論文99-36S-7-11

가상스튜디오 구현을 위한 실시간 카메라 추적

(Real-Time Camera Tracking for Virtual Studio)

朴 星 禹 *, 徐 龍 德 *, 洪 起 祥 *

(Seong-Woo Park, Yong-Duek Seo, and Ki-Sang Hong)

요 약

가상스튜디오의 구현을 위해서 카메라의 움직임을 실시간으로 알아내는 것이 필수적이다. 기존의 가상스튜디오 구현에 사용되는 기계적인 방법을 이용한 카메라의 움직임 추적하는 방법에서 나타나는 단점들을 해결하기 위해 본 논문에서는 카메라로부터 얻어진 영상을 이용해 컴퓨터비전 기술을 응용하여 실시간으로 카메라변수들을 알아내기 위한 전체적인 알고리듬을 제안하고 실제 구현을 위한 시스템의 구성 방법에 대해 다룬다. 본 연구에서는 실시간 카메라변수 추출을 위해 영상에서 특징점을 자동으로 추출하고 인식하기 위한 방법과, 카메라 캘리브레이션 과정에서 렌즈의 왜곡특성 계산에 따른 계산량 문제를 해결하기 위한 방법을 제안한다.

Abstract

In this paper, we present an overall algorithm for real-time camera parameter extraction which is one of key elements in implementing virtual studio. The prevailing mechanical methods for tracking cameras have several disadvantages such as the price, calibration with the camera and operability. To overcome these disadvantages we calculate camera parameters directly from the input image using computer-vision technique. When using zoom lenses, it requires real time calculation of lens distortion. But in Tsai algorithm, adopted for camera calibration, it can be calculated through nonlinear optimization in triple parameter space, which usually takes long computation time. We proposed a new method, separating lens distortion parameter from the other two parameters, so that it is reduced to nonlinear optimization in one parameter space, which can be computed fast enough for real time application.

Ⅰ. 서 론

방송 분야에서 실제 물체와 그래픽으로 만들어진 배경의 합성은 오래 전부터 Chromakeying[6] 기법을 이용해서 이루어져 왔었다. Chromakeying 기법은 특정색의 배경 앞에 진행자가 위치해 있으면, 카메라로부터 얻어진 영상에서 이 특정색의 배경을 다른 영상으로 대치 시킴으로써 진행자가 다른 배경에 위치해 있는 것처럼 보이도록 하는 것이다. 이러한 Chromakeying 기술은 그 유용성 때문에 여러 분야(일기예보, 도표설명 등)에 이용되고 있지만, 카메라가 움직일 수 없으며 줌의 변화 역시 불가능하였다. 이는 카메라가 움직이거나 줌동작을 할 경우 실제 스튜디오에 위치한 진행자는 다르게 보여도 진행자의 그래픽 배경은 그에 따라 변할 수 없기 때문이었다.

최근 몇 년간 컴퓨터의 속도 향상과 그래픽 기술의 발전으로 카메라의 움직임에 따라 합성되는 배경 역시 변화시켜 보고자 하는 노력들이 있었다. 이러한 노력의 결과가 가상스튜디오의 구현이다. 가상스튜디오[1]

* 正會員, 浦項工科大學校 電子電氣工學科
(Dept. of E.E., Pohang Univ. of Sci. and Tech.)
接受日字:1999年2月26日, 수정완료일:1999年6月1日

[2] [12] [13] [14] [15] [16]는 실제의 스튜디오에서 연기를 하는 진행자가 그래픽으로 만들어진 가상의 무대에서 진행하는 것처럼 보이게 하고자 하는 것이 그 목적이다. 가상스튜디오를 구현하기 위해 필요한 기술분야는 크게 카메라의 움직임을 정확히 알아내는 카메라 트랙킹(Tracking) 기술과 카메라의 움직임에 대한 정보를 이용해 가상의 무대를 빠르게 그려주는 그래픽 기술로 나누어 볼 수 있다. 이 가운데, 본 논문은 카메라 트랙킹에 관한 내용을 다루게 된다.

기존의 가상스튜디오의 구현에서는 카메라의 움직임을 추적하기 위해서 기계장치에 의한 카메라의 움직임을 측정하는 방법이 주로 이용되어 왔었다. 이 방법은 카메라에 부착된 센서에서 측정되는 카메라의 움직임과 줌 등의 정보를 이용하는 방법으로 높은 정확도와 빠른 측정 속도때문에 처음부터 가상스튜디오 구현시 카메라의 움직임을 측정하는 방법으로 많이 이용되었다[14] [15] [16]. 하지만 이 방법에는 몇 가지 단점이 있다. 우선 이런 기계장치는 굉장히 고가이며, 또한 이 기계장치에서 측정된 값을 이용하기 위해서는 카메라에 대한 정확한 정보가 있어야 한다는 점과, 이 기계장치에 의해 측정된 값은 초기상태에 대한 상대적인 값이기 때문에 초기상태에 대한 정확한 조율이 필요하다는 점, 그리고 이런 기계장치가 카메라 한 대당 하나씩 설치되야 하기 때문에 여러 대의 카메라를 이용하기 위해서는 같은 수의 기계장치가 요구된다는 점 등이다. 영상을 이용한 카메라의 움직임을 알아내는 방법[12] [13]은 앞에서 언급한 기계장치를 이용한 방법에서의 단점들을 보완할 수 있음에도 불구하고 실시간으로 정확하게 카메라의 움직임을 계산하는데 있어서 생기는 기술적인 어려움 때문에 별로 이용되고 있지 않다. 따라서, 본 연구에서는 기계장치를 이용하는데 있어서 나타나는 단점들을 해결하기 위해 카메라로부터 얻어진 영상만을 이용해서 카메라의 움직임을 알아내기 위한 알고리즘을 제안하고 전체 시스템을 구성하기 위한 방법을 다룬다.

그림 1에서는 영상으로부터 카메라의 움직임을 알아내서 가상스튜디오를 구현하기 위한 전체적인 영상의 흐름을 보여준다. 여기서는 카메라의 움직임을 알아내기 위해 같은 색의 서로 다른 두 개의 명도를 써서 만들어진 특별한 패턴 앞에 진행자가 위치하게 된다. 카메라로부터 들어온 영상 (a)는 카메라 트랙킹을 위한 시스템의 입력으로 주어지고, 이 시스템에서는 입력 영상 중에서 실제 필요한 패턴 부분만을 뽑아낸 영상 (b)를 만들고, 패턴을 인식함으로써 카메라의 움직임을 알아내어 이를 그래픽 생성기(Graphics Generator)에 전달하고 가상의 무대 영상 (c)를 만들게 한다. 이러한 과정이 본 논문에서 다루어지는 내용이다. 또한 가상의 무대를 그려내기 위한 과정과는 별도로 카메라로부터 들어온 영상은 카메라변수 계산에 의한 시간 지연을 맞추기 위해 같은 시간만큼 지연시켜서 Chromakeyer에 의해 사람 부분의 영상 (d)만을 남기고 배경은 가상의 무대 영상 (c)로 대체하여 결과 영상 (e)가 만들어진다.

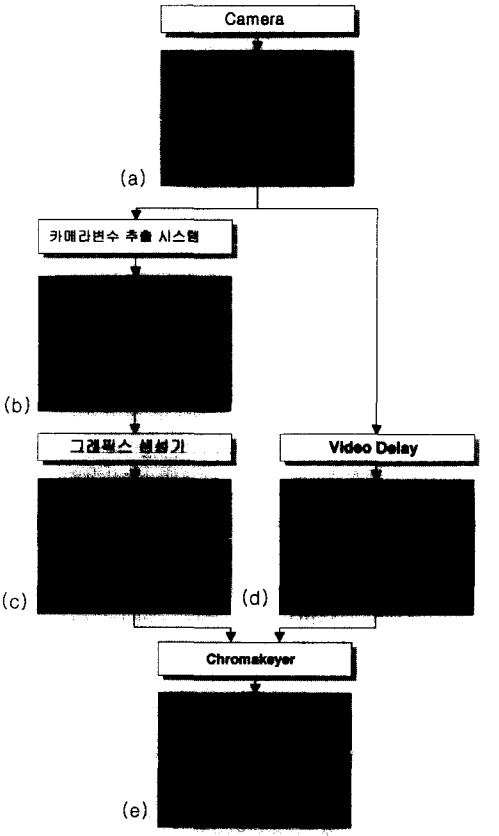


그림 1. 가상스튜디오의 구현에서의 전체 영상의 흐름
Fig. 1. Total image flow in implementing virtual studio.

