 목적이다. 그리고 데이터는 이러한 특징 벡터들 상의 고차원 데이터들의 투영으로 나타난다.

본 장에서 기술하는 LLE 알고리즘 내용은 [11]을 참고하여 요약 정리한 것이다. 표정상태벡터를 나타내는 n개의 다차원 공간상의 점들을 $x_i, i=1,...,n$라 하고, 이를 집합 $\{x_i\}$로 표시한다. 그리고 다차원 표정공간을 근사적으로 나타내는 2차원 평면상의 점들을 $y_i, i=1,...,n$라 하고, 집합 $\{y_i\}$로 표시한다. 다차원 공간상의 점들의 집합 $\{x_i\}$에 LLE 알고리즘을 적용하면, 집합 $\{x_i\}$가 2차원 평면에 투영되면서 집합 $\{x_i\}$와 가장 근사한 분포를 가지도록 하는 2차원 평면상의 점들의 집합 $\{y_i\}$를 구할 수 있다. LLE 알고리즘은 [그림 4]와 같이 3 단계로 이루어진다.

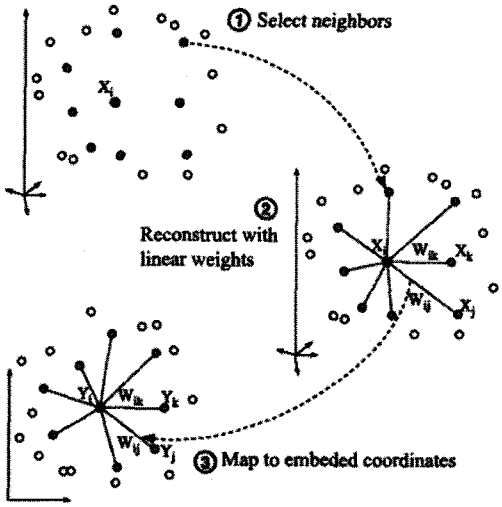


그림 4. LLE 알고리즘의 3단계 과정[5][11]

단계 1에서는 집합 $\{x_i\}$에서 각 표정상태벡터별로 가장 가까운 k개의 인접표정 상태벡터 $x_{i1}, ..., x_{ik}$을 찾는다. 각 표정상태벡터별로 인접표정 상태벡터를 찾는 방법은 x_i에서 x_j까지 직선거리를 계산하고, 각 표정상태벡터로부터 k개의 가장 작은 거리를 찾음으로써 구할 수 있다.

단계 2에서는 가장 가까운 인접표정 상태벡터로 x_i의 근사 값에서 나온 복원 오차 결과를 측정하고, 이 오차를 최소화하는 가중치 w을 복원한다. 복원 오차는 식 (1)과 같은 최적화 함수에 의해서 측정되어진다.

$$\epsilon(w) = \sum_i |x_i - \sum_j w_{ij} x_j|^2 \qquad (1)$$

단계 3에서는 가중치 w을 사용하여 가장 잘 복원되

어진 좌표 벡터를 계산하여 집합 $\{y_i\}$를 구한다. 이를 위해서는 먼저 최적화 행렬 $M=(I-w)'*(I-w)$을 생성하고, 이로부터 고유치와 고유벡터를 구한다. 행렬 M의 고유치를 오름차순 정렬하고, 0이 아닌 가장 작은 고유치에 대응되는 최저 $d+1$번째까지의 고유벡터를 찾음으로서 집합 $\{y_i\}$를 구한다.

