투영 텍스춰를 이용한 렌더링 후 3차원 와핑
(Post-Rendering 3D Warping using Projective Texture)

박 희 원 † 임 인 성 ††
(Huiwon Park) (Insung Ihm)

요 약 그래픽스 하드웨어가 급속도로 발전해왔음에도 불구하고, 복잡한 전경을 실시간으로 렌더링하는 것은 중요한 문제로 남아 있다. 최근 이미지 기반 렌더링에 대한 연구가 활발히 진행되면서 렌더링 후 3차원 와핑에 기반을 둔 렌더링 기법들이 제시되었다. 이러한 방법들은 좋은 결과를 산출하지만, 적지 않은 계산량을 요구하며, 또한 3차원 그래픽스 가속기 상에서 구현하기가 어렵기 때문에, 실시간 문맥에 사용하기가 용이하지가 않다. 본 논문에서는 이러한 문제를 해결하기 위하여 투영 텍스춰 기법에 기반을 둔 3차원 와핑을 통한 실시간 렌더링 기법을 제시한다. 이 방법에서는 복잡한 물체를 약간의 시간 차이를 두고 렌더링한 두 장의 참조 영상을 원래의 모델을 단순화시킨 물체에 각각 투영하여, 그 결과 영상을 혼합함으로써, 중간 시간에 대하여 원래의 물체를 렌더링한 것과 같은 우수한 화질의 영상을 빠르게 생성할 수 있다. 이 기법은 3차원 게임이나 가상 현실과 같은 실시간 그래픽스 응용 소프트웨어를 개발하는데 유용하게 쓰일 수 있을 것이다.

키워드 : 이미지 와핑, 영상 기반 렌더링, 투영 텍스춰, 실시간 렌더링

Abstract Due to the recent development of graphics hardware, real-time rendering of complex scenes is still a challenging task. As results of researches on image based rendering, the rendering schemes based on post-rendering 3D warping have been proposed. In general, these methods produce good rendering results. However, they are not appropriate for real-time rendering since it is not easy to accelerate the time-consuming algorithms within graphics subsystem. As an attempt to resolve this problem of the post-rendering 3D warping technique, we present a new real-time scheme based on projective texture. In our method, two reference images obtained by rendering complicated objects at two consecutive points of time are used. Rendering images of high quality for intermediate points of time are obtained by projecting the reference images onto a simplified object, and then blending the resulting images. Our technique will be effectively used in developing real-time graphics applications such as 3D games and virtual reality software and so on.

Key words : Image warping, image-based rendering, projective texture, real-time rendering

1. 서 론

최근 컴퓨터 그래픽스 하드웨어의 급속한 발전으로 인하여 보다 사실적인 영상을 실시간으로 생성할 수 있게 되었다. 하지만 이러한 하드웨어의 비약적인 발전에도 불구하고 사용자들의 욕구는 항상 하드웨어의 처리 능력을 넘어서고 있다. 따라서 제한된 성능을 가지는 그래픽스 하드웨어를 사용하여 방대한 그래픽스 데이타에 대하여 빠른 시간 내에 실시간 렌더링을 수행해야하는 문제는 아직도 중요한 연구 주제로 남아 있다. 이러한 문제를 극복하고자 하는 방법으로서 다양한 방식의 기법이 개발되어왔는데, 본 논문과 직접적으로 관련이 있는 기존 연구에 대하여 간략히 살펴보면 다음과 같다.

우선 첫째로, 복잡한 다면체 모델을 적은 개수의 다각형을 사용하는 물체로 단순화(mesh simplification)시키는 기법을 들 수가 있다[1]. 이러한 기법은 물체의 기하적인 성질과 위상적인 성질을 가능한 한 잘 보존하면서 물체를 표현하는데 사용되는 다각형의 수를 적절히 줄임으로써 렌더링을 하는데 필요한 시간을 단축시키는

· 본 연구는 서강대학교 산업기술연구소로부터 지원을 받았음.
† 비 회 원 : 서강대학교 컴퓨터학과
hwpark@simtech.co.kr
†† 종신회원 : 서강대학교 컴퓨터학과 교수
ihm@sogang.ac.kr
논문접수 : 2001년 10월 9일
심사완료 : 2002년 6월 18일

것을 목표로 한다. 특히 이 기법들은 보통 LOD(Level-Of-Detai) 기법과 연동되어 쓰이고는 한다[2,3]. LOD 기법은 물체를 단순화시킬 때 여러 단계로 단순화한 물체를 준비해 두었다가 상황에 따라서 적절한 해상도의 물체를 사용하는 것을 말한다. 그러나 이 방법은 물체가 가까이 있게되면 복잡한 물체를 그대로 사용해야 하므로 속도의 저하를 피할 수가 없게 된다. 만약 속도를 향상시키기 위해서 단순화된 물체를 그대로 사용할 경우, 세밀한 부분에 대한 표현은 잘 나타나지 않게 된다.

실시간 렌더링을 위한 또 다른 속도 향상 기법으로서 텍스춰를 이용한 영상 기반 렌더링 기법을 들 수가 있다. 최근 몇 년 사이에 영상 기반 렌더링에 대한 관심이 높아지면서 많은 연구가 진행되고 있다. 영상 기반 렌더링이란 기존의 기하 모델 대신에 사진이나 미리 렌더링을 한 영상을 이용하여 임의의 시점에서 보았을 때의 새로운 영상을 만드는 작업을 말한다[4]. 영상 기반 렌더링에 기반을 두어 복잡한 물체들을 빠른 시간 안에 렌더링을 하는 기법으로 '렌더링 후 3차원 와핑'(post-rendering 3D warping)이라는 기법을 들 수 있다[5]. 이 방법은 대체적으로 복잡한 물체를 직접 렌더링을 했을 때와 흡사한 영상을 얻을 수 있다. 하지만 3차원 그래픽스 가속기를 사용하지 않기 때문에 속도의 저하가 있을 수 있고, 2차원 영상을 이용하기 때문에 전경과 배경의 구분을 하는데 많은 시간을 낭비하게 된다.

