POSIT을 이용한 3D 객체의 비디오 삽입

이범종[0] 김형목* 박종승 노성렬**
인천대학교 컴퓨터공학과, 경기대학교 정보과학부*, 시리우스**
{leeyanga, park, mysung}@incheon.ac.kr, noh@sirius.co.kr**

Inserting Virtual Object into Video using POSIT Algorithm

Bum-Jong Lee[0], Hyung-Mok Kim*, Jong-Seung Park, Sung-Ryul Noh**
Department of Computer Science and Engineering, University of Incheon, Information Science Division, Kyonggi University*, SIRIUS**

요 약

본 논문에서는 비디오 프레임에서의 특징점 추출과 3D 객체의 점들간의 상대적인 좌표만을 사용하여 비디오에 가상객체를 삽입하는 방법을 제안한다. 가상객체의 삽입위치는 사용자가 대응관계를 지정하여 결정한다. 3D 객체의 비디오 삽입을 위한 투사행렬은 POSIT 알고리즘을 사용하여 계산한다. 가상 객체를 삽입함에 있어 실객체가 가상객체를 가리는 현상을 해결하였다. 본 논문에서 제안한 방법은 특징점과 3D 객체의 점들 간의 상대적인 좌표만 준비된다면 각 비디오 프레임의 카메라 행렬을 계산할 수 있고 가상객체의 가려짐을 고려하여 합성할 수 있다.

1. 서론

본 논문에서는 비디오에서의 특징점과 가상객체를 삽입하고자 하는 부분의 로컬 3D 좌표를 알고 있다고 가정한다. POSIT 알고리즘을 사용하여 카메라 행렬을 계산함으로써 가상 객체를 비디오에 삽입하고 가려짐 현상을 고려하여 합성 하는 방법을 제안한다.

가상 객체를 비디오에 합성하는 몇 가지 방법을 소개한다. 비디오의 특징점과 합성할 객체의 텍스처 이미지와의 변환 관계를 구하여 이미지를 합성하는 방법[1]이 있다. 3D 객체의 기하정보를 이용하여 합성하는 방법에는 여러 장의 이미지로 깊이 좌표를 계산하여 위치 관계를 고려한 합성방법[2]이 있다. 3D 객체의 기하 정보를 바탕으로 3D 객체의 각 면의 이미지를 각 프레임의 해당 페이스 영역에 매핑하는 방법[3]도 제안되었다. 또한 3D 메쉬를 비디오 프레임내의 실 객체로 표현하여 비디오 내의 실 객체를 만드는 방법[4]도 제안되었다.

본 논문에서는 사용자 인터렉션을 통하여 사용자가 비디오의 프레임에서 가상 객체를 삽입할 위치를 지정한다. 사용자가 지정한 위치의 로컬 3D 좌표와 그에 대응되는 2D 좌표를 입력으로 하는 POSIT 알고리즘[5]을 사용하여 카메라 행렬을 구하는 방법에 대해 소개한다. 또한 실 객체의 2D 좌표만을 가지고 가상 객체의 가려짐을 고려하는 합성을 소개한다.

2. POSIT 알고리즘을 사용한 가상객체의 삽입

그림 1은 시스템의 프레임워크를 나타낸다. 전체 시스템은 여섯 단계로 나눌 수 있다. 사용자 인터렉션은 비디오 입력 프레임과 3D 객체간의 대응관계를 지정함으로써 입력 프레임에 가상 객체가 삽입될 위치를 지정한다. 이 과정은 첫 번째 프레임에서 한번만 수행하고 이 대응관계를 이용해 첫 번째 프레임에서부터 마지막 프레임까지 해당 위치에 가상 객체를 삽입한다. 그 다음 단계로 가상 객체의 로컬 3D 좌표와 그에 대응되는 비디오 프레임에서의 2D 좌표를 입력으로 사용한다. 이 정보로 POSIT 알고리즘을 사용하여 각 프레임에 대한 카메라 투사 행렬을 계산 할 수 있다. 투사 단계에서는 POSIT 알고리즘을 사용하여 구해진 투사 행렬과 가상객체의 기하 정보를 입력으로 투사된 가상객체를 얻어낸다. 가상 객체의 각 정점들의 2D 좌표를 계산하는 단계이다.

