마커리스 증강현실의 구현과 효율적인 레퍼런스 데이터 그룹의 생성 및 활용

구자명* · 조태훈*

*한국기술교육대학교

An Implementation of Markerless Augmented Reality and Creation and Application of Efficient Reference Data Sets

Ja-Myoung Koo* · Tai-Hoon Cho*

*Korea University of Technology and Education

E-mail : hackerme@kut.ac.kr, thcho@kut.ac.kr

요 약

본 논문은 마커리스 증강현실(Markerless Augmented Reality)의 구현과 레퍼런스(reference) 데이터 그룹을 효율적으로 생성하고 활용하는 방법을 제안한다. 구현은 카메라 설정과 레퍼런스 데이터 그룹 생성, 트래킹(tracking) 부분으로 되어 있다. 효율적인 레퍼런스 데이터 그룹을 생성하기 위해서는 CAD모델과 같은 3D모델을 필요하며, 다양한 관점에서 본 레퍼런스 데이터 그룹을 생성해야 한다. 모델에 대한 영상에서 특징점들을 추출하고, 광선 추적법을 이용하여 그 특징점에 대응하는 3D 좌표를 추출하여, 모델의 특징점 들에 대한 2D/3D 대응점의 레퍼런스 데이터 그룹이 구성된다. 트래킹 할 때 현재 프레임영상에서 특징점 들이 가장 많이 매칭되는 레퍼런스 데이터와 그 주위의 모델 데이터만을 이용하기 때문에 빠르게 트래킹 할 수 있다.

ABSTRACT

This paper presents how to implement Markerless Augmented Reality and how to create and apply reference data sets. There are three parts related with implementation: setting camera, creation of reference data set, and tracking. To create effective reference data sets, we need a 3D model such as CAD model. It is also required to create reference data sets from various viewpoints. We extract the feature points from the model image and then extract 3D positions corresponding to the feature points using ray tracking. These 2D/3D correspondence point sets constitute a reference data set of the model. Reference data sets are constructed for various viewpoints of the model. Fast tracking can be done using a reference data set the most frequently matched with feature points of the present frame and model data near the reference data set.

키워드

증강현실, 마커리스, Markerless, AR, Augmented Reality, Reference Data Set

Ⅰ. 서 론

증강현실이란 현실 세계에 추가적인 정보를 사용자에게 보여 줌으로서 증강된 현실을 만들어 낸다[1]. 증강현실에 대한 관심이과 학습, 시뮬레이션, 게임, 광고 등 적용사례가 많아지고 있다. 증강현실은 마커를 사용하는 경우[2]와 사용하지 않은 마커리스[3]로 분류되어진다. 마커리스 증강현실은 마커를 사용하는 경우보다 일반적이고 적용분야도 넓다. 하지만 일반적으로 마커리스 증강현실은 조명에 민감하고, 특징점을 추출하고 매칭하는 과정이 복잡하며, 시간이 오래 걸려 실시간

으로 처리하기가 힘들다.

본 논문은 마커리스 증강현실 구현에 대한 방법과 실제 모델의 자세가 변하지 않았음에도 떨리는 현상을 해결하기 위한 이전 프레임 영상과 현재 프레임 영상의 차 영상을 이용한 방법을 제시한다. 또한 실시간으로 트래킹 할 때 자세가 추정되면 다음 프레임에서는 추정된 자세와 근접한 레퍼런스 데이터만을 이용하여 빠르게 추적하는 방법을 제시한다.