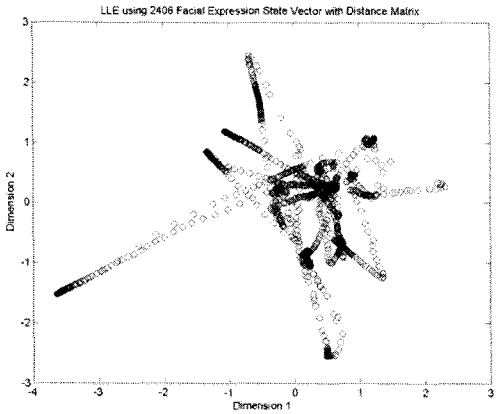


그림 5. 약 2400개의 표정상태벡터를 LLE 알고리즘에 적용하여 2차원 평면에 투영한 결과

[그림 5]는 위치벡터로부터 임의의 마커와 마커 사이의 거리로 구성된 표정상태벡터들의 집합을 LLE 알고리즘에 적용하여 2차원 평면에 투영한 결과이다. 본 논문에서는 2차원 평면상의 점들의 집합 $\{y_i\}$을 구하기 위해 Matlab V7.0을 사용하였다. Matlab 함수 lle[11]에 집합 $\{x_i\}$, 인접표정 상태벡터의 수 k 및 $d=2$을 입력으로 사용하여 2차원 평면상의 점들의 집합 $\{y_i\}$를 구하였다. [그림 5]의 가로축은 행렬 M으로부터 계산된 0이 아닌 가장 작은 고유치에 대응되는 2번째 고유벡터에 의해서 구해진 $y_{i,1}$의 값이다. 그리고 세로축은 최저 $d+1$번째 즉, $d=2$이므로 3번째의 고유치에 의해서 구해진 $y_{i,2}$의 값이다.

IV. 사용자 인터페이스 및 실험

본 논문에서는 약 2400개의 표정상태벡터로 구성된 표정공간을 생성한 후에 이를 LLE 알고리즘을 통해 2차원 공간으로 투영하고, 애니메이터로 하여금 이 공간을 항해하면서 얼굴애니메이션을 생성하거나 표정제어를 할 수 있게 하였다. 이를 위한 사용자 인터페이스는 [그림 6]과 같다.

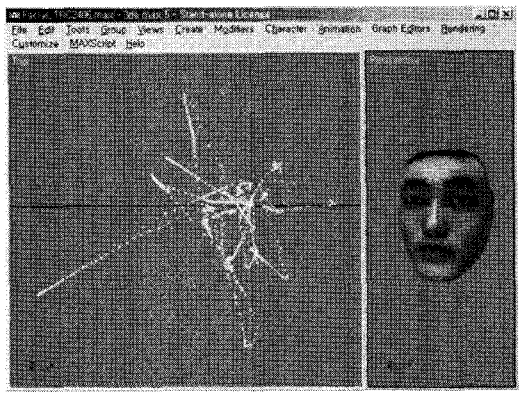


그림 6. 사용자 인터페이스 (좌: 항해 공간, 우: 3차원 얼굴 모델), 애니메이터가 2차원 항해공간에 분포된 각 얼굴 표정상태(위치벡터)를 대표하는 작은 점을 마우스로 선택하면서 항해를 하면, 선택된 점에 해당되는 표정이 3차원 얼굴 모델에 실시간으로 표시된다.

사용자 인터페이스의 왼쪽은 항해공간으로서, 실험에 사용된 약 2400개의 표정상태벡터들이 LLE 알고리즘에 의해서 분포된 결과이다. 항해공간의 작은 점 하나는 하나의 얼굴 표정상태인 위치벡터를 가지고 있으며 애니메이터가 마우스로 선택하게 되면 해당되는 표정상태가 사용자 인터페이스의 오른쪽에 있는 3차원 얼굴 모델에 실시간으로 적용되어 디스플레이하게 하였다. 이를 위해 본 논문에서는 미리 3차원 아바타의 얼굴 모델을 제작할 때, 배우의 얼굴에 부착된 마커 100개의 위치와 동일한 위치에 해당하는 3차원 아바타의 정점(Vertex)들을 마커 정보와 연결시켰으며, 거리에 비례하여 주변 정점들이 움직일 수 있도록 구축하였다.