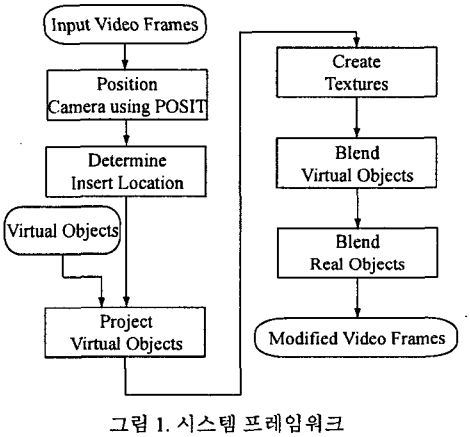

그림 1. 시스템 프레임워크

텍스처 생성 단계는 가상 객체가 실 객체에 의해 가려질 경우에는 실 객체의 텍스처가 자동으로 생성된다[6]. 실 객체의 각 면에 대한 텍스처를 생성함으로써 실 객체가 가상 객체를 가리는 부분의 처리에 사용한다. 가상 객체 합성단계에서 투사된 가상객체를 입력으로 가상 객체를 각 비디오 프레임에 합성한다. 출력으로 가상 객체가 합성된 프레임을 생성한다. 실 객체 합성단계에서는 실

[1] 본 연구는 산업자원부 (MOCIE)의 지원으로 인천정보산업진흥원 (IIT)을 통하여 수행되었음.

객체가 가상 객체를 가리는 면에 대해서만 수행한다. 생성된 실 객체의 텍스처를 사용하여 실 객체를 매핑 시킴으로써 실 객체가 가상객체를 가리는 문제를 해결한 최종의 합성된 비디오를 얻어낸다.

2.1 사용자 인터랙션

비디오 프레임에서 추적된 특징점 중 가상 객체를 삽입할 위치와 3D객체간의 관계를 지정한다. 가상 객체의 각 정점과 비디오 프레임의 추적된 2D 점을 지정한다. 이 단계에서 3D 모델은 여러 개를 정의한다. 가상 객체는 텍스처의 정보가 있는 모델과 대응시키고 실 객체는 텍스처의 정보가 없는 모델과 대응 시킨다. 이러한 3D 모델의 정의를 사용하여 실 객체의 합성과 가상 객체의 합성을 구분한다. 사용자 인터랙션은 첫 번째 프레임에 대해서만 행한다. 이 후 프레임에 대해서는 각 3D 모델들과 대응하는 특징점들의 추적[7]을 통해 카메라 행렬을 계산하고 합성한다.

2.2 POSIT을 사용한 투사행렬 계산과 적용

POS(Pose from Orthography) 알고리즘은 로컬 3D 좌표를 카메라 좌표로 변환하기 위한 알고리즘이다. 가상객체의 3D 좌표와 그 점에 해당하는 이미지상의 2D 좌표가 준비된다고 가정한다. POS 알고리즘은 가상객체의 4개 이상의 로컬 3D좌표와 그에 대응되는 특징점을 가정한다. 또한 점들이 한 평면상에 있지 않아야 투사행렬을 계산할 수 있다. POS는 선형 시스템을 풀어내어 3D 객체의 회전 행렬과 이동 백터를 구한다. POSIT(POS with ITerations) 알고리즘은 POS 알고리즘을 반복적으로 수행 하는 알고리즘이다. POS를 반복 수행 할수록 회전행렬과 이동백터는 정확한 값에 가까워지게 된다. POSIT 알고리즘은 지정한 반복횟수만큼 또는 지정한 오차만큼 반복하여 실제에 가까운 회전행렬과 이동백터를 구한다.