II. 실제 카메라와 가상 카메라 모델

가상의 3D 오브젝트를 보여주기 위해서는 가상 카메라를 실제 카메라의 모델과 같게 해주어야 한다. 카메라 모델은 내부 카메라 파라미터 설정과 외부 카메라 파라미터 설정의 두 부분으로 이루어져 있다[4]. 먼저 실제 카메라의 내부 파라미터와 가상 카메라의 내부 파라미터는 다음과 같은 행렬로 표현된다.

$$\begin{bmatrix} f_x & 0 & u_0 \\ 0 & f_y & v_0 \\ 0 & 0 & 1 \end{bmatrix} \qquad \begin{bmatrix} \frac{2n}{r-l} & 0 & \frac{r+l}{r-l} & 0 \\ 0 & \frac{2n}{t-b} & \frac{t+b}{t-b} & 0 \\ 0 & 0 & \frac{-(f+n)}{f-n} & \frac{-2fn}{f-n} \\ 0 & 0 & -1 & 0 \end{bmatrix}$$

왼쪽이 실제 카메라, 오른쪽은 OpenGL 카메라에 대한 행렬이다. 실제 카메라 파라미터를 이용하여 OpenGL 카메라 파라미터를 적용한 행렬은 아래와 같다[5].

$$\begin{bmatrix} \frac{2f_x}{width} & 0 & \frac{2u_0}{width}-1 & 0 \\ 0 & \frac{2f_y}{height} & \frac{2v_0}{height}-1 & 0 \\ 0 & 0 & \frac{-(Far+Near)}{Far-Near} & \frac{-2FarNear}{Far-Near} \\ 0 & 0 & -1 & 0 \end{bmatrix}$$

width, height는 영상의 해상도이며, Near, Far는 OpenGL의 뷰볼륨에서 각각 앞평면, 뒷평면의 값이다. 일반적으로 Near는 0.01, Far는 1000로 설정한다. 이 부분은 처음에 한번만 설정을 해주면 된다.

다음은 실제 카메라의 외부 카메라 파라미터를 OpenGL의 외부 카메라 파라미터로 설정하는 부분이다. OpenGL에서 외부 카메라 파라미터는 모델 좌표계로 표현된다. 또한 실제 카메라의 외부 파라미터와 OpenGL의 모델 좌표계는 아래처럼 같은 동차좌표계(Homogeneous Coordinates)로 표현된다. 단 OpenGL에서는 오른손 좌표계를 사용한다는 것에 주의해야한다.

$$\begin{bmatrix} r_1 & r_2 & r_3 & t_x \\ r_4 & r_5 & r_6 & t_y \\ r_7 & r_8 & r_9 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

이 부분은 자세가 바뀔 때마다 설정해야 한다. 즉 자세를 나타내는 행렬은 외부 카메라 파라미터 또는 OpenGL의 좌표계의 행렬과 동일하다.

III. 레퍼런스 데이터 그룹 생성

레퍼런스 데이터 그룹을 생성하기 위해서는 실제 모델을 모델링한 3D모델이 필요하다. 레퍼런스 데이터 그룹 생성은 다음과 같은 순서로 진행된다.

1. 실제 모델을 카메라로부터 영상 획득.
2. 동일평면상이 아닌 최소 네 점의 2D좌표와 그에 대응하는 3D좌표를 각각 입력.
3. 2에서 입력된 좌표를 이용하여 POSIT[6]으로 3D모델의 자세를 계산.
4. 영상에서 특징점들을 추출
5. 특징점들에 대응하는 3D좌표를 얻기 위해 가상 카메라를 지나고 특징점들 좌표 방향으로 가는 광선과 3D모델과 교차점들을 구함.
6. 특징점들의 좌표와 5에서 구한 3D좌표들로 자세를 구하고 그 자세로부터 5번과 같은 방법으로 3D좌표를 다시 구함.
7. 6에서 구한 3D좌표들 중 RANSAC[7]을 통해 강건한 3D좌표들만 추출
8. 7에서 구한 3D좌표들과 그에 대응되는 특징점들의 데이터를 레퍼런스 데이터에 추가.
9. 실제 모델을 한쪽 방향으로 회전하면서 1번부터 반복

그림 1은 입력 영상과 3번의 결과 화면이다.