본 논문은 다음과 같은 상황에서 영상 기반 렌더링 기법을 이용하여 다면체 모델에 대한 실시간 렌더링의 속도를 향상시킬 수 있는 새로운 기법을 제안한다. 정해진 경로를 따라, 또는 입력 장치에 반응하여 물체가 움직인다고 가정하자. 이때 매 프레임에 대하여 같은 방식으로 렌더링하는 것이 아니라, 예를 들어, 10 프레임 건너 한 번씩 원래의 다면체 모델을 사용하여 정확하게 렌더링을 하는데, 그 결과 생성되는 영상을 참조 영상(reference image)이라 하자. 참조 영상은 텍스춰의 형태로 저장이 되는데, 두 참조 영상의 중간에 해당하는 프레임에 대해서는 양쪽 끝의 참조 영상을 원래의 다면체 모델을 단순화시킨 물체에 대하여 투영을 시켜 합성을 하게 된다. 이러한 과정을 통해서 복잡한 데이타를 렌더링한 두 시점 사이에 존재하는 영상을 3차원적으로 보간을 해낼 수 있다. 이 기법은 모든 과정이 3차원 가속기를 이용하여 가속을 할 수 있기 때문에 효율적으로 구현을 할 수가 있다.

본 논문에서 제시하는 기법은 일정한 시간 차이를 가지는 임의의 두 렌더링 패러메터를 사용하여 렌더링한 참조 영상을 이용해서 중간의 렌더링 패러메터에 대한 영상을 빠르게 얻어내는 기법이다. 이 기법을 적용하기 위해서는 미리 정해진 경로를 따라서 물체가 움직이는 경우에나, 또는 짧은 시간 앞에 대한 예측이 가능한 경우에 유용하게 쓰일 수가 있다. 따라서 실시간으로 3차원 애니메이션을 제작할 때 각 프레임 사이의 영상을 보간 하거나, 가상 공간에서의 시각의 움직임, 3차원 게임에서 인공지능에 의한 움직임, 미끄러운 바닥 위에서의 무거운 물체의 움직임과 같이 짧은 시간 후에 대한 상황이 예측 가능한 움직임에 대한 렌더링에 대해서 효과적으로 사용을 할 수가 있다.

본 논문은 다음과 같이 구성된다. 2장에서는 이 논문의 기반이 되는 참조 영상의 3차원 와핑과 투영 텍스춰에 대해서 설명을 하고, 3장에서는 본 논문에서 제시하고자하는 투영 텍스춰를 이용한 참조 영상의 3차원 와핑에 대해서 알아본다. 4장에서는 제시된 기법의 실제 구현과 실험 결과를 제시하고, 마지막으로 5장에서는 결론을 맺도록 하겠다.

2. 기존 연구들

2.1. 영상의 3차원 와핑

OpenGL과 같은 렌더링 파이프라인을 이용할 경우, 렌더링의 결과물로써 깊이 정보를 얻을 수 있다. 결과 영상에서 픽셀의 위치와 이 깊이 버퍼의 정보를 결합하면 각 화소를 통하여 보이는 물체 표면에 대한 3차원적인 위치 정보를 얻을 수 있다. 보통 근접한 시점에서 렌더링이 된 두 개의 결과 영상은 비슷한 정보를 가지고 있기 때문에, 이 두 영상에서 얻은 픽셀의 3차원적인 위치 값과 색상 값은 두 영상 사이의 중간 영상들을 만들어 내는데 있어 유용한 정보를 제공한다. 이러한 방식의 렌더링 후 3차원 와핑 방법은 [5]를 비롯한 여러 기법들에서 사용되었다. [5]에서는 이러한 정보를 이용하여 두 영상을 각각 다면체 메쉬로 재구성한 후, 메쉬의 위치 좌표를 평면에서 평면으로의 3차원 와핑(planar-to-planar 3D image warping) 함으로써 새로운 시점에 대한 물체를 재구성할 수 있다 (<그림 1>). 이러한 물체들을 래스터화한 후 조합함으로써 얻고자 하는 새로운 유도 영상을 얻게 된다.

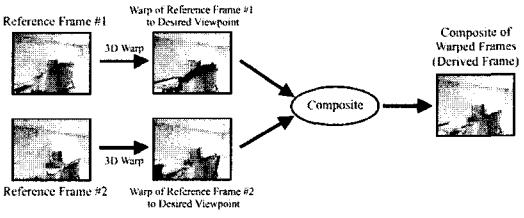

그림 1 렌더링 후 3차원 와핑의 과정[5]

일반적으로 영상을 화소 단위로 평면에서 평면으로의 3차원 와핑을 하게 될 경우, 화면에 빈 공간(hole)이 생기는 현상이 나타난다. 이러한 빈 공간을 막기 위해서는 화소들을 재구성해야 하는데 [5]에서는 영상 정보를 삼각형으로 이루어진 메쉬 정보로 재구성하여 하나의 연속된 물체 정보로 사용을 한다. 하지만 보통 전경과 배경이 구별이 되지 않기 때문에 렌더링이 된 영상은 위의 그림에서와 같이 원하지 않는 영상 정보가 추가된다. 이러한 정보를 제거하기 위해서 필요한 부분과 불필요한 부분을 판별해내기 위한 정보가 필요한데, 이를 위해서 연결성(connectedness)과 신뢰성(confidence)이라는 정보를 사용하여 적절한 합성 유도 이미지를 생성하였다.

[7]에서는 이러한 빈 공간을 채우기 위하여 몇 가지 방법을 제시하였으나, 기본적으로 없는 정보로부터 영상의 일부를 생성하는 것이기 때문에, <그림 2>에서와 같이 부자연스러운 부분이 생성되는 것을 근본적으로 막을 수가 없다.