카메라 내부파라미터 행렬 K의 요소 중 초점거리는 POSIT 알고리즘에서 상수로 사용한다. 따라서 구해진 회전행렬과 이동 백터는 그 초점거리에 대해 구해지기 때문에 K행렬을 구할 수 있다. 투사행렬 P는 식 (1)에 의해 카메라 내부파라미터 행렬 K와 회전행렬 R, 이동백터 t를 사용하여 구한다.

$$P = K[R|t] \quad (1)$$

투사 단계에서는 가상 객체를 각 비디오 프레임에 합성하기 위해 가상 객체의 투사된 2D 좌표를 구한다. 입력으로 POSIT 알고리즘을 통해 구해진 각 프레임에서의 투사 행렬과 3D 객체의 로컬 3D 좌표가 주어진다. 각 비디오 프레임에서 2D 좌표 x_i는 투사 행렬 P와 가상객체의 3D좌표 X_i를 입력으로 식 (2)를 통해 구할 수 있다.

$$x_i = PX_i \quad (2)$$

2.3 가려짐을 고려한 실 객체와 가상객체의 삽입

가상 객체는 실 객체 뒤에 삽입이 될 수도 있다. 이 경우에는 실 객체에 대한 찌그러져 있는 이미지를 직사각형의 이미지로 변환하여 텍스처로 생성해야 한다. 실 객체에 대한 텍스처를 생성하기 위해서는 실 객체의 모서리들에 대한 특징점을 추적하는 선행작업이 필요하다. 식 (3)은 실 객체의 한 면에 대한 특징점 x_i와 생성될 텍스처 모서리의 4점 x'_i와 행렬 R의 관계를 나타내는 식이다. 행렬 R은 식 (3)을 SVD(Singular Value Decomposition)를 이용하여 풀면 구할 수 있다.

$$x'_i = Rx_i \quad (3)$$

3D 가상객체의 기하 데이터와 식 (1)에서 계산된 카메라 행렬을 이용하여 가상 객체가 삽입될 위치에 가상객체를 삽입한다. 가상객체 삽입은 가상객체와 실 객체의 위치관계를 고려하여 뒤쪽에 위치한 객체부터 차례대로 렌더링한다. 실객체 렌더링은 생성된 텍스처를 사용하여 실 객체의 각 면에 텍스처를 매핑하는 방식으로 렌더링한다. 이러한 렌더링으로 가상객체가 실객체에 의해 가려짐을 고려하여 합성할 수 있다.

그림 2는 비디오 프레임에서 텍스처를 생성해내는 예를 보여준다. 책의 표지에 해당하는 텍스처를 생성하기 위해서는 표지의 네 모서리 점에 대한 2D 좌표가 주어지면 식 행렬 R을 구할 수 있다. 행렬 R을 사용하여 오른쪽 그림과 같이 실 객체의 한 면에 대한 하나의 텍스처를 생성할 수 있다.

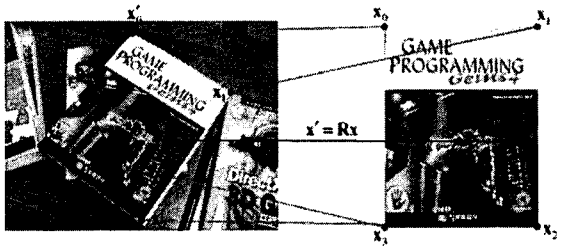

그림 2. 텍스처 생성을 위한 텍스처 좌표에 대한 관계

3. 실험결과

(a) (b) (c)

그림 3. 가상 객체의 합성: (a) 원본 프레임, (b) 가상객체의 와이어 프레임, (c) 가상객체가 삽입된 프레임.