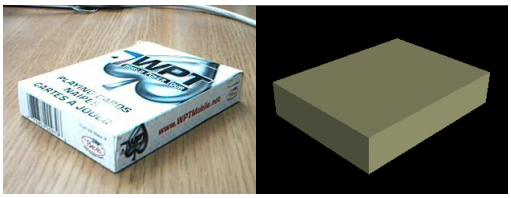

그림 1. 입력영상과 3D모델의 자세 설정

4번에서 영상의 특징점들을 추출하기 위해서 SURF[8]를 사용하였다. 조명에 강건한 특징점들을 추출하기 위해서 같은 자세에서 연속해서 얻은 여러 장의 영상을 사용하여 특징점들을 추출한 후 모든 영상에서 추출되어진 특징점들만 사용한다. 이러한 특징점을 사용한 결과 조명에 의해서 떨리는 현상을 감소시킬 수 있었다. 그림 2는 4번의 결과를 보여준다.

5번에서 3D좌표를 구하기 위해 광선과 3D모델의 교차점의 계산은 OpenSG[9]의 IntersectAction의 클래스를 이용하여 구현하였다.

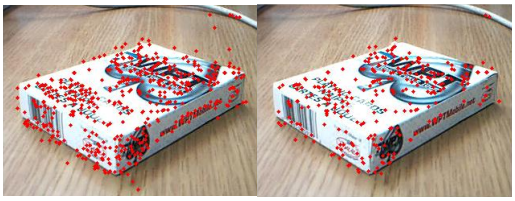

그림 2. 왼쪽 영상: 한 영상으로부터 추출한 특징점 표시, 오른쪽 영상: 연속해서 얻은 10개의 영상 모두에서 추출된 특징점만을 표시

8번에서 특징점들과 그에 대응하는 3D좌표들을 레퍼런스 데이터 그룹에 추가 할 때, 트래킹시 매칭되는 특징점들을 빠르게 찾기 위해서 다음과 같은 구조로 n개의 레퍼런스 데이터를 생성한다.

1	⇄	2	⇄	3	⇄	...	⇄	n	⇄	1

만약 오른쪽 방향으로 회전을 하면서 레퍼런스 데이터를 추가했으면 2번째 레퍼런스 데이터에서 오른쪽으로 회전한 경우의 레퍼런스 데이터가 3번째 이미지의 자세가 되고 왼쪽으로 회전한 경우의 레퍼런스 데이터가 1번째 이미지의 자세가 된다. 즉 가장 인접한 자세의 레퍼런스 데이터가 왼쪽 또는 오른쪽에 존재하게 된다. 아래의 그림 3은 이러한 레퍼런스 데이터의 예를 나타낸다.

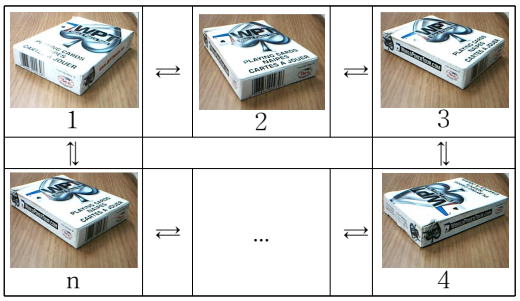

그림 3. 레퍼런스 데이터 구조의 예

Ⅳ. 트래킹

트래킹 단계는 현재 프레임에서 레퍼런스 데이터와의 매칭되는 특징점을 찾아 현재 자세를 추정하게 된다. 트래킹 순서는 다음과 같은 순서로 진행된다.

1. 현재 프레임에서 특징점들을 추출한 후 모든 레퍼런스와 비교하여 매칭되는 특징점이 가장 많은 래퍼런스를 i_m으로 설정하고 5번으로 진행.
2. 현재 프레임의 특징점을 추출한다.
3. 현재 프레임의 특징점들과 레퍼런스 i_m번째, i_m-1번째, i_m+1번째와 매칭되는 특징점 추출.
4. a) 충분한 특징점들이 매칭되지 않을 경우 :
   - j = 2로 초기화.
   - 레퍼런스 i_m+j번째, i_m-j번째와 매칭되는 특징점을 추출.
   - 충분한 특징점들이 매칭 되지 않으면, j를 1증가시키고 다시 위의 과정을 반복
   - 충분한 특징점들이 매칭되면 , i_m을 i_m+j로 지정하고 5번으로 진행.
   - 모든 레퍼런스 데이터와 비교해도 충분한 특징점이 매칭되지 않으면 자세 추정 실패

   b) 매칭되는 특징점이 i_m번째에서 가장 많은 경우 : 5번으로 진행.