애니메이터는 [그림 6]의 왼쪽에 있는 2차원 표정공간의 점들 중에서 3차원 아바타의 표정을 제어하기 위한 표정상태들을 마우스 좌측 버튼으로 연속적으로 선택하여 적색의 항해 경로를 생성한다. 이때 사용자 인

터페이스는 애니메이터가 선택하는 점마다 그에 해당되는 표정상태 즉, 위치벡터를 오른쪽 3차원 아바타에게 적용하여 애니메이터가 표정상태를 실시간으로 확인할 수 있게 해준다. 또한 어느 정도 선택이 되고나면 마우스 오른쪽 버튼을 클릭하여 중단시키며, 그 결과로는 하나의 연결된 파란색 선으로 항해 궤도가 표시된다. 애니메이터의 항해 과정이 끝나면 애니메이터의 항해 궤도에 해당되는 얼굴 표정상태들을 연속적으로 보고 얼굴 표정상태의 변화를 확인할 필요가 있다. 이때는 애니메이터의 항해 궤도를 처음부터 끝까지 자동으로 반복 항해하여 연속된 얼굴 표정상태의 변화를 3차원 얼굴 모델을 통해서 보여주게 된다. 또한 이 경우 사용자 인터페이스의 항해공간에는 항해 궤도를 따라 표정상태들을 반복적으로 보여줄 때 현재 위치를 보여주기 위한 지시자로서 적색의 십자를 보여주게 된다. 즉, 애니메이터는 마우스를 사용하여 항해 공간을 항해하고 동시에 3차원 얼굴 모델에 적용된 얼굴 표정을 보게 되는데, 이때에는 표정상태 하나하나를 항해 과정과 항해 속도에 따라 확인하면서 보게 된다. 물론 3차원 아바타의 실시간 표정제어의 목적에 따라 즉각적인 표정의 변화를 줄 수도 있지만, 다양한 표정상태들을 선택하여 항해 경로 및 항해 궤도를 생성하고 재생하여 연속적인 표정의 변화를 반복적으로 줄 수도 있다. 또한 항해 경로와 항해 궤도를 재생성하여 3차원 아바타의 표정을 제어할 수도 있다. 이와 같은 과정으로 애니메이터는 사용자 인터페이스를 사용하여 3차원 아바타의 표정을 실시간으로 제어할 수 있게 된다.

[그림 7]은 애니메이터가 모든 2차원 표정공간을 항해하면서 마우스로 선택한 경로들 상에 있는 대표적인 얼굴 표정상태들을 표시한 것으로서, 원래 얼굴모델은 [그림 6]에서와 같이 사용자 인터페이스의 오른쪽 창에 표시되지만, 설명의 편리를 위해 항해공간에 표시하였다. 대표적인 표정상태들 사이의 선형 분포들은 이들의 순차적인 중간 표정들로 구성되어져 있다. 또한 [그림 7]은 거리행렬 방식에 의해 표현된 표정상태를 사용한 결과로서 애니메이션을 생성하거나 표정제어를 수행하기에 매우 효율적으로 분포되어져 있었다. 거리행렬 방식에 의해 표현된 표정상태들을 사용한 실험에서는 최적의 인접표정 상태벡터의 수 k값을 결정해야 한다. 본 논문에서는 각 표정별 인접표정 상태벡터의 수를 정할 때, 다음과 같은 기준을 적용하였다. 인접표정 상태벡터의 수에 따라 4950차원 공간에서의 표정상태벡터들의 거리 분포와 2차원 공간에서의 표정상태벡터들의 거리 분포 사이의 상관도가 달라진다. 이때, 가능하면 두 분포 사이의 상관도가 높게 나오는 인접표정 상태벡터의 수를 사용하는 것이 좋다.

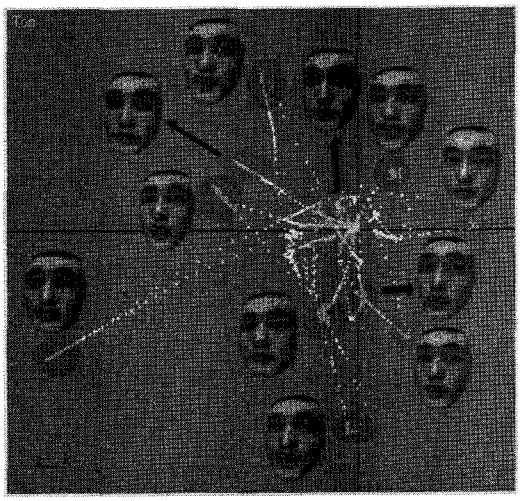


그림 7. 애니메이터가 LLE 알고리즘에 의한 2차원 표정공간을 항해하면서 마우스로 선택한 경로들 상에 있는 대표적인 얼굴 표정상태들을 설명의 편리를 위해 항해공간에 표시함