영상으로부터 카메라의 움직임을 알아내기 위해서는 먼저 영상에 나타난 특징점을 자동으로 찾아내고 그 점이 공간상의 어느 점에 대응되는가를 알아내는 특징점의 인식과정이 필요하다. 영상에서 찾아진 특징점을 자동으로 인식하기 위해서는 공간 상의 점들과 영상에 나타난 그것들의 대응점에 대해서 같은 값을 갖는 성

질이 필요한데 이것을 기하적 불변량 (Geometric Invariant)[3]이라고 한다. 본 연구에서는 여러 불변량 가운데 Cross-ratio를 이용하여 패턴을 제작하고, 영상에서 불변량의 성질을 이용하여 패턴을 자동으로 찾고 인식할 수 있게 하는 방법을 제안한다.

특징점의 인식과정 다음에는 인식된 특징점의 공간좌표와 영상좌표 사이의 대응점을 이용한 카메라 캘리브레이션을 수행한다. 본 논문에서의 카메라 캘리브레이션 방법은 기본적으로 Tsai[4]에 의해 제안된 카메라 모델에 따르게 된다. 하지만 Tsai의 캘리브레이션 방법에서 렌즈의 왜곡을 계산하기 위해 사용되는 비선형최적화의 방법은 많은 계산량이 요구되므로 카메라 변수들을 실시간(30프레임/초)으로 계산하는데 있어서는 문제가 된다. 본 논문에서는 이러한 계산량 문제를 해결하기 위하여 실시간 응용에 적용이 가능한 실질적인 렌즈의 왜곡 계산 방법들을 제안한다.

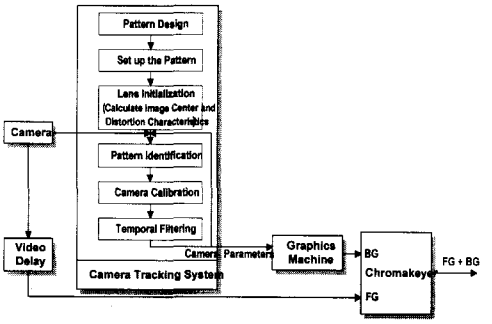


그림 2. 실시간으로 카메라변수를 알아내기 위한 전체 작업 순서도

Fig. 2. Total flowchart of calculating camera parameters in real-time.

그림 2에서는 카메라변수를 실시간으로 알아내기 위한 전체 순서도를 보여준다. 카메라변수를 추출하기 위한 준비과정으로 먼저 패턴의 제작과 실제 스튜디오에 패턴을 설치하는 작업이 필요하며, 렌즈의 초기화 과정으로 렌즈의 특성(영상중심, 렌즈의 왜곡 특성)을 측정하는 작업이 이루어진다. 카메라로부터 얻어진 영상은 카메라변수 추출을 위한 시스템의 입력으로 주어져서 패턴 인식, 카메라 캘리브레이션 그리고 Temporal 필터링까지의 과정을 통해 카메라 변수가 구해지게 된다. 구해진 카메라 변수는 그래픽 스튜디오의 생성을 위한 입력값이 되며 또한 다음 프레임에서의 특징점 인식을 위한 기준점 생성을 위해 이용된다. 이렇게 생성된 가상의 무대와 실제 영상에 나타난 진행자를 합성하기 위해서는 카메라로부터의 영상은 카메라변수 계산에서 생기는 시간 지연을 맞추기 위해서 같은 시간만큼 지연이 되고 Chromakeyer에 의해 배경 부분을 가상의 무대 영상으로 대체하게 된다.

다음 2장에서는 본 논문에서 사용된 카메라 모델에 대해서 설명하고, 3장에서는 기하적 불변량을 적용한 패턴의 제작에 대해서 다루어지며, 4장에서 실제 인식과정을 소개한다. 5장에서 다루어지는 내용은 실시간 카메라 캘리브레이션에 관한 내용이다. 여기서는 실시간 카메라변수 추출에서의 영상중심과 렌즈의 왜곡에 관한 문제를 다룬다. 그리고, 6장에서는 캘리브레이션 결과에 대한 후처리 과정으로서 Temporal 필터링이 소개되고, 7장에서는 전체적인 시스템 구성과 실제 구현 방법에 대해 소개한다. 그리고 8장에서는 본 논문에서 제시한 렌즈 왜곡변수 계산 방법에 대한 실험 결과를 보여준다.

Ⅱ. 카메라 캘리브레이션 모델

본 연구에서의 카메라 캘리브레이션 모델은 기본적으로 Tsai[4]에 의해 제안된 모델을 따르며, 렌즈의 왜곡 모델로는 가장 현저한 Radial Distortion 첫 번째 계수, κ_1만을 고려한다[5].

Tsai의 카메라 모델에서 공간좌표 (x_w, y_w, z_w)에서 영상에 나타난 점의 좌표(X_f, Y_f) 사이에는 아래와 같은 변환관계가 있다.

step 1 : 공간좌표(x_w, y_w, z_w)에서 카메라좌표 (x, y, z)로의 변환 :

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = R \begin{bmatrix} x_w \\ y_w \\ z_w \end{bmatrix} + T \tag{1}$$

여기서 회전행렬

$$R = Rot(R_x)Rot(R_y)Rot(R_z) = \begin{bmatrix} r_1 & r_2 & r_3 \\ r_4 & r_5 & r_6 \\ r_7 & r_8 & r_9 \end{bmatrix} \tag{2}$$

이며, 이동행렬

$$T = [T_x T_y T_z]^T \tag{3}$$

로 쓰여진다.

step 2 : 카메라좌표에서 왜곡이 없는 영상면좌표 (X_u, Y_u)로의 변환 :

$$X_u = f\frac{x}{z} \tag{4}$$

$$Y_u = f\frac{y}{z} \tag{5}$$

step 3 : 왜곡된 영상면좌표(X_d, Y_d)로의 변환 :

$$X_d(1+\kappa_1 r^2) = X_u \tag{6}$$

$$Y_d(1+\kappa_1 r^2) = Y_u \tag{7}$$

여기서 $r^2 = x_d^2 + Y_d^2$ 이다.

step 4 : 컴퓨터에서 얻어진 영상에서의 점의 좌표 (X_f, Y_f)

$$X_f = d_x^{-1} X_d + C_x \tag{8}$$

$$Y_f = d_y^{-1} Y_d + C_y \tag{9}$$

여기서 (C_x, C_y)는 영상중심이며, d_x, d_y 는 스케일을 나타낸다.