   c) 매칭되는 특징점이 i_m+1 또는 i_m-1이 많은 경우 : i_m에 i_m+1 또는 i_m-1를 지정하고 5번으로 진행.
5. i_m번째 레퍼런스 데이터와 매칭된 특징점들중 RANSAC을 통해 강건한 특징점들만 추출.
6. 현재 프레임의 영상과 이전 프레임의 영상과의 차영상을 구한 후, 차 영상의 픽셀값의 평균값을 계산.
7. a) 평균값이 경계값보다 크거나 이전 영상에서 매칭된 특징점들보다 많은 경우:
   4번에서 얻은 특징점들로 POSIT으로 자세를 계산.

   b) 평균값이 경계값 보다 작고, 이전 영상에서 매칭된 특징점들보다 작은 경우:
   이전영상에서 구한 자세를 사용.
8. 7번에서 구한 자세로 OpenGL의 모델좌표계를 설정하고 2번부터 반복.

레퍼런스 데이터가 많아지면 많아질수록 급격히 트래킹하는 연산량이 증가하게 되어 속도 저하가 심하게 된다. 레퍼런스 데이터가 증가하더라도 빠르게 트래킹을 하기 위해서 4번 과정에서 레퍼런스 데이터를 효율적으로 활용하여 속도 저하를 해결 한다. 일반적으로 실제 모델의 자세가 갑자기 바뀌는 경우보다는 이전의 자세에서 점진적으로 바뀌게 된다. 따라서 이전 자세에서 가까운 레퍼런스 데이터부터 비교를 하기 때문에 많은 수의 레퍼런스 데이터가 있어도 빠르게 자세를 구할 수 있다. 또한 갑자기 변하는 경우에도 순차적으로 레퍼런스 데이터를 활용할 때보다 더 빠르게 자세를 구할 수 있다.

증강현실에서 떨리는 현상은 일반적으로 발생하는 문제다. 떨리는 현상은 미세한 조명의 변화로 특징점들이 다르게 추출되기 때문에 자세가 미세하게 계속 바뀌게 된다. 이러한 문제는 6, 7번에서 현재 영상과 이전의 영상의 차영상을 이용해서 해결하였다. 차영상의 픽셀 평균값이 경계값보다 작을 때 이전 영상의 자세와 같은 경우이다. 단 이때 이전 영상에서 구한 특징점보다 현재 영상에서 구한 특징점이 더 많으면 더 정확한 자

세를 구할 수 있으므로 현재 영상에서 구한 특징점들로 자세를 다시 구한다. 차영상의 픽셀 평균값이 경계값보다 클 경우는 실제 모델의 자세가 바뀐 경우이므로 다시 자세를 구한다. 경계값은 2.0~3.0사이의 값을 사용한다. 이러한 방법으로 실제 모델의 자세가 바뀌지 않았을 경우 떨리는 현상의 문제를 해결했다.

Ⅴ. 결론 및 연구 방향

위의 방법으로 마커리스 증강현실을 320x240해상도의 USB웹캠에 적용하였다. 레퍼런스 데이터에 사용한 이미지는 21개를 사용하였다. 현재 영상과 이전 영상과의 차영상을 이용한 결과, 실제 모델의 자세가 바뀌지 않았을 경우 미세한 조명 변화로 떨리는 현상을 해결할 수 있었고, 3D 오브젝트가 안정적으로 보였다. 속도는 초당 13프레임정도로 나왔다. 그림 4-1과 4-2는 결과 영상이다. 카메라의 왜곡에 의해서 약간의 오차가 발생하였다.