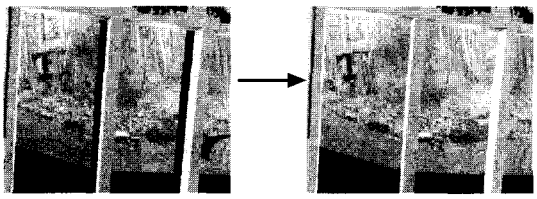

그림 2 빈 공간 채움의 결과[7]

2.2. 투영 텍스춰

투영 텍스춰(projective texture)란 <그림 3>에서와 같이 영사기가 화면이나 벽에 영상을 비추듯이 준비되어있는 텍스춰를 3차원 공간의 기하 물체 위에 투영시키는 기법으로서, OpenGL의 기본 기능으로 구현이 되어 왔다[9]. 실제로 이 방법에서는 기하 물체의 각 꼭지점에 대하여 텍스춰가 투영이 되는 지점에 대한 텍스춰 좌표를 계산하여 사용을 한다. 투영 텍스춰 기법은 OpenGL의 자동 텍스춰 좌표 생성 함수인 glTexGen() 함수와 텍스춰 행렬 스택(texture matrix stack)을 이용하여 쉽게 구현할 수 있다[14]. 투영 텍스춰 기능을 사용하려면 우선 물체의 위치 정보를 텍스춰 좌표로 변환하기 위하여 자동 텍스춰 좌표 생성 함수를 사용한다. 영사기를 기준으로 하는 눈좌표계에서의 물체의 꼭지점 좌표를 $(x_e\ y_e,\ z_e,\ w_e)$라고 할 때, $s = x_e$, $t = y_e$, $r = z_e$, $q = w_e$가 되도록 자동 텍스춰 좌표 생성 함수를 설정한다. 이러한 텍스춰 좌표에 대하여 기존의 OpenGL 파이프라인에서 모델뷰 행렬 스택과 투영 행렬 스택에서 수행했던 계산을 텍스춰 행렬 스택에서 행하게 된다.

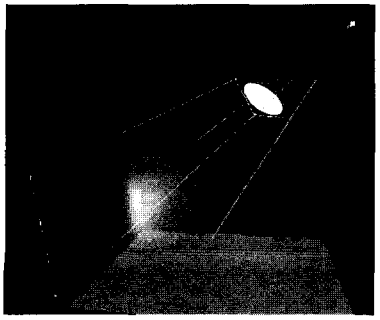

그림 3 투영 텍스춰의 예

이 텍스춰 행렬 스택을 적절히 설정하면 텍스춰를 기준으로 하는 좌표계로 좌표 변환이 이루어진다. 여기서의 투영 변환은 영사기를 기준으로 하는 눈좌표계에서 일종의 텍스춰에 대한 정규 디바이스 좌표계로 변환하는 과정에 해당한다. 정규 디바이스 좌표계에서는 (x, y, z)의 각각의 변수가 -1에서 1사이의 값을 가지기 때문에 0과 1 사이의 값을 가지는 텍스춰 좌표로 변환을 하기 위해 좌표 각각에 대해서 반만큼 축소한 후 0.5만큼 이동시키면 올바르게 텍스춰 이미지를 3차원 공간에 투영을 할 수가 있다.

투영 텍스춰 기법은 원래 그림자나 광원 효과를 생성하기 위하여 개발이 되었으나, [19]에서는 기존의 영상 기반 렌더링 방법인, [20]에서 제안된 뷰 종속 텍스춰 매핑(view dependent texture mapping) 방법의 효율을 높이기 위하여 투영 텍스춰 기법을 도입하기도 하였다. 이 방법은 투영 텍스춰 기법을 적용하는데 필요한 가시성 검사를 수행하기 위하여 입력 이미지의 각 부분을 다각형 패치로 분할을 할 수 있는 경우에만 적용을 할 수 있으며, 영상 기반 렌더링 방법의 특성상 빈 공간의 문제 등을 내포하고 있다.

3. 투영 텍스춰 기법을 이용한 참조 영상의 3차원 와핑

3.1 영상 기반 렌더링 기술을 이용한 영상의 3차원 와핑의 문제점

기존에 제시된 영상의 3차원 와핑 기술은 대부분 영상 기반 렌더링 분야에서 사용되고 있는 기법들에 의존하기 때문에 일반적으로 그래픽스 하드웨어로 가속하기가 수월하지 않다. 이러한 문제 외에도 전경과 배경의 구별을 위한 시간 소비, 2차원 영상을 기반으로 함으로써 발생하는 기존 다각형 데이타 렌더링 파이프라인과

결합의 어려움 등의 문제점이 있다. 따라서 영상 기반 렌더링 기법을 사용하는 3차원 와핑은 실시간이나 엄격한 시간 제한이 있는 분야에서의 사용을 목적으로 하는 것이 아니라, 보통 시간 제한이 엄격하지 않은 애니메이션 분야에 초점을 맞추고 있다.

3.2 영상 정보에서 다각형 정보로의 전환

본 논문에서 제시하는 기법은 3차원 그래픽스 가속기를 사용한 실시간 환경에 적용하는 것을 목표로 하기 때문에 기존 영상 기반 렌더링 기법을 이용한 3차원 와핑과는 달리 처리해야 하는 모든 과정들이 기존의 전통적인 다각형 데이타를 처리하는 가속기 상에서 효과적으로 수행되어야 한다. 이를 위해서는 우선 기존의 방식에서 다루는 데이타의 형태를 3차원 가속기에서 다루기 쉬운 형태로 변환해야 한다.