Tsai의 캘리브레이션 과정은 먼저 공간좌표 (x_w, y_w, z_w)와 영상좌표 (X_f, Y_f)사이의 변환관계에서 렌즈의 왜곡이 Radial Distortion의 특성으로부터 RAC(Radial Alignment Constraint), 즉, $\overrightarrow{O_t P_u}$ // $\overrightarrow{O_t P_d}$ 인 성질을 이용해 선형적으로 카메라변수 [T_x, T_y, R_x, R_y, R_z] 를 구해내게 되며, 다음 과정으로 세 변수(f, T_z, κ_1)의 비선형최적화 방법에 의해 f, T_z, κ_1 를 구하게 된다. 이러한 세 변수 비선형 최적화 과정은 만일 렌즈의 왜곡변수가 독립적으로 구해질 수 있으면 식 (6)-(9)를 이용해 영상좌표(X_f, Y_f)로부터 왜곡 보상된 영상면좌표(X_u, Y_u)를 구할 수 있으며, 식 (1)-(5)로부터 아래와 같이 f, T_z 를 포함한 선형식을 유도할 수 있다.

$$[\, y_i \;\; -Y_{ui}] \begin{bmatrix} f \\ T_z \end{bmatrix} = w_i Y_{ui} \tag{10}$$

$$y_i = r_4 x_{wi} + r_5 y_{wi} + r_6 z_{wi} + T_y \tag{11}$$

$$w_i = r_7 x_{wi} + r_8 y_{wi} + r_9 z_{wi} \tag{12}$$

여기서 R, T_x, T_y 는 이미 구해진 값이므로 y_i, w_i 는 고정된 값으로 구해진다. 이 식에서 카메라의 영상면과 특징점들을 포함한 평면이 서로 평행하게 되면 ($R_x = 0$), 선형적으로 종속적이 되므로 값을 찾을 수가 없다. 본 논문에서는 왜곡변수, κ_1,을 영상으로부터 다른 변수와 독립적으로 구해내는 방법을 제안함으로써 전체적인 카메라변수 추출 과정을 선형적인 방법으로 수행하여 실시간 구현에 적합하도록 하였다. 이 방법에 대해서는 5장에서 다루어진다.

Ⅲ. 패턴 제작

카메라의 움직임을 알아내기 위해서는 공간상에 인식이 가능한 물체가 있어야 한다. 즉, 어느 위치에서 보더라도 영상에 나타난 특징점을 찾을 수 있고, 공간상의 어느 점에 대응되는 점인지를 알 수 있어야 한다. 여기서는 인식 가능한 대상으로 진행자의 뒤에 특별한 패턴을 위치시킨다. 이 패턴상의 특징점은 공간상에서 항상 정지해 있으므로 공간상의 좌표를 미리 알 수 있고 그 점이 영상의 어느 위치에 나타나는 지를 알 수 있으면 이러한 점들의 대응쌍으로부터 카메라의 위치를 알아낼 수 있다. 패턴이 인식 가능하기 위해서는 카메라가 어느 위치, 어느 자세로 보던지 항상 같은 값을 갖는 기하적 불변량(Geometric Invariant) [3] 이 필요하다.

여기서 사용될 불변량 Cross-ratio는 한 직선에 있는 네 개의 점들 사이의 거리의 비로 정의된다.

$$C(x_1, x_2, x_3, x_4) = \frac{(x_2 - x_1)(x_4 - x_3)}{(x_4 - x_2)(x_3 - x_1)} \tag{13}$$

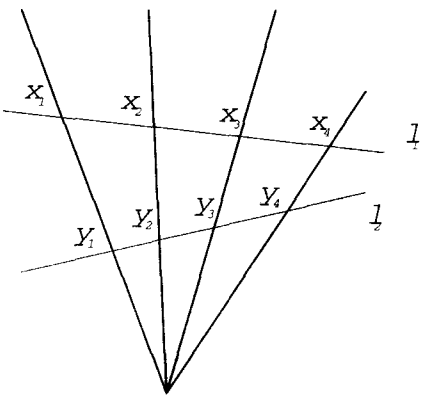


그림 3. 한 점에서 만나는 네 개의 직선(Pencil of lines)에 대한 Cross-ratio
Fig. 3. Cross-ratio of Pencil of lines.

이 Cross-ratio는 네 개의 직선에 대해서 적용되어질 수 있다. 그림 3에서는 한 점에서 만나는 네 개의 직선(Pencil of Lines)에 대한 Cross-ratio의 성질을

보여준다. 이 네 개의 직선과 만나는 직선(l_1)상의 교점들 (x_1, x_2, x_3, x_4)이 이루는 Cross-ratio는 다른 임의의 직선(l_2)과의 교점들 (y_1, y_2, y_3, y_4)이 이루는 Cross-ratio와 같은 값을 가지게 된다.

$$C_{l1} = C_{l2} = \frac{(x_2 - x_1)(x_4 - x_3)}{(x_4 - x_2)(x_3 - x_1)} = \frac{(y_2 - y_1)(y_4 - y_3)}{(y_4 - y_2)(y_3 - y_1)} \tag{14}$$

네 개의 평행한 직선에 대해서도 만나는 점이 무한대 점(Ideal Point)에 있는 경우이므로 같은 성질을 만족한다.

여기서 사용하고자 하는 패턴은 Cross-ratio의 이러한 성질을 이용하여, 평행한 가로선들과 세로선들이 서로 교차하여 격자 무늬를 이루도록 만들어진다. 그림 4는 실제 제작된 패턴의 일부분이다.

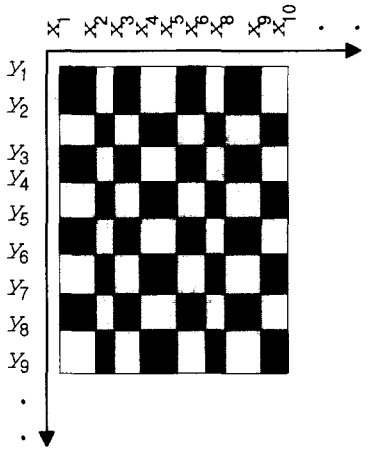


그림 4. 실제 제작된 패턴의 일부분
Fig. 4. A part of the designed pattern.

이렇게 만들어진 패턴은 영상에 나타난 패턴의 가로선들과 세로선들을 찾아서 Cross-ratio를 계산해 미리 알고 있는 패턴에서의 Cross-ratio와 비교함으로써 인식이 가능하다.

실제 패턴을 제작할 때 간격이 좁은 선이 있을 경우 인식이 어려워지므로 가능한 한 균등한 간격을 가지도록 해주는 것이 중요하다.

식 (13)에 의해 계산되는 Cross-ratio는 이론상으로 0과 1사이의 값을 가지지만 선들 사이의 간격이 너무 좁아지는 것을 막기 위해서 적용범위가 0.2-0.7 정도로 제한된다. 실제 실험에서 볼 때 영상으로부터 구해진 선들에 의한 Cross- ratio는 0.03정도의 오차를 가질 수 있었다. 따라서 적용범위 내에서 에러범위 안에 들지 않도록 Cross- ratio를 적용할 경우 10개 이하의 서로 다른 Cross- ratio밖에는 만들 수가 없다. 결국, 하나의 Cross- ratio만을 비교해서 인식할 경우 이 오차범위를 벗어나 다른 선으로 인식될 가능성이 높으므로 좀 더 안정적인 인식을 위해서는 두개 이상의 연속적인 Cross- ratio를 비교하는 방법을 사용할 수 있다. 이를 위해서는 두 개의 연속된 Cross-ratio의 쌍이 다른 쌍들과 비교해서 원하는 에러범위 안에 들어가지 않도록 Cross-ratio를 적절히 조정해 주는 것이 반드시 필요하다. 실제 실험에서 사용된 패턴을 제작할 때는 잡음에 의한 잘못된 인식을 줄이기 위해 세 개의 연속된 Cross-ratio를 비교해서 인식하도록 하였다.

평면상의 점들을 이용한 Tsai의 캘리브레이션 알고리듬을 이용할 경우, 패턴면과 카메라의 영상면이 정확히 평행이 될 경우 카메라변수를 구할 수 없으므로, 실제 스튜디오에 패턴을 설치할 때에는 벽면에 대해서 15도 이상 경사를 두어야 한다.

이 경우 카메라의 Tilt동작은 아래로는 15도 이하로 제한된다.