그림 3(a)의 이미지들은 실험 프레임을 보여주고 있다. 각 프레임에서 달력과 티슈상자 사이에 가상 객체를 삽입한다. 그림 3(b)는 투사행렬을 사용하여 가상객체의 점들을 표시한 것이다. 파란색의 점과 선은 구해진 투사행렬로 3D 좌표들을 투사하여 정확도를 알아보기 위한 것이다. 그림 3(c)는 티슈상자의 가상객체가 삽

입된 모습을 보여준다. 가상객체 전면에 실 객체인 티슈상자가 있다. 실 객체 티슈상자의 텍스처를 생성하여 가상객체 삽입 후 실 객체에 다시 합성 함으로서 가려짐을 고려한 합성의 결과를 보여준다.

그림 4(왼쪽)은 그림 3(b)의 두 번째 이미지를 확대한 것이고, 그림 4(오른쪽)은 그림 3(c)의 두 번째 이미지를 확대한 것이다.

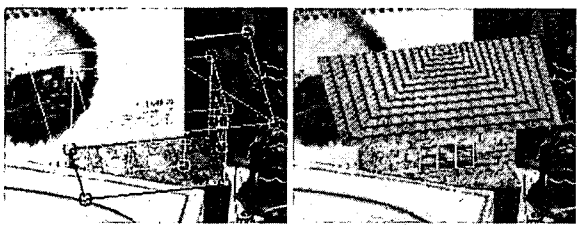

그림 4. 삽입된 가상객체의 모습

비디오에서 두 개의 3D 객체에 대한 투사 행렬을 구하는 실험을 하였다. 비디오는 x축에 대하여 회전하는 방향으로 촬영하였다. 또한 두 객체는 거의 모든 프레임에서 화면의 중간 정도에 위치해있다.

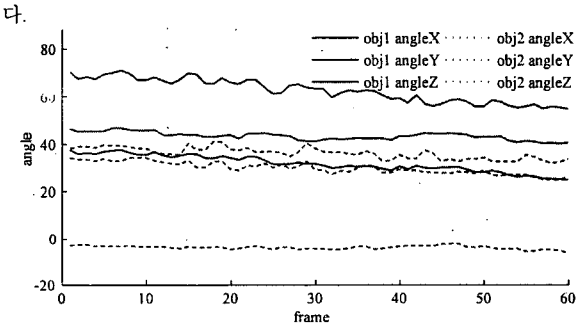

그림 5. 각 프레임에서의 카메라의 상대적인 두 객체의 회전

그림 5는 각 프레임 별 두 객체의 x, y, z축에 대한 회전각을 나타내는 그래프이다. 삽입된 가상객체의 전면이 정확하게 전면으로 보일 때를 기준으로 한다. 각 프레임에 대해 가상객체의 회전각을 비교해볼 때 회전각은 정확하게 변화되는 것을 볼 수 있다.

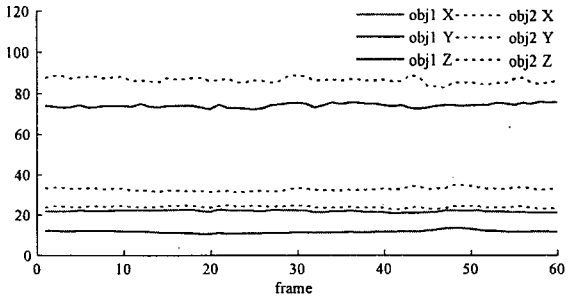

그림 6. 각 프레임에서의 두 객체의 상대적인 카메라 위치

그림 6는 각 프레임 별 두 3D 객체의 상대적인 카메라 위치를 표시한 데이터이다. 두 객체가 화면상에서 거의 고정된 위치에 있기 때문에 큰 변화는 없고 촬영 시 손의 떨림 정도에 의해 약간씩 변하는 것을 볼 수 있다.