2차원 공간은 4950차원 공간을 근사적으로 표현하는 것이므로 두 공간 사이의 상관도가 높을수록 좋다. 이 기준을 적용하기 위해 4950차원 거리 분포와 2차원 거리 분포사이의 상관도는 피어슨의 상관계수(Pearson correlation coefficient), r[6]을 이용하여 구하였다.

4950차원 공간상의 임의의 표정상태벡터 x_i 와 x_j 사이의 거리를 하나의 벡터로 표현하고, 이를 V_d라고 하자. 2차원 평면에 투영된 임의의 점 y_i와 y_j 사이의 거리를 하나의 벡터로 표현하고, 이를 V_y라고 하자. 상관계수 r은 $-1 \leq r \leq 1$의 범위 값을 가지며, 벡터 V_d와 벡터 V_y을 이용하여 다음 식 (2)와 같이 계산되어진다.

$$r = \frac{\sum_{i=1}^{n}(v_{di} - \overline{V_d})(v_{yi} - \overline{V_y})}{\sqrt{\sum_{i=1}^{n}(v_{di} - \overline{V_d})^2 \cdot \sum_{i=1}^{n}(v_{yi} - \overline{V_y})^2}} \quad (2)$$

여기서 $v_{di} \in V_d$, $v_{yi} \in V_y$, $\overline{V_d}$는 V_d의 평균, $\overline{V_y}$는 V_y의 평균을 의미한다. 일반적으로 상관계수 r의 값이 $0.40 \leq r \leq 0.70$의 범위를 가지게 되면 비교적 높은 상관관계를 가진다고 한다. 본 논문에서는 가장 높은 상관관계를 가지는 인접표정 상태벡터의 수 k를 찾는다. 실험 결과 가장 높은 상관관계는 r=0.6335이며, 이때의 인접표정 상태벡터의 수는 37개이다. [그림 7]은 인접표정 상태벡터의 수 37개를 이용하여 구성한 2차원 거리 분포이다. 두 공간간의 상관도를 비교적 높게 만드는 인접표정 상태벡터의 수(35(r=0.6276), 39(r=0.6211), 40(r=0.6168) 등)를 사용했을 때, 임의의 두 표정 간의 경로가 존재하지 않는 경우는 없었다.

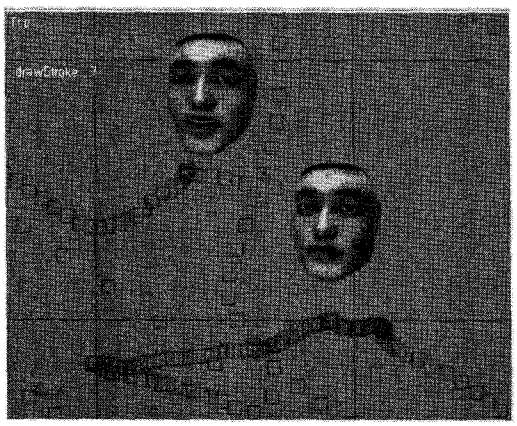


그림 8. 항해공간에서 임의의 두 표정간의 경로가 존재하지 않은 경우 : 인접한 표정상태의 상이함

그러나 두 공간간의 상관도를 낮게 만드는 상관계수(r=0.5000 이하)일 때의 인접표정 상태벡터들의 수를 사용할 때는 경로가 존재하지 않는 표정들이 많아진다. 이것은 인접표정 상태벡터들의 수를 너무 작거나 너무 많이 설정하면 임의의 두 표정상태벡터 사이에 인접표정 상태벡터들을 거쳐서 가는 경로를 구할 때, [그림 8]과 같이 경로가 존재하지 않는 경우가 많이 생기기 때문이다.