Ⅳ. 패턴 인식

패턴인식의 목적은 카메라변수 추출 (Camera Calibration)을 위한 실제 3차원상의 점의 좌표와 그 점이 영상에 나타난 점의 쌍을 구하는 것이다. 이러한 대응점으로서 우리가 찾고자 하는 특징점은 패턴에서의 가로선과 세로선이 만나는 교점들이다. 이 교점들을 찾기 위해서는 먼저 가로선과 세로선을 찾아야 하는데 이를 위한 기본 방법은 패턴 상의 흰 부분과 검은 부분의 (실제는 크로마키잉을 위해 진한 파란색과 약한 파란색이 이루는) 경계점들을 찾아서 이것들을 하나의 직선으로 연결하는 것이다.

패턴상의 점을 인식하는 과정은 초기 인식과정과 실제 동작과정에서 서로 다른 인식 방법을 사용하게 된다.

초기 인식과정에서는 패턴 상의 교점을 인식하기 위해 패턴의 제작과정에서 설명한 것처럼 영상에서 구해진 가로선과 세로선의 Cross-ratio를 패턴의 가로선과 세로선이 가지는 Cross-ratio와 비교함으로써 몇 번째 선인지를 인식하게 된다. 이러한 방법을 이용해

영상으로부터 자동으로 특징점을 찾고 인식할 수 있지만, 실제 적용 상에서는 몇 가지 제한점이 따르게 된다. 우선,

● 카메라가 최대로 줌-아웃(zoom-out)되어질 경우 : 영상에 나타나는 패턴의 격자간의 간격이 작아지게 되므로 (실제 실험에서 볼 때 패턴간의 간격이 5픽셀이하로 나타난다.) 두개의 연속된 선을 구별하기가 어렵다. (그림 5(a))

● 카메라가 줌-인(zoom-in) 되어질 경우 : 기본적으로 Cross-ratio를 이용한 인식방법은 연속된 가로선과 세로선들이 4개 이상 존재해야 하므로 (실제로 두개의 이상의 연속된 Cross-ratio를 비교할 경우 이보다 많은 수의 연속된 패턴상의 직선이 보여져야 된다.) 줌-인해 들어갈 수 있는 범위가 제한되게 된다. (그림 5(b))

● 패턴이 진행자에 의해 가려질 경우 : 하나의 선이 끊어져서 나타나기 때문에 잘 못 인식되어질 수가 있으며, 또한 중간에 안 보이는 선이 생기게 될 경우 연속된 선의 개수가 굉장히 제한되게 되므로 사실상 인식이 어려워진다. (그림 7(b))

(a) (b)

그림 5. Cross-ratio를 이용한 인식이 어려워지는 경우 : (a) 카메라가 최대 Zoom-Out된 경우, (b) 카메라가 최대 Zoom-In된 경우

Fig. 5. Cases that make identification using cross-ratio difficult. : (a) When the camera is maximally zoomed-out, (b) maximally zoomed-in.

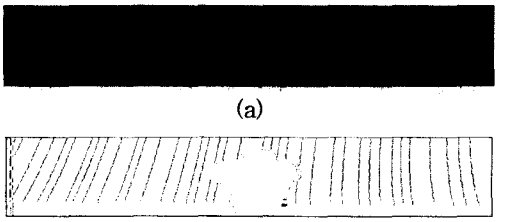
이러한 제약점 때문에 실제 인식에 있어서는, 초기 인식 과정에서 중간정도의 줌 상태에서 Cross-ratio를 이용해 패턴 상의 교점을 인식한 다음, 이 점들을 초기점으로 해서 다음 프레임에서의 교점의 위치를 예측하고 이를 추적해 나가면서 인식하게 된다.

1. 초기 인식 과정

초기 인식 과정에서의 패턴인식은 기본적으로 패턴에 적용된 Cross-ratio를 이용하여 아래와 같은 순서로 이루어진다.

● **패턴 부분 추출 (Chromakeying)** : 카메라로부터 얻어진 영상에는 인식을 위해 필요한 패턴 부분 외에 사람 등과 같은 원치 않는 물체가 포함되어진다. 이러한 것들은 인식에 있어서는 에러적인 역할을 하므로 먼저 이것을 제거해주는 작업이 필요하다. 패턴을 격자의 두 가지 색을 진한 파란색과 약한 파란색으로 칠했기 때문에 파란색을 키(Key)로한 크로마키잉[6]으로 패턴부분만을 뽑아낼 수 있다(그림 7(a)). 영상에서 특정한 색을 추출해내기 위해서 RGB형식을 이용할 경우에는 세 개의 색 밴드 (Band)를 모두 이용하여야 하므로 적절치 못하다. 따라서 RGB형식으로 입력영상이 받아들여질 경우 YUV형식[6]으로 변환해서 패턴 부분을 추출하게 된다. YUV형식에서는 U,V의 색상정보를 이용하여 우리가 원하는 파란색 패턴부분만을 뽑아낼 수 있다 (Y 밴드는 밝기정보를 나타낸다).

● **패턴상의 경계점 추출 (Gradient Filtering)** : 패턴상의 세로선들과 가로선들을 찾기 위해 우선 격자들의 경계점을 찾는 작업이 이루어진다. 길이 7의 일차미분 가우시안 필터 (First-Order Derivative Gaussian Filter)를 이용하여 가로방향과 세로방향으로 컨볼루션을 할 경우 그 결과값은 흰 부분과 검은색 부분의 경계점에서 극대값을 가지게 되며 x방향의 Gradient와 y방향의 Gradient의 절대값을 비교함으로써 경계점이 세로선에 의한 것이 가로선에 의한 것인지를 판단할 수 있다.



(a)

(b)

그림 6. (a) y축 방향으로 샘플링된 영상, (b) (a)에 대한 가우시안 필터링 결과에서의 극대값의 위치

Fig. 6. (a) An image 1/4-subsampled in y-direction, (b) positions of the local maxima of x-directional convolution.

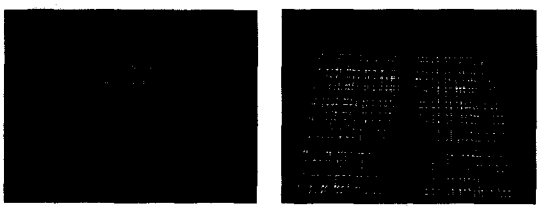
여기서 사용된 일차 미분 가우시안 필터는 실제의 실수 계수를 적당한 스케일링(Scaling)과 반올림을 통

해 [-1, -7, -15, 0, 15, 7, 1]와 같은 정수 계수를 가지도록 해주었다.

영상 전체에 대해 필터링을 하는 것은 시간이 많이 걸리므로 여기서는 네 줄에 한번씩 필터링하는 방법으로 계산시간을 줄였다.

그림 6은 전체 영상에 대해서 세로축에 대해 1/4로 축소한 영상(a)에 대해 가로방향으로 가우시안 필터를 적용한 결과에서 극대값을 가지는 점들의 위치를 보여준다 (b). 여기서의 극대값들은 패턴의 세로선의 경계점들에서 발생하게 된다.

● **패턴상의 특징점 인식** : 위의 방법으로 구한 극대값(Local Maximum)이 패턴상에서의 경계점들이 되며 가로 방향의 필터링으로 구해진 세로선의 경계점들을 연결하여 패턴에서의 세로선들을 찾을 수가 있고, 같은 방법으로 가로선들을 찾을 수가 있다. 그림 7(b)에서는 (a)영상에 대해서 찾아진 가로선과 세로선들을 직선으로 피팅(fitting)한 선들을 보여준다.



(a) (b)

그림 7. (a) 카메라로부터 들어온 영상에서 패턴부분만 추출한 영상, (b) Gradient 필터링 결과에서 극대값을 연결해 찾은 가로선과 세로선

Fig. 7. (a) Pattern extracted from an input image, (b) Vertical lines and horizontal lines fitted.

이렇게 찾아진 가로선과 세로선을 이루는 경계점은 원래 직선이지만 실제 영상에는 렌즈의 왜곡현상 때문에 실제 직선이 아닌 곡선의 형태로 나타나게 된다. 그림 7(b)에서도 보여지듯이 본래 패턴 상에서의 하나인 직선이 사람에 의해 가려지면서 나뉘어져서 두 개로 피팅될 때 두 직선의 기울기가 서로 다름을 볼 수 있다.