복잡한 물체를 렌더링한 결과로 생성되는 영상에서 얻을 수 있는 정보는 각 픽셀에 대하여 가로와 세로의 위치 (x, y), 깊이 정보 z, 그리고 색상 정보 (r, g, b, a)로 나눌 수 있다. 이러한 정보들 중에 가로-세로의 위치와 깊이 정보를 합치면 해당 픽셀을 통하여 보이는 물체 표면에 대한 3차원 기하 정보를 얻을 수가 있다. 이 정보를 사용할 경우 물체 표면에 대한 다각형 메쉬를 구성할 수가 있으나, 일반적으로 물체의 복잡도가 높아질 뿐만 아니라, 매 프레임마다 메쉬를 구성해야하는 문제가 발생한다. 본 논문에서는 이러한 방법대신 원래의 기하 물체를 단순화시켜 기하 정보를 표현하도록 하였다. 한편 렌더링 결과 생성되는 색상 정보는 텍스춰 형태로 표현하여 사용을 하였다. 본 논문에서 제안하는 기법에서는 이러한 텍스춰 데이타를 투영 텍스춰 기법을 사용하여 단순화된 물체에 3차원적으로 입히게 되는데, 물체를 충분히 단순화를 시켜도 원래의 텍스춰를 사용할 경우 단순화에 따른 부작용을 많이 줄일 수가 있다.

3.3 기본 알고리즘

기본적으로 본 논문의 기법은 기존의 영상 기반 렌더링 기법을 이용한 영상의 3차원 와핑 방법과 동일한 구조를 가진다. <그림 4>에서와 같이 이미 원래의 물체를 렌더링하여 생성한 참조 영상(참조 영상 1)과 시간적으로 미리 예측하여 생성한 참조 영상(참조 영상 2)을 조합해서 시간적으로 변화하는 이 두개의 영상 사이의 렌더링 패러메터에 대하여 새로운 유도 영상을 만들어낸다. 이러한 새로운 유도 영상을 생성하기 위해서 각 참조 영상을 사용하여 해당 렌더링 패러메터에 대하여 3차원 와핑을 한 참조 영상을 만들어낸 후(와핑된 참조 영상 1, 2), 이들을 적절히 조합을 해서 새로운 유도 영상을 생성한다. 이때 참조 영상으로부터 와핑된 참조 영상을 만들어 내기 위해서 단순화시킨 다면체 물체를 해당 렌더링 패러메터에 대한 위치로 옮긴 후 여기에 텍스춰 형태로 준비된 참조 영상을 투영한다. 이렇게 투영 텍스춰 기법을 사용하여 3차원 와핑을 수행한 결과 영상들을 조합하면 원하는 유도 영상을 생성할 수 있다.

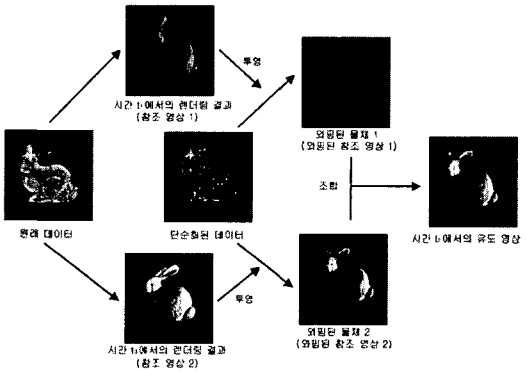

그림 4 본 논문에서 제시한 기법의 전체적인 흐름도

3.3.1 투영 텍스춰를 이용한 와핑된 참조 영상의 생성

본 논문의 방법을 효과적으로 구현하기 위해서는 미리 생성한 참조 영상과 단순화시킨 다면체 모델을 효율적으로 결합하여 3차원 와핑을 수행해야 한다. 이 과정에서 사용하는 텍스춰는 <그림 4>에서와 같이 미리 렌더링 한 참조 영상이다. 이 영상은 물체를 해당 관찰자 위치에서 바라본 모양을 영상화시킨 것이다. 따라서 반대로 참조 영상을 렌더링한 관찰자 시점에서 관찰자가 바라보는 방향으로 영상을 투영시킨다면 원래의 물체와 동일한(수치적인 오차를 제외한다면) 영상을 얻을 수 있다. 본 방법에서는 두 개의 참조 영상에 대한 시간 t_1과 t_3의 차이가 그리 크지 않다고 가정을 한다. 따라서 영상을 유도하려는 시간 t_2에 대한 렌더링 패러메터는 t_1와 t_3에서의 렌더링 패러메터와 크게 다르지 않기 때문에, t_2에서의 기하 물체가 투영된 모양은 양 끝의 참조 영상의 그것과 유사한 모양을 가진다고 가정할 수 있다. 이제 t_2에 대한 3차원 공간에서의 기하 물체에 양 참조 영상을 각각 투영을 하면, 두 개의 와핑된 참조 영상을 생성할 수가 있다.

3.3.2 법선 벡터를 이용한 다각형의 제거

본 논문에서 제안하는 방법에서 투영 텍스춰 기법을 적용하여 와핑된 영상을 생성할 때 주의를 해야한다. 참조 영상을 렌더링할 때 관찰자 시점에 대하여 뒷면이 보이는 다각형은 제거가 되기 때문에 결과 영상에 그러

한 다각형의 색상 정보는 표현이 되지 않는다. 한편 일반적으로 참조 영상과 유도 영상을 생성할 때 보이는 기하 물체의 모습이 다르기 때문에, 참조 영상을 생성할 때 제거되었던 다각형이 유도 영상을 생성할 때 제거되지 않을 수 있다. 이런 다각형에 대하여 참조 영상을 투영할 경우, 잘못된 영상을 생성할 수가 있다.

따라서 원래의 물체를 렌더링할 때의 렌더링 패러메터를 사용한 다각형의 제거가 필요하다. 참조 영상을 새로운 뷰잉 패러메터를 가지는 기하 물체에 투영할 때, 다각형의 법선 벡터와 참조 영상을 생성할 때 관찰자가 바라보았던 방향을 이용하여 불필요한 다각형을 제거할 수가 있다. 하지만 개개의 다각형에 대해서 제거 여부를 판단할 경우, 3차원 기하 데이타에 대한 실수 연산을 하드웨어적 가속기의 도움 없이 수행해야하기 때문에, 실시간으로 계산이 불가능하거나 처리 속도의 저하를 가져오게 된다.