향후 연구 방향으로 카메라의 왜곡을 보정해서 오차를 줄이고, SURF가 SIFT[10]보다 빠른 대신 조명변화와 관점(viewpoint)변화에 더 민감하다. SURF와 SIFT 또는 다른 특징점 추출 알고리즘을 절충하여 정확하고 안정적이게 할 계획이다. 또한 트래킹의 속도를 증가하기 위해 이미지 전체에서 특징점을 추출하는 것보다 실제 모델의 이미지 영역만을 특징점 추출하게 할 계획이다. 많은 수의 레퍼런스 데이터를 사용하면 정확도는 증가하지만 레퍼런스 데이터를 생성하는 데이터를 직접 좌표를 입력해야 하므로 불편한 점이 있다. 적은 수의 레퍼런스를 사용할 수 있도록 실시간으로 백프로젝션(back-projection)을 통해 3D좌표를 계산하여 레퍼런스 데이터를 추가하게 되면 누적 오차가 발생 하게 되는데 이를 해결하여 실시간으로 레퍼런스 데이터를 추가할 계획이다.

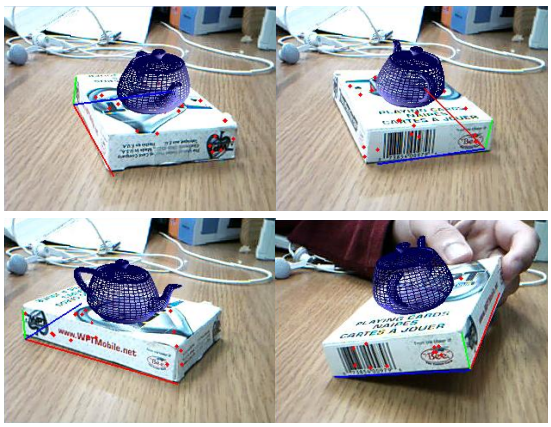

그림 4-1. 구현 결과 영상

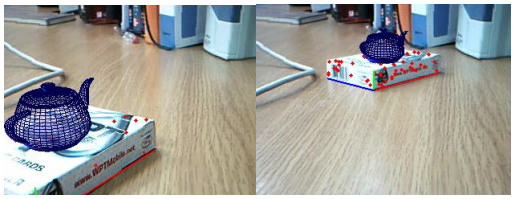

그림 4-2. 구현 결과 영상

참고문헌

[1] Azuma, R. A Survey of Augmented Reality, In Computer Graphics SIGGRAPH Proc., pp. 1-38, 1995.

[2] H.Kato and M.Billinghurst, Marker Tracking and HMD Calibration for a Video-Based Augmented Reality Conferencing System, iwar, pp.85, 2nd IEEE and ACM International Workshop on Augmented Reality, 1999.

[3] Bleser, G., Pastarmov Y., Stricker, D., Real-time 3d camera tracking for industrial augmented reality applications, Proc. WSCG, pp. 47-53, Plzen, 2005.

[4] David A., Forsyth , Jean Ponce, Computer Vision : A modern approach , Prentice Hall, International Edition, pp. 20-50, 2003.

[5] Ming Li, Correspondence Analysis Between The Image Formation Pipelines of Graphics and Vision, Proceedings of the IX Spanish Symposium on Pattern Recognition and Image Analysis, pp.187-192, 2001.

[6] DeMenthon, D. and Davis, L.S. Model-based object pose in 25 lines of code, In European Conference on Computer Vision, pp. 335-343, 1992.

[7] Hartley, R. and Zisserman, A.. Mutiple View Geometry in Computer Vision, Cambridge University Press, 2000.

[8] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool, SURF: Speeded Up Robust Features, Computer Vision and Image Understanding (CVIU), Vol. 110, No. 3, pp. 346-359, 2008.

[9] OpenSG, http://www.opensg.org.

[10] Lowe, David G., Object recognition from local scale-invariant features, Proceedings of the International Conference on Computer Vision, vol.2, pp. 1150 - 1157, 1999.