이렇게 곡선으로 나타난 가로선과 세로선을 직선으로 피팅할 경우 Cross-ratio는 왜곡현상 때문에 이 선들에 대해서는 보존되지 않게 된다. 따라서 정확한 피팅을 위해서는 아래와 같이 렌즈의 왜곡변수를 고려한 이차곡선으로의 피팅이 필요하다.

$$Y_d = aX_d + b(1+x_1 r^2) = aX_d + b(1+x_1(X_d^2+Y_d^2)) \tag{15}$$

식 (8),(9)를 이용해 구해진 영상면좌표(X_d, Y_d)를 이용해 식 (15)을 피팅해서 계수 a, b, x_1를 구하고, 여기서 $x_1=0$으로 두면 왜곡이 보상된 점에 대한 직선식을 구할 수 있다. 이렇게 구해진 직선들을 패턴의 가로선들과 세로선들의 Cross-ratio와 비교함으로써 영상에서 찾아진 선들을 인식할 수 있다. 또한, 영상에서의 특징점은 식 (15)에 의해 피팅된 가로선들과 세로선들의 교점으로 정확하게 구할 수 있다.

2. 실제 동작 시 인식 과정

위와 같은 초기화 과정에서 계산된 카메라변수가 일정 프레임에 거쳐서 안정하게 계산되어 질 때 인식된 교점들을 초기점의 위치로 해서 연속된 프레임에서 교점의 위치변화를 추적하도록 인식방법이 바뀐다. 이 과정에서는 Cross-ratio를 이용한 경계선의 인식이 필요치 않고, 패턴상의 격자들의 교점만을 정확히 찾으면 되므로 초기인식 과정에서의 제한점들을 해결할 수 있다.

영상에서 정확한 교점을 찾기 위해서 패턴의 교점이 가지는 특성을 이용하여 아래와 같은 교점필터, H,을 고안하였다.

$$H=\begin{bmatrix} -1 & 0 & 0 & 0 & 1 \\ 0 & -1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & -1 & 0 \\ 1 & 0 & 0 & 0 & -1 \end{bmatrix} \tag{16}$$

이 교점필터를 영상에 적용할 경우, 격자들의 교점 위치에서 결과값이 극대값 (Local Maximum) 또는 극소값 (Local Minimum)을 가지게 된다. 왜냐하면, 교점 필터는 45도 방향의 밝기 합에서 135도 방향의 밝기 합을 뺀 값을 출력하기 때문이고, 이 값은 당연히 격자들의 교점 위치에서 최대 또는 최소가 된다. 그림 8은 실제 영상에서 교점 근처를 확대한 영상과 이 부분에 대해서 교점필터를 적용한 결과이다.

실제 가상스튜디오를 구현하기 위해서는 카메라변수의 계산이 초당 30프레임의 속도로 계산되므로 연속된 프레임 간의 교점의 위치 변화가 그리 크지 않으므로 이전 프레임에서의 점의 위치를 기준점으로 해서 현재 프레임에서 찾아진 교점에 대해 가장 가까운 이전점을 찾음으로써 인식이 가능하다. 그러나 줌-아웃된 영상

에서는 격자간의 간격이 작아지므로 카메라의 이동이나 회전에 의해 이웃한 점을 잘못 추적할 가능성이 높아진다. 여기서는 이러한 제약점을 줄이기 위해 패턴상의 교점을 두개의 부류로 나누게 된다. 그림 8에서 보여지듯이 연속된 두 개의 교점(1)과 (2)는 서로 다른 부호의 극대값을 가지게 된다. 따라서 교점 위치를 교점필터 결과의 부호에 따라 분류하게 되면, 이웃하는 교점들끼리는 서로 다른 부류에 속하게 되므로 이웃하는 점들이 잘못 인식되는 경우는 생기지 않는다. 이러한 방법에서도 정확한 추적을 위해서는 어느 정도 카메라의 운동 속도가 제한되야 하지만 실험적으로 가상스튜디오의 구현시 요구되는 속도에 대해서는 이러한 추적방법이 가능하였다. 이러한 방법으로 인식을 할 경우 줌 아웃이나 카메라의 이동에 의해 새롭게 이미지 상에 나타난 점은 인식할 수 없게 되는데 이를 해결하기 위해서, 계산된 카메라변수를 이용해 모델 패턴상의 교점을 영상으로 투영시켜 이 점들을 다음 프레임에 대한 기준점으로 사용함으로써 새로이 나타나는 점들에 대해서도 인식이 가능하게 할 수 있다.

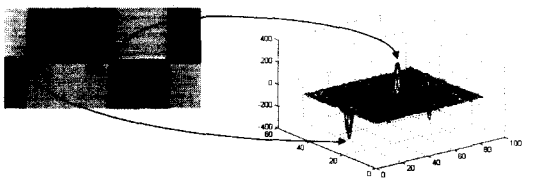


그림 8. 교점 근처 영상에 대해 교점필터를 적용한 결과
Fig. 8. Result of the convolution of the intersection-filter, H.

V. 실시간 카메라변수 추출

그래픽에 의한 가상의 배경은 사람 눈에 매끄럽게 움직이도록 하기 위해서는 초당 30프레임의 속도로 변할 수 있어야 한다. 따라서 카메라의 움직임 역시 이와 같은 속도로 계산되어야 한다. 이러한 실시간 응용분야에 있어서 카메라변수 추출에 가장 문제가 되는 점은 렌즈의 왜곡을 고려하기 위해 계산과정이 비선형 최적화 (Nonlinear Optimization)방법으로 이루어진다는 것이다. 이 방법을 이용하면 카메라변수를 정확하게 계산할 수는 있지만 많은 계산량이 요구되므로 실시간 처리의 기본적인 요구에 맞지 않게 된다.

지금까지 많은 캘리브레이션 방법들이 렌즈의 왜곡을 고려해서 수행되었지만[4] [7] [8] [9], 이 방법들은 하나의 프레임에 대해서 얻어진 특징점들에 대해서 오프-라인으로 캘리브레이션이 이루어졌기 때문에 계산량 문제를 고려하지 않았다. 본 연구에서는 실시간 구현에서의 렌즈의 왜곡 계산에 따른 계산량 문제를 해결하기 위해 두 가지 실질적인 방법을 제시한다. 첫째는 초기화과정에서 미리 구해진 렌즈의 왜곡 특성을 실제 실시간 동작에서 참조하게 되며, 나머지 한가지는 초기화 과정이 없이 영상으로부터 직접 왜곡변수를 구해내는 방법이다. 이 방법으로 영상으로부터 직접 왜곡변수, κ_1를 계산하기 위해서는 영상 중심이 사전에 미리 주어져야 한다. 영상 중심을 구하기 위해서 본 논문에서는 초기화 과정으로 카메라가 정지된 상태에서 줌동작만을 했을 경우에 대해서 서로 다른 두개의 프레임에 나타난 같은 점들을 연결해 이 선들의 교점을 고정된 영상중심(Image Center)[10] [11]으로 사용하였다. 이상적인 렌즈의 Optical Axis가 영상면에 수직이고 변하지 않는다고 할 때, 영상 중심은 카메라의 줌동작 동안 고정된 값으로 계산된다. 그러나 실제 렌즈의 불완전한 특성 때문에 카메라의 줌동작 동안 영상 중심 역시 변하게 되는데 이 변화량은 적용범위 이내에서 2픽셀 이하이다. 따라서 본 연구에서는 이러한 변화를 무시하고 이상적인 렌즈를 가정하여, 줌동작에 의한 영상 중심을 구하게 된다.