POSIT 알고리즘으로부터 구해진 투사행렬의 정확도를 측정해 보았다. 측정은 각 프레임에서 추적된 2D 좌표 p_i와 각 프레임 별로 구해진 투사행렬을 통해 가상객체의 3D 좌표를 투사한 2D 좌표 p'_i를 사용했다.

$$e = \frac{1}{N}\sum_{i=0}^{N}|p_i - p'_i| \tag{4}$$

에러율 e는 POSIT 알고리즘에 사용된 점의 수와 p_i와 p'_i의 거리를 입력으로 하여 픽셀의 오차 값을 갖는다.

그림 7은 식 (4)와 같이 정의한 에러율에 대한 그래프를 보여준다. 프레임마다 에러율이 차이가 나는 이유는 보간 기법에 의해 구해진 특징점이 정확하게 구해지지 않고 오차가 있기 때문이다. 하지만 그 오차가 매우 작기 때문에 에러는 평균 2픽셀 미만의 정확도를 보여준다.

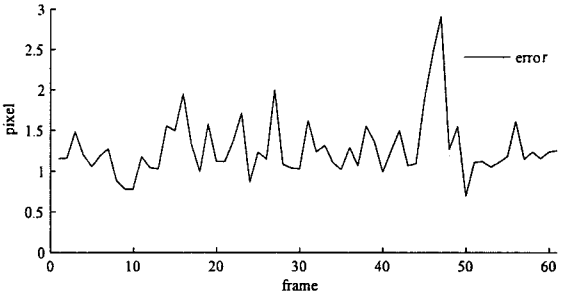

그림 7. 각 프레임별 실제 특징점과 투사된 점과의 에러율

4. 결론

본 논문에서는 POSIT 알고리즘을 사용한 가상객체를 삽입하는 방법을 제안하였다. 본 논문의 방법은 첫 번째 프레임에서만 사용자 인터렉션으로 사용자가 지정한 위치에 가상객체를 삽입한다. 또한 가상객체와 가상객체를 가리는 실객체와의 위치관계를 고려하여 실객체가 가상객체를 가리는 것을 고려하여 합성하였다.

POSIT 알고리즘은 프레임에서의 특징점 좌표를 입력으로 한다. 가려짐으로 인해 특징점 추적이 불가능한 경우 보간 기법을 사용하였다. 보간 기법으로 특징점을 추적하는 경우에 오차가 발생할 수 있다. 그러한 경우에 투사행렬에 오차가 생길 수 있다. 앞으로는 좀더 정확한 카메라 투사 행렬을 계산하기 위한 연구를 진행할 계획이다.

Reference

[1] Daniel G. Aliaga, Dimah Yanovsky, Thomas Funkhouser, Ingrid Carlbom, "Interactive Image-Based Rendering Using Feature Globalization," Proc. of ACM SIGGRAPH 1995, pp.39-46, 1995.

[2] Max N. and Ohsaki K., "Rendering Trees from Precomputed Z-Buffer Views," Rendering Techniques '95: Proceedings of the 6th Eurographics Workshop on Rendering, pp. 45-54, 1995.

[3] Matsushita K., Kaneko T., "Efficient and Handy Texture Mapping on 3D Surfaces," In Proc. of Eurographics, pp. 349-358, 1999.

[4] Rocchini, C., Cignoni, P. and Montani, C., "Multiple textures stitching and blending on 3D objects," Proc. 10th Eurographics Workshop on Rendering, pp. 127-138, 1999.

[5] Daniel F. DeMenthon and Larry S. Davis, "Model-Based Object Pose in 25 Lines of Code," IJCV, vol. 15, pp. 123-141, 1995.

[6] Jinhui Hu, Suya You, Ulrich Neumann, "Texture Painting from Video," The Journal of WSCG, Vol.13, pp. 119-125, 2005.

[7] Jean-Yves Bouguet, "Pyramidal Implementation of the Lucas Kanade Feature Tracker Description of the Algorithm," Intel Corporation, Microprocessor Research Labs, 1999.