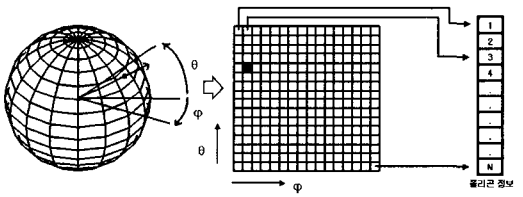

그림 5 대표 법선 벡터에 따른 다각형의 분류

본 논문에서는 계산량을 줄이기 위해서 다각형이 가지는 법선 벡터들을 일정한 수의 대표 법선 벡터로 대신하는 방법을 사용했다. 이를 위해서 <그림 5>에서와 같이 법선 벡터 (x, y, z)를 극좌표계 (r, θ, ϕ)로 변환한 후, 여기서 r을 제외한 두 변수를 사용하여 각각의 변수가 나타내는 값의 범위를 N개의 구역을 나누고 이 구역에 속하는 다각형들의 법선 벡터는 그 구역을 대표하는 대표 법선 벡터로 대신하도록 하였다. 이러한 대표 법선 벡터를 이용하여 해당 구역에 있는 모든 다각형에 대하여 다각형 제거를 하게 된다. 이런 제거 기법은 대표 법선 벡터에 대해서만 연산을 하기 때문에 관찰자가 바라보는 방향과 정확히 직각을 이루는 부분에서 어는 정도 오차가 발생을 할 수 있다. 이러한 부자연스러운 다각형의 제거를 최소화하기 위해서는 속도가 크게 저하하지 않는 범위에서 적당히 큰 N을 설정해 주어야 한다.

이를 구현하기 위해서 다각형들을 각각의 구역에 따라서 분류를 하고, NxN의 2차원 행렬에 각 구역에 해당하는 다각형 정보가 저장되어 있는 위치를 저장하도록 하였다.

3.3.3. 유도 영상의 생성

유도 영상을 생성하기 위하여 사용하는 참조 영상은 각각 해당 렌더링 패러메터에 대한 뒷면 제거 연산을 수행하여 생성한 영상이다. 따라서 그러한 참조 영상을 사용하여 새로운 렌더링 패러메터의 상황에서 3차원 물체에 투영을 할 경우 부자연스러운 영상을 생성할 수가 있다. 따라서 자연스러운 영상을 얻기 위해서는 두 참조 영상으로부터의 유도 영상이 가지고 있는 정보를 적절하게 조합할 필요가 있다. 조합을 목적으로 유도 영상의 지역을 분류하면, 하나의 와핑된 물체에 의해서만 픽셀의 값이 영향을 받는 부분과, 두개의 와핑된 물체에 의해서 픽셀의 값이 모두 영향을 받는 부분으로 나눌 수 있다. 이러한 분류는 유도 영상을 만드는데 사용하는 다각형 데이타에 그대로 적용할 수 있다. 다각형의 분류는 앞 절에서 언급한 방식의 법선 벡터 기법을 사용하여 효과적으로 수행할 수 있다.

이 분류에 따라서 하나의 와핑된 물체에 의해서만 영향을 받는 부분은 각각 투영 텍스춰 기법을 이용해서 각자의 참조 영상을 입혀서 렌더링을 한다. 한편 겹치는 부분의 경우에는 두 번의 렌더링을 거쳐 생성한다. 즉, 첫 번째 참조 영상을 이용해서 한번 렌더링을 한 후 그 다음에 두 번째 참조 영상을 이용해서 한번 더 렌더링을 하게 된다. 이 때 자연스러운 결과를 생성하기 위하여 두 개의 영상을 적절히 혼합해주어야 한다.

겹치는 영역에 대하여 두 개의 참조 영상은 약간씩 서로 다른 값을 가진다. 따라서 겹치는 부분을 0.5씩의 투명도를 이용해서 혼합을 할 수가 있으나, 이 경우 겹치는 부분과 겹치지 않는 부분의 경계가 눈에 거슬리게 나타나게 된다. 투영 텍스춰는 일정한 방향으로 텍스춰를 투영하는 기법이기 때문에, 각 참조 영상을 렌더링할 때의 관찰자가 바라보는 방향과 다각형의 법선 벡터가 평행하면 할수록 더 정확한 색상 정보를 가지고 있다고 할 수 있다. 따라서 두 개의 참조 영상에 대한 관찰자 방향과 3차원 물체의 각 다각형의 법선 벡터와의 관계에 따라 적절한 불투명도를 사용하면 부자연스러운 경계가 많이 없어지게 된다. 이를 위하여 첫 번째 참조 영상을 투영할 때 한 쪽 경계에서 다른 경계까지 선형적으로 불투명도를 줄이면서 혼합을 하고, 반대로 두 번째 참조 영상은 불투명도를 높이면서 혼합을 하였다. 이때 각 다각형마다 불투명도를 결정하면 계산량이 많아지기 때문에 앞 절에서 언급한 바와 같이 다각형을 법선 벡터에 따라 분류를 하여 구역에 따라 동일한 불투명도를 사용하였다.

3.4 문제점

본 논문에서 제시하는 기법은 투영 텍스춰를 이용하여 참조 영상을 단순화한 물체에 투영하기 때문에 결과 영상에 몇 가지 문제가 발생할 가능성이 있다. 첫째, <그

림 6>에서 점선으로 표시된 다각형과 같이 참조 영상을 만들 당시 물체의 뒷면에 해당하여 제거한 부분이, 유도 영상을 만들 때 보이게 되어 참조 영상의 정보가 붙게 되는 경우이다. 두 번째, 밝은 회색 선으로 표현한 다각형과 같이 참조 영상이 만들어지는 시점에서 다른 다각형이 겹쳐져 안 보이던, 앞면이 보이는 다각형이 새로운 시점에서 새롭게 보이게 되는 경우이다.