1. $f-\kappa_1$ LUT 참조 방법에 의한 왜곡변수 계산

렌즈의 왜곡은 줌동작이 가능하지 않은 렌즈에 대해서는 하나의 고정된 값으로 사용할 수 있지만, 줌렌즈에서는 여러 개의 렌즈에 의한 조합으로 줌동작이 이루어지므로 전체적인 렌즈뭉치의 왜곡변수가 줌동작에 의해 변하게 된다. 이러한 렌즈의 왜곡변수에 영향을 주는 요인은 줌동작 동안의 렌즈들의 복합적인 상호 거리관계의 변화에 기인하므로 렌즈에 의한 초점거리 역시 줌동작 동안 연속적으로 변하게 된다. 본 절에서는 이러한 줌동작 동안의 초점거리, f ,와 왜곡변수, κ_1, 사이의 관계를 초기화 과정에서 LUT (Look-Up Table)형태로 만들고 실시간 동작에서는 이를 참조하는 방법을 이용한다. 초기화과정에서는 카메라의 줌동작이 최대 줌-아웃 (Zoom-Out)에서부터 최대 줌-인 (Zoom-In)까지 천천히 변하도록 하면서 얻어진 영상의 특징점들을 저장한 뒤 오프-라인으로 비선형최적화 방법을 이용해 계산된 f와 κ_1을 LUT로 만들게 된다. 이 경우 카메라변수 추출에 있어서 공간상의 특징

점들이 모두 하나의 평면상에 존재할 때는 초점거리, f와 z-방향으로의 이동, T_z가 상호 연관(Coupling)되어 계산값의 안정성이 결여되기 쉽다. 그래서, 이 실험에서는 평면 상에 존재하는 특징점들에 대해서 T_z/f를 인덱스로 사용하는 편법을 이용하였으며, 이 방법은 고정된 카메라에 한정되어서 사용될 수 있다. 만일, 카메라가 전.후진할 경우에 LUT 방법을 적용하기 위해서는 3차원 패턴을 사용해야 하겠지만, 다음에 설명하는 Colinearity를 이용하는 방법은 그러한 제약이 없다. 실시간 처리에서는 30프레임/초의 짧은 시간 간격으로 카메라변수 들이 계산되기 때문에 실제 연속적인 프레임간의 카메라변수의 변화가 크지 않다. 그래서 실제 인덱스로 사용될 T_z/f는 이전 프레임의 결과를 이용할 수가 있으며, 시간이 허용되는 한도 내에서의 참조과정의 반복을 통해 좀더 정확한 왜곡변수를 찾을 수가 있다. 즉, 이전 프레임에서의 T_z/f에 의해 참조된 x_1를 이용해 선형적인 캘리브레이션을 수행하고, 다시 결과로 구해진 새로운 T_z/f에 의한 참조를 반복하는 것이다.

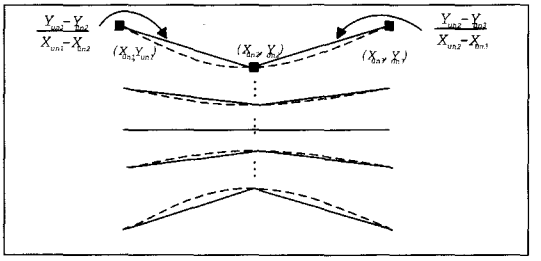


그림 9. 영상에 나타난 N개의 가로선에 대해 Collinearity가 성립하도록 하는 최적의 왜곡 변수계산

Fig. 9. Finding lens distortion using collinearity preservation.

2. Collinearity를 이용한 렌즈 왜곡변수, x_1,계산

Collinearity란 공간 상에서의 직선은 영상으로 투영될 때에도 직선으로 나타나는 성질을 말하는데, 이 성질은 렌즈의 왜곡이 없는 경우에 성립되며, 렌즈에 왜곡이 있는 경우에는 공간 상의 직선이 영상에서는 곡선으로 나타나게 된다. 여기서는 이러한 Collinearity성질을 이용하여 영상으로부터 직접 왜곡변수, x_1를 구해내기 위해, 인식된 교점들에 대해서 원래 하나의 직선에 속하는 점들이 왜곡보상 되었을 때 가장 직선이 되게 하는 왜곡변수, x_1를 구해내게 된다. 실제 구현에서는 그림 9에서처럼 영상에 나타난 N개의 가로선에 각각에 대해서 같은 가로선에 속하는 교점들 가운데 세 개의 교점 (가운데 점과 양 끝점)을 찾아서, 식 (8), (9)를 이용해 왜곡된 영상면좌표 (X_{dni}, Y_{dni}) 를 구하고, 가정된 왜곡변수, x_1,에 대해 식 (6), (7)을 이용해 구한 왜곡 보상된 영상면좌표 (X_{uni}, Y_{uni}) 에 대해 아래와 같이 에러 함수를 정의하고,

$$E(x_1)=\frac{1}{N}\sum_{n=1}^{N}\left|\frac{Y_{un1}-Y_{un2}}{X_{un1}-X_{un2}}-\frac{Y_{un2}-Y_{un3}}{X_{un2}-X_{un3}}\right|^2, \quad (17)$$

이것을 최소화하는 왜곡변수, x_1,를 구하였다.

$$x_1=\min_{x_1}E(x_1). \quad (18)$$

이 과정을 통해 왜곡변수, x_1를 구하는 과정 역시 비선형최적화에 의해서 구하지만 한번의 반복수행(Iteration)에 따른 계산량이 굉장히 적으며, 또한 비선형최적화의 초기치로 이전 프레임에서 구해진 왜곡변수를 사용하면, 연속된 프레임에서의 카메라변수의 변화가 그리 크지 않기 때문에 적은 수의 반복수행에 의해서도 원하는 에러수준 이하로 수렴되었다.

VI. Temporal Filtering

모든 시스템에는 필연적으로 잡음이 따르게 된다. 이 잡음의 영향으로 카메라가 정지상태에 있을 때 역시 찾아진 교점이 조금씩 변하게 되므로 구해진 카메라변수 역시 변하게 되고 결과적으로 그래픽으로 생성된 가상의 무대에 떨림이 나타나게 된다.

본 논문에서는 이러한 잡음의 영향을 줄이기 위해 구해진 카메라변수 값들에 대하여 필터링을 하게 된다. 시간축에 대해서 구해진 카메라변수에 대해 필터링을 할 경우 필연적으로 시간 지연이 생기게 되며 가상스튜디오 구현에 있어서는 이러한 시간 지연이 항상 같은 값을 가지게 하는 것이 필수적이다. 필터링 방법으로 많은 방법이 연구되었지만 실제 적용에서는 이러한 제약점 때문에 예측(Prediction)이 들어가는 필터링 방법(예를 들면, Kalman Filter)은 사용할 수가 없었다. 그래서 본 논문에서는 일정길이의 평균필터를 사용하였다.

$$X(k)=\frac{1}{N}\sum_{n=0}^{N}Z(k-n), \quad (19)$$

여기서 $Z(k)$는 실제 측정된 카메라 변수이고 $X(k)$는 필터링된 값이다. 여기서 N이 필터의 길이가 되며 N이 클수록 잡음을 없애는 효과가 좋지만 시간 지연 역시 길어지므로 적당한 N을 결정하는 것이 중요하다. 실제로 N의 결정은 정지상태에서 필터링된 카메라 변수에 의해 생성된 그래픽이 카메라가 정지해 있는 상태에서 떨림이 보이지 않는 범위 안에서 최대한 작게 잡아주는 방법으로 결정을 하였으며, 실제 실험에서 3이나 5정도면 이러한 요구를 만족시키는 것으로 나타났다.