V. 결 론

본 논문에서는 다량의 고차원 얼굴 모션 데이터들을 직관적인 저차원 공간에 분포시키고, 애니메이터가 이 공간을 항해하면서 원하는 얼굴 표정 데이터들을 실시간으로 선택하고 디스플레이 하여 3차원 아바타의 표정을 제어하기 위한 방법을 기술하였다. 약 2400개의 얼굴 표정상태벡터를 사용하여 항해 공간을 구성하였으며, 이 공간을 구성하기 위해서는 위치벡터의 임의의 두 마커간의 거리를 표현하는 거리행렬들의 집합을 LLE 알고리즘에 적용하여 계산하였다. 실시간 얼굴 표정 애니메이션을 생성하거나 표정제어를 위해서 개발한 사용자 인터페이스는 표정상태 표현법을 위한 위치벡터 및 거리행렬 방식을 실험적으로 확인하는데 유용하게 사용되었다. 사용자 인터페이스는 애니메이터가 생성하고자 하는 얼굴 표정 애니메이션뿐만 아니라 3차원 아바타의 표정제어를 직관적인 공간을 자유자재로 항해하면서 실시간으로 생성, 확인, 수정 및 재생성이 가능하기 때문에 유용하고 효율적이라는 것을 애니메이터로 하여금 확인할 수 있었다.

참 고 문 헌

[1] F. I. Parke and K. Waters, *Computer facial animation*, A K Peters, 1996.

[2] C. Kouadio, P. Poulin, and P. Lachapelle, "Real-time facial animation based upon a bank of 3D facial expressions," Proc. Computer Animation '98, pp.128-136, 1998.

[3] J. H. Lee, J. Chai, P. Reitsma, J. Hodgins, and N. Pollard, "Interactive Control of Avatars Animated with Human Motion Data," ACM Transactions on Graphics (SIGGRAPH 2002),

Vol.21, No.3, pp.491-500, 2002.

[4] M. Polito and P. Perona, "Grouping and dimensionality reduction by locally linear embedding," In NIPS, 2001.

[5] S. Roweis and L. Saul, "Nonlinear dimensionality reduction by locally linear embedding," Science, Vol.290, No.5500, pp.2323-2326, 2000.

[6] Uprendra Shardanand, *Social information filtering for music recommendation*, Master's thesis, MIT, 1994.

[7] D. Vlasic, M. Brand, H. Pfister, and J. Popovic, "Face Transfer with Multilinear Models," ACM Transactions on Graphics(TOG), Vol.24, pp.426-433, 2005.

[8] Z. Deng, P. Y. Chiang, P. Fox, and U. Neumann, "Animating blendshape faces by cross-mapping motion capture data," Proceedings of the 2006 symposium on Interactive 3D graphics and games 2006, pp.43-48, 2006.

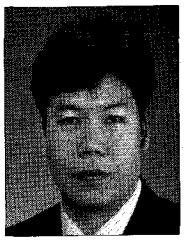
[9] D. Fidaleo and U. Neumann, "Analysis of co-articulation regions for performance- driven facial animation," Journal of Visualization and Computer Animation, Vol.15, pp.15-26, 2004.

[10] J. B. Tenenbaum, V. de Silva, and J. C. Langford, "A global geometric framework for nonlinear dimensionality reduction," Science, Vol.290, No.5500, pp.2319-2323, 2000.

[11] http://www.cs.toronto.edu/~roweis/lle/

저 자 소 개

김 성 호(Sung-Ho Kim) 정회원



- 1998년 8월 : 숭실대학교 컴퓨터학과 (공학석사)
- 2005년 2월 : 숭실대학교 컴퓨터학과 (공학박사)
- 2006년 3월 ~ 현재 : 상지대학교 컴퓨터정보공학부 교수

<관심분야> : 컴퓨터그래픽스, 컴퓨터애니메이션, 모션 캡쳐 애니메이션, 가상현실, Web3D, 멀티미디어