<그림 7>의 오른쪽 그림에서 토끼의 귀 부분에 나타난 경계는 이러한 문제 때문에 발생을 하였다. 첫 번째 문제는 앞에서도 설명한 바와 같이 물체의 대표 법선 벡터를 이용하여 그러한 다각형을 제거함으로써 해결하였지만, 두 번째 문제는 아직 향후 연구 문제로 남아 있다.

또 한가지 본 방법의 문제는 참조 영상을 생성하는 시간과 중간 영상을 생성하는 시간의 차이로 인하여 애니메이션 시 끊김 현상이 발생할 수 있다는 점이다. 하지만 이는 다음 참조 시점에 대한 정보를 미리 예측을 할 수 있다면(여러 번 강조한 바와 같이 본 논문의 기법은 어느 정도 예측을 할 수 있는 상황을 가정하여 개발하였음.), 다음 번 참조 영상에 대한 계산을 여러 부분으로 나누어 중간 프레임들에 대한 계산들 사이에 끼워 넣을 경우 그러한 문제를 감소시킬 수 있을 것이다. 이러한 계산은 렌더링을 위한 기존의 더블 버퍼 외에 P-버퍼와 같은 세 번째 오프 스크린 버퍼가 제공되는 가속기를 사용할 경우 어렵지 않게 구현할 수 있을 것이다[18].

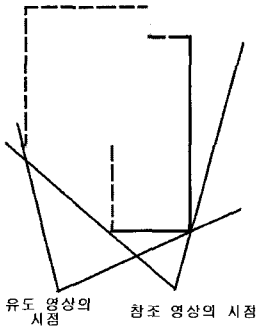

그림 6 투영 텍스춰 기법의 사용으로 인한 문제

그림 7 투영 텍스춰 기법의 앨리어싱 예

4. 실험 및 결과

본 논문에서 제안한 기법은 SGI Visual Workstation 320 상에서 구현되었다. 이 워크스테이션에는 450MHz의 Intel Pentium II CPU와 364 MByte의 메인 메모리, 그리고 SGI의 Cobalt 칩셋과 156 MBytes의 그래픽 메모리가 장착되어 있으며, 프로그램은 Visual C++ 6.0과 OpenGL 1.1의 환경에서 구현하였다. 실험을 위한 전처리 과정에서는 [21]에서 제공하는 VripPack 소프트웨어를 사용하여 다면체 모델을 단순화시켰다.

본 실험에서는 다양한 상황에서 원래의 다면체 모델을 매 프레임 마다 렌더링 할 때와 본 논문에서 제시한 방법을 사용하여 두 개의 참조 영상을 렌더링한 후 그것을 사용하여 중간 프레임을 렌더링 할 때의 렌더링 시간과 결과 영상의 화질로 성능을 비교하였다. 특히 이를 위하여 참조 영상 사이에 보간을 하는 프레임의 개수와 물체의 단순화 정도를 달리 하며 실험을 하였다. <표 1>에 네 개의 프레임 개수(10, 20, 30, 40)에 대한 실험 결과를 요약을 하였는데, 여기서 Original은 원래의 물체를 매번 렌더링할 때의 시간이고, 나머지 세 개의 비율은 다각형의 개수를 해당 비율만큼 단순화시킨 다면체 모델을 사용하여 본 논문의 방법을 사용할 때의 시간을 의미한다. 여기서 이 세 가지 비율은 각각 1. 렌더링 영상에 어느 정도 원래의 물체의 형상이 제대로 나타나기 시작할 때, 2. 결과 영상이 만족할 만큼 좋아지기 시작할 때, 그리고 3. 원래의 물체를 사용할 때와 구별이 힘들어지기 시작하는 시점의 비율을 의미한다(<그림 8, 9, 10> 참조).

표 1 보간하는 프레임 수에 따른 속도 변화(단위: ms/프레임, 괄호 안은 원래의 다면체 모델의 다각형 개수)

물체	단순화 비율(%)	보간 하는 프레임 개수			
		10 프레임	20 프레임	30 프레임	40 프레임
Bunny (69,451)	Original	80.00	80.10	80.07	80.05
	5%	21.50	17.35	16.00	15.35
	2%	21.20	17.35	16.03	15.35
	1%	21.20	17.35	16.00	15.35
Dragon (871,414)	Original	850.20	850.90	853.80	851.93
	1%	118.00	72.35	57.17	49.55
	0.4%	104.70	59.05	43.83	36.20
	0.09%	104.50	59.05	43.83	36.20
Buddha (1,085,634)	Original	1062.40	1063.90	1067.33	1065.13
	0.6%	140.10	83.45	64.57	55.05
	0.2%	127.30	70.40	51.37	41.88
	0.07%	126.70	70.15	51.13	41.70

10 프레임을 사용할 때의 결과를 <표 2>에 요약을 하였는데, 여기서 볼 수 있듯이 데이타의 크기가 상대적으로 적은 Bunny와 같은 물체는 최대 3.77배까지 속도가 향상된 반면에, Dragon과 Buddha와 같이 상대적으로 많은 다각형을 사용하는 물체의 경우는 각각 8.14과 8.39배 정도까지 속도가 향상되어, 많은 개수의 다각형을 사용하는 물체일수록 속도의 향상이 큼을 알 수 있다. 또한 와핑을 하는 프레임의 수에 따른 속도의 변화를 관찰해 보면, 당연히 그러하듯이 프레임의 수가 많아질수록 한 프레임을 렌더링하는데 걸리는 시간이 줄어줄게 됨을 알 수 있다. 이때 렌더링 시간은 대체적으로 와핑하는 프레임 수에 반비례하여 감소함을 알 수 있다. 이는 참조 영상을 생성하는데 필요한 시간을 더 많은 유도 영상에 분배를 하기 때문이라 할 수 있다.