Ⅶ. 실시간 구현

카메라변수를 실시간으로 알아내기 위한 전체 시스템의 구성은 TMS320C80 프로세서를 탑재하고 Grab 모듈이 부착된 Main DSP Board 한 장과 TMS 320C80 DSP 프로세서를 각각 두개 탑재한 프로세싱 보드 두 장, 그리고 전체 동기를 조절하며 호스트역할을 하게될' Pentium 200 PC 한 대로 구성되어진다. 이러한 구성은 5개의 프로세서가 각각 하나씩의 실행 노드를 구성하고 각기 독립적으로 동작하는 것을 가능케 한다. 각 노드들은 각자 자신의 메모리를 가지며, 각 보드사이에는 데이터 채널이 연결되어 각 노드 사이의 메모리로의 빠른 데이터 전송이 가능하다. 또한 호스트 역할을 하는 PC와는 PCI버스를 통해 데이터의 전송이 이루어진다. 호스트 PC가 각 노드 간의 동기화를 이루어주며 5개의 TMS320C80 프로세서가 서로 병렬적으로 동작하게 할 수 있게 한다. 여기서 사용되는 5개의 프로세서는 순서적으로 연속적인 프레임을 하나씩 맡아서 처리하게 된다. 따라서 하나의 프로세서가 하나의 프레임을 165msec(33msec x 5processor)내에 처리를 해야만 전체적인 실시간 처리가 가능하다.

각 DSP 프로세서 (TI사의 TMS320C80 Processor)는 하나의 프레임을 카메라로부터 얻어서 그 영상에 대해서 패턴 부분을 추출하고, 초기인식 과정에서는 가우시안 필터링을 통한 경계선 추출, 그리고 선의 교점을 계산하는 작업까지를 수행하게 되며, 실제 동작 시에는 교점필터를 이용한 패턴상의 교점을 찾는 일을 수행한다. 이렇게 찾아진 선에 대한 정보와 교점의 위치는 다음 처리과정을 위해서 PCI버스를 통해 PC 메모리의 지정된 위치로 옮겨지고, DSP 프로세서는 이미 받아들여져 있는 다음 프레임에 대한 처리를 시작하게 된다. DSP 프로세서는 실수 연산이 가능하지만 실제로는 그 속도가 느려서 많은 실수연산이 요구되는 경계선의 인식이나 카메라 캘리브레이션 등의 작업에는 적합치가 않다. 따라서 이 작업들은 각각의 프로세서에서 보내져 온 경계선의 정보와 교점의 위치를 이용하여 PC의 CPU가 처리를 하게 된다. PC는 각각의 프레임에 대해서 연속적으로 처리하게 되므로 하나의 프레임에 대한 처리가 33msec이내에 이루어져야만 실시간 처리가 가능하게 된다. 표 1에서는 하나의 프로세서가 수행하는 작업들과 수행시간, 그리고 PC가 처리해야 하는 작업과 수행시간을 보여준다.

표 1. 각 프로세서들에 대한 수행작업과 수행시간

Table 1. Processing time in each processor.

수행 프로세서	수행작업	수행시간 (msec/frame)
DSP Processor	패턴 추출	7.69
	Image subsampling	5.11
	Gradient Filtering	22.03
	선, 교점 찾기	75.00
	PC에 결과 전달	3.00
PC CPU	선 인식	11.00
	교점 인식	12.00
	카메라변수 계산	3.2
	다음프레임의 기준점 계산, 결과 전송	4.00

여기서 각각의 프로세서에서의 수행시간은 Grab 타이밍을 놓치기 않기 위해서 약간의 여유가 필요하며, PC에서의 작업시간 역시 프로세서들의 동기를 정확히 조절하기 위해서는 실제 33msec보다 적은 30msec내에 작업을 처리해 주는 게 좋다. 실제 운용에서는 PC에서 계산된 카메라변수를 그래픽생성기에 RS-232C를 통해 전송되므로 이 전송시간이 추가로 요구된다.

그림 10에서는 5개의 프로세서가 연속적인 프레임에 대해서 병렬적으로 동작하게 하는 시간흐름과 PC가 정보를 처리하는 시간흐름을 나타낸다. 여기서 우선 t_1의 시간에 들어가기 전에 t_0의 시간에 1번 프로세서에는 미리 Grab명령이 주어지며 1번 프로세서가 t_1의 시간에 t_0의 시간에서의 프레임 fg_0에 대한 Grab

이 완료되었음을 확인하고 프레임 fg_0에 대한 처리를 시작할 때 2번 프로세서에 t_1시간의 Grab명령이 주어지게 된다. 연속적으로 다음 시간 t_2에서는 2번 프로세서가 t_1에서의 프레임 fg_1에 대한 Grab동작이 완료되었음을 확인하고 처리를 시작할 때 3번 프로세서에 t_2시간의 Grab 명령이 주어지게 된다. 여기서 하나의 시간간격은 영상신호의 수직동기가 발생하는 33msec 단위이다. 이런 작업이 계속되면 t_5의 시간에 다시 1번 프로세서에 Grab명령이 주어지게 되는데, 이 때까지 프레임 fg_0에 대한 처리가 끝나지 않게 되므로 저장공간의 중복을 피하기 위해 이중 버퍼링 (Double Buffering)방법을 이용해서 서로 다른 두개의 저장공간에 교대로 영상이 Grab되도록 해주었다. 하나의 프로세서 내에서는 Grab동작과 프로세싱을 동시에 할 수 있으므로 이 시간 동안에도 계속해서 프로세싱이 이루어질 수 있다.

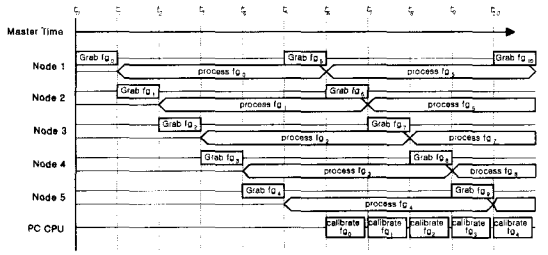


그림 10. 프로세서간의 병렬적인 처리
Fig. 10. Parallel processing of 5 nodes.

이런 식의 여러 개의 프로세서에 의한 병렬적인 작업은 필연적으로 시간 지연이 발생하게 된다. 여기서는 다섯 개 프로세서의 병렬적인 작업과 PC에 의해 계산되는 시간 지연을 더해서 6프레임의 시간 지연이 발생하게 된다. 총 시간 지연은 위에서 언급한 Temporal 필터링 과정에서 필터 길이를 5로 했을 경우 발생하게 되는 2 프레임의 시간 지연이 더해져서 총 8 프레임의 시간 지연이 발생하게 된다. 실제 구현에 있어서 이러한 시간 지연 문제를 해결하기 위해서는 추출된 진행자의 영상에 대해서도 같은 정도의 시간 지연을 줘서 동일한 시간에서 구한 카메라 변수가 이용되도록 해야 한다.

파란색 패턴 부분 추출은 RGB형식의 영상을 YUV형식으로 변환해서 이루어지며, 하나의 TMS 320C80 프로세서에 의해 수행 될 경우 40msec정도의 수행시간이 필요하다. 따라서 실시간 구현을 위해서는 두개의 프로세싱 노드로 하나의 영상을 나눠서 작업을 수행해야 한다. 근래에는 이러한 컬러 형식의 변환을 하드웨어의 도움으로 수행할 수 있으며 또한 방송용 디지털 카메라의 발전으로 직접 YUV출력을 얻을 수 있게 되어서 실제 변환 시간을 없앨 수 있다. 표 1에서의 수행시간은 카메라로부터 YUV출력을 얻어서 계산하는 경우를 가정한다.

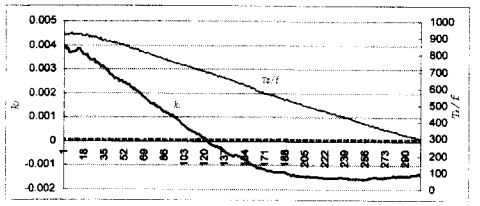
Ⅷ. 렌즈의 왜곡 계산에 대한 실험 결과

본 장에서는 위에서 소개된 실시간 처리에 적합한 렌즈의 왜곡변수 계산 방법에 대한 실험결과를 보여준다.