표 2 단순화 비율에 따른 속도 향상 비율(10 프레임 사용)

물체	단순화 비율		
Bunny	5%	2%	1%
	3.72 (80.00ms/ 21.50ms)	3.77 (80.00ms/ 21.20ms)	3.77 (80.00ms/ 21.20ms)
Dragon	1%	0.4%	0.09%
	7.21 (850.20ms/ 118.00ms)	8.12 (850.20ms/ 104.70ms)	8.14 (850.20ms/ 104.50ms)
Buddha	0.6%	0.2%	0.07%
	7.58 (1062.40ms/ 140.10ms)	8.35 (1062.40ms/ 127.30ms)	8.39 (1062.40ms/ 126.70ms)

결과 영상의 화질은 <그림 8>, <그림 9>, 그리고 <그림 10>에서 비교하였다. 각 그림에서 첫 번째 그림은 원래의 물체를 그대로 렌더링 했을 때의 결과 영상이다. 다음 뒤의 세 쌍의 이미지는 각각 앞에서 설명한 바와 같은 비율의 다각형을 가지는 단순화된 데이타를 사용하여 렌더링 했을 때의 결과로서, 실제로 사용한 단순화된 물체와 3차원 와핑을 통하여 생성한 결과 영상을 보여주고 있다. 여기서 두 번째 쌍의 경우 자세히 보면 원래의 물체를 렌더링 한 것과 비교하면 약간의 차이를 발견할 수 있으나, 영상의 화질이나 렌더링 시간을 고려할 때 여러 실시간 응용에서 충분히 사용될 수 있을 정도의 화질을 보인다고 사려가 된다.

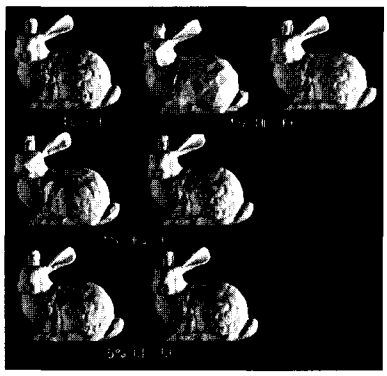

그림 8 Bunny 모델의 결과 영상

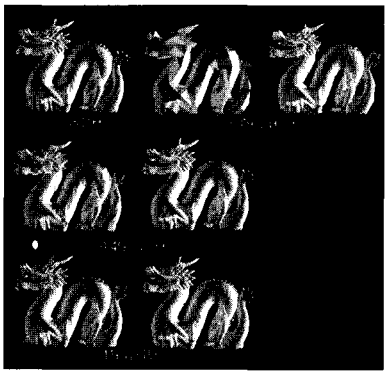

그림 9 Dragon 모델의 결과 영상

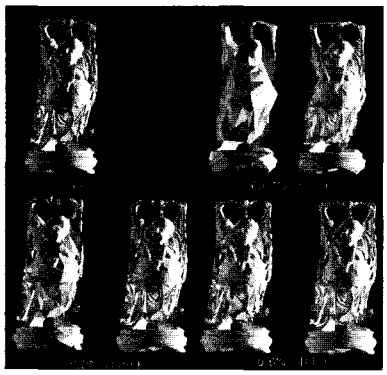

그림 10 Buddha 모델의 결과 영상

마지막으로 <표 3>에는 한 쌍의 참조 영상을 사용하여 동일한 물체를 여러 개 렌더링할 때의 시간이 요약되어 있다. 이 실험에서는 같은 참조 영상을 참조할 수 있도록 하기 위해서 물체들이 모두 비슷한 방향을 향하고 있다고 가정을 하였다. 이 경우에도 렌더링 시간은

표 3 와핑하는 물체의 개수에 따른 속도 변화 (단위: ms/프레임)

물체	단순화 비율(%)	렌더링하는 와핑된 물체 수			
		1	2	3	4
Bunny	Original	79.50	73.50	75.17	73.38
	5%	20.50	17.00	11.50	11.75
	2%	20.00	10.50	7.00	8.38
	1%	20.00	10.50	7.00	5.13
Dragon	Original	839.00	833.00	838.67	839.38
	1%	110.00	62.25	50.17	41.00
	0.4%	96.50	55.25	37.33	30.88
	0.09%	96.50	48.75	32.50	24.38
Buddha	Original	1052.00	1045.50	1047.83	1049.13
	0.6%	13.50	72.50	52.67	42.88
	0.2%	118.00	59.25	43.67	32.88
	0.07%	117.50	59.00	39.50	29.50

대체적으로 와핑하고자 하는 물체의 개수에 반비례하여 시간이 감소함을 알 수가 있다.

5. 결 론

본 논문에서는 투영 텍스춰 기법을 이용해서 참조 영상을 3차원 와핑을 함으로써 실시간으로 영상을 생성할 수 있는 기법을 제안하였다. 기존에 제안된 방법들은 주로 영상 기반 렌더링 기법에 기반을 둔 3차원 와핑을 통하여, 보간 영상을 생성하였다. 이러한 방법들은 그래픽스 하드웨어를 사용한 가속이 어렵고, 전경과 배경을 구별하는데 많은 시간을 소요하기 때문에, 비교적 좋은 화질의 결과를 얻는 반면 실시간 환경에서 사용하기가 어렵다는 단점이 있다.

본 논문의 방법에서는 기존 방법과는 달리 각각의 픽셀에 대해서 와핑 기법을 적용하는 대신에 단순화시킨 3차원 물체에 대하여 참조 영상을 텍스춰 형태로 투영하여 와핑 계산을 수행하였다. 즉 단순화된 물체에 원래의 물체를 랜더링한 영상을 투영하는데, 이때 이 물체에서 대표 법선 벡터를 사용하여 불필요한 부분을 제거한 후 렌더링함으로써 원래의 물체를 렌더링 했을 때와 비슷한 영상을 얻을 수 있었다. 특히 원래의 물체를 렌더링한 시점과 비슷한 시점에서 단순화시킨 물체를 렌더링할 경우 매우 흡사한 영상을 빠르게 얻을 수 있었다. 본 논문에서 제시한 방법은 그 전체 과정의 대부분을 OpenGL이나 DirectX와 같은 API를 사용하여 쉽게 구현할 수 있기 때문에, 3차원 그래픽스 가속기를 사용하여 가속하기가 용이하다. 실험 결과에서 볼 수 있듯이 원래의 3차원 모델을 렌더링하는 것에 비해서 상당히 짧은 시간 내에 원래의 데이타를 렌더링한 것과 유사한 영상을 생성할 수 있었다. 다만 전체 알고리즘 중 시점 방향과 법선 벡터의 방향에 따라 다각형을 제거하는 부분(3.3.2절 참고)은 최근 그래픽스 가속기들이 제공하는 꼭지점 쉐이더(vertex shader)와 같은 프로그램 가능한 쉐이더 기능을 사용하지 않을 경우, 소프트웨어적으로 구현을 해야하나 이 경우 N을 비교적 작은 수로 제한을 하기 때문에, 전체 과정에서 다각형 제거 계산을 위한 내적 계산이 차지하는 부분은 상당히 작다고 할 수 있다.

본 논문의 기법에서는 단순화시킨 물체가 원래의 데이타의 기하학적인 특성을 어느 정도만 유지하면 되므로, 매우 적은 수의 다각형을 사용하는 단순화 물체를 사용해도 비교적 좋은 결과를 얻어낼 수 있다. 따라서 본 방법은 3차원 게임, 가상 현실 등의 실시간 그래픽스 응용에서 시점이나 물체의 움직임의 변화를 어느 정도 예상할 수 있는 경우에 매우 빠른 속도로 영상을 생성할 수 있을 것이다.

참 고 문 헌

[1] G. Taubin et al., *3D Geometry Compression*, SIGGRAPH 1998 Course Note #21, 1998.

[2] P. Heckbert et al., *Multiresolution Surface Modeling*, SIGGRAPH 1997 Course Note #25, 1997.

[3] D. Luebke et al., *Advanced Issues in Level of Detail*, SIGGRAPH 2000 Course Note #41, 2000.

[4] P. Debevec et al., *Image-Based Modeling, Rendering, and Lighting*, SIGGRAPH 2000 Course Note #35, 2000.

[5] W. Mark, L. McMillan, and G. Bishop, "Post-Rendering 3D Warping," *Proceedings of 1997 Symposium on Interactive 3D Graphics*, pp. 7-16, April 1997.

[6] L. McMillian, and G. Bishop, "Head-Tracked Stereoscopic Display Using Image Warping," *Proceedings of SPIE*, Vol. 2409, pp. 21-30, 1995.

[7] W. Mark, *Post-Rendering 3D Image Warping: Visibility, Reconstruction, and Performance for Depth Image Warping*, technical report TR 99-022, Dept. of Computer Science, Univ. of North Carolina at Chapel Hill, April 1999.

[8] R. Azuma, *Predictive Tracking for Augmented Reality*, PhD Thesis, University of North Carolina at Chapel Hill, 1995.

[9] M. Segal, C. Korobkin, R. van Widenfelt, J. Foran, and P. Haeberli, "Fast Shadows and Lighting Effects Using Texture Mapping," *Computer Graphics (SIGGRAPH '92 Proceedings)*, pp. 249-252, 1992.

[10] P. Heckbert, *Fundamentals of Texture Mapping*

and Image Warping, Master's Thesis, University of California at Berkeley, June 1989.
[11] J. Dorsey, "Design and Simulation of Opera Lighting and Projection Effects," *Computer Graphics (SIGGRAPH '91 Proceedings)*, pp 41-50, 1991.
[12] L. Williams, "Casting Curved Shadow on Curved Surfaces," *Computer Graphics (SIGGRAPH '78 Proceedings)*, pp. 270-274, 1978.
[13] W. Reeves, D. Salesin, and R. Cook, "Rendering Antialiased Shadow with Depth Maps," *Computer Graphics (SIGGRAPH '87 Proceedings)*, pp. 283-291, 1987.
[14] M. Woo, J. Neider, T. Davis, and D. Shreiner, *OpenGL Programming Guide, 3rd Edition*, Addison-Wesley, Reading MA, 1999.
[15] R. Fosner, *OpenGL Programming for Windows 95 and Window NT*, Addison-Wesley, Reading MA, 1996.
[16] J. Foley, A. van Dam, S. Feiner, and J. Hughes, *Computer Graphics: Principles and Practice, 2nd Edition*, Addison-Wesley, Reading MA, 1996.
[17] A. Bourgoyne, and R. Bornstein, *Silicon Graphics Visual Workstation OpenGL Programming Guide*, https://www.sgi.com/developers/nt/sdk/files/OpenGLEXT.pdf, 1999.
[18] C. Wynn, "Using P-Buffers for Off-screen Rendering in OpenGL," NVIDIA white paper, 2001.
[19] P. Debevec, G. Borshukov, and Yizhou Yu, "Efficient View-Dependent Image-Based Rendering with Projective Texture-Mapping," In 9th Eurographics Rendering Workshop, pp. 105-116, Vienna, Austria, June 1998.
[20] P. Debevec, C. Taylor, and J. Malik, "Modeling and Rendering Architecture from Photographs: A Hybrid Geometry- and Image-Based Approach," *Computer Graphics (SIGGRAPH '96 Proceedings)*, pp. 11-20, 1996.
[21] VripPack: Volumetric Range Image Processing Package, http://www.graphics.stanford.edu/software/vrip, 2001.

박 희 원
1999년 2월 서강대학교 컴퓨터학과 졸업(공학사). 2001년 8월 서강대학교 컴퓨터학과 졸업(공학석사). 현재 심테크 시스템 부설 연구소 재직. 관심분야는 실시간 렌더링, 이미지 기반 렌더링 등임.

임 인 성
정보과학회논문지 : 시스템 및 이론
제 29 권 제 5·6 호 참조