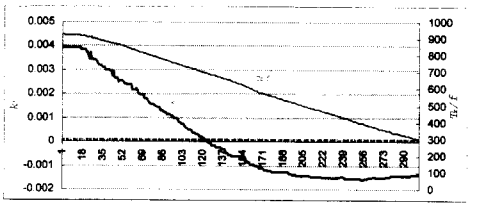
처음 제안된 $f-\kappa_1$ LUT 방법에서는 줌동작 동안의 초점 거리, f,와 렌즈의 왜곡변수, κ_1, 사이의 관계를 미리 계산해서 LUT형태로 만든 후 실시간 처리에서는 이를 참조하게 된다. 이 실험에서는 특징점들이 모두 공간 상에서 하나의 평면상에 존재하기 때문에 초점거리, f,와 z-방향으로 이동, T_z,를 정확하게 분리해 내기가 어렵다. 그래서 이번 실험에서는 고정된 카메라를 가정하고 T_z/f를 인덱스로 해서 κ_1의 LUT을 생성하였다. 그림 11(a)는 초기화과정에서 Tsai모델에 의한 세변수 최적화에 의해 구해진 줌동작 동안의 T_z/f와 κ_1의 관계이다. 이 그래프에서 보여지듯이 $T_z/f-\kappa_1$ LUT는 카메라의 줌-인동작동안 선형적으로 변하는 T_z/f에 대해 κ_1를 매핑시키는 것으로 볼 수 있다.

실시간 응용에 있어서는 카메라 변수가 실시간 (30 프레임/초)의 빠른 속도로 계산되기 때문에 프레임간의 카메라 변수의 변화는 작다고 가정할 수 있다. 이러한 가정 하에서 현재 들어온 프레임에 대한 왜곡변수, κ_1,은 이전 프레임에서 계산된 T_z/f를 이용해 $T_z/f-\kappa_1$ LUT에서 찾아지게 되며, 이 왜곡변수, κ_1, 을 이용해 다른 카메라변수들을 선형적으로 구할 수 있게 된다. 좀 더 정확한 계산을 위해서는 이러한 과정을 여러 번 반복할 수 있다. 그림 11(b)에서는 이러한 반복과정이 없이 이전 프레임에서의 T_z/f를 인덱스 이용해서 찾아진 κ_1를 이용해 선형적인 방법에 의한 카메라 캘리브레이션을 수행한 결과를 보여준다.

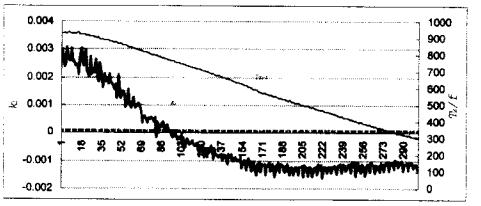
본 논문에서는 구해진 x_1의 정확성 비교를 위하여 구해진 카메라 변수를 이용해 공간좌표를 영상으로 투영시킨 점의 위치와 영상에서 찾아진 특징점의 위치 사이의 거리로부터 재투영 에러 (UDPE or Undistorted Projection Error)를 정의하였다.



(a)



(b)

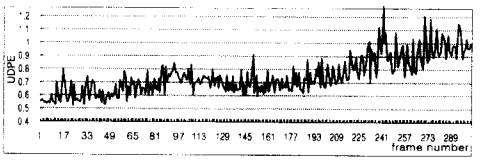


(c)

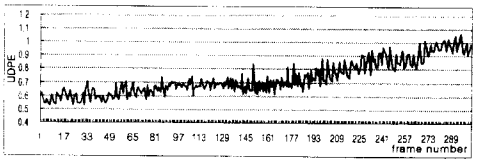
그림 11. 세 가지 렌즈 왜곡변수 계산 방법에 의해서 구해진 T_z/f, x_1 : (a) Tsai 모델에 의한 세 변수 (f, T_z, x_1) 최적화 방법, (b) $T_z/f-x_1$ LUT 참조 방법, (c) Collinearity 보존 성질을 이용한 렌즈 왜곡변수 계산 방법. 여기서 가로축은 프레임 번호를 나타낸다.

Fig. 11. The resulting T_z/f and x_1 of the (a) Three-variable optimization method, (b) $T_z/f-x_1$ LUT -based method, (c) collinearity method.

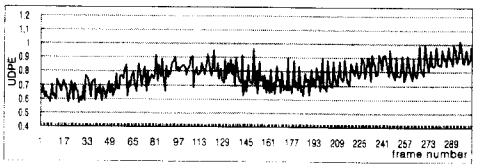
그림 12(a), (b)는 각각 Tsai 모델에 의한 세변수 최적화에 의한 UDPE와 $T_z/f-x_1$ LUT 참조 방법에 의한 UDPE결과를 보여준다. 이 결과에서 보여지듯이 두 방법에 의한 UDPE는 거의 비슷한 결과를 보여주며, 프레임번호가 증가함에 따라 UDPE가 증가하는 이유는 카메라가 줌-인해 가면서 영상에서 찾아지는 특징점의 수가 감소하였기 때문이다. 그림 11(c)는 본 논문에서 제시한 Collinearity 보존 성질을 이용한 렌즈 왜곡 변수 계산에 대한 결과를 보여준다. 여기서의 비선형최적화의 초기치로는 이전 프레임에서 구해진 왜곡변수를 사용한다. 연속된 프레임에서의 카메라변수의 변화가 그리 크지 않기 때문에 실험적으로 약 20번 이내에 원하는 에러수준 이하로 수렴되었다. 실험에서 사용된 패턴상의 최대 20개의 가로선에 대해서 위의 방법으로 최적화를 수행할 경우 시간은 1.2msec 이하로 (Pentium 200MHz) 선형적인 방법에 의한 카메라 캘리브레이션을 수행하는 시간을 더해 3.2msec (300개 점)내에 카메라 변수들을 찾을 수 있다(Tsai의 모델에 의한 세변수 최적화의 경우 30msec이상이 걸린다). 그림 12(c)는 앞에서의 결과와 마찬가지로 이 방법에 대한 UDPE 결과를 보여주고 있다. 여기서의 결과도 위의 두 방법에 의한 결과와 거의 비슷한 정도의 에러를 가진다.



(a)



(b)



(c)

그림 12. 세 가지 렌즈 왜곡변수 계산 방법에 의한 UDPE : (a) Tsai 모델에 의한 세변수 (f, T_z, x_1) 최적화 방법, (b) $T_z/f-x_1$ LUT 참조 방법, (c) Collinearity 보존 성질을 이용한 렌즈 왜곡변수 계산 방법. 여기서 가로축은 프레임 번호를 나타낸다.

Fig. 12. Reprojection errors of the (a) Three-variable optimization method, (b) $T_z/f-x_1$ LUT method, (c) collinearity preservation method.

Ⅸ. 결 론

본 논문에서는 가상스튜디오를 구현하기 위해 필수적인 카메라변수를 추출하기 위한 전체적인 시스템의 구성과 방법 등을 소개하였다. 가상스튜디오의 결과는 무엇보다도 진행자와 함께 합성된 그래픽으로 만들어진 가상의 무대가 사람 눈에 그럴듯해 보이게 되는가 하는 것이기 때문에 이를 위해서는 실제 카메라의 움직임을 정확히 그리고 빠르게 알아내는 것이 필수적이다. 본 논문에서는 영상에서의 특징점을 자동으로 찾고 인식하게 하기 위해서 기하적 불변량을 적용한 특별한 패턴을 제작하였으며, 이러한 특성을 이용한 인식 방법을 소개하였다.

실시간 가상스튜디오의 구현에 있어서의 카메라 변수계산에는 렌즈의 왜곡특성 계산을 위한 비선형최적화에 따른 계산량이 문제가 된다. 본 논문에서는 이러한 문제를 해결하기 위해 두 가지의 실질적인 렌즈 왜곡특성 계산을 위한 방법을 제시하였다. 이 가운데 $f-x_1$ LUT을 참조하는 방법은 실제 계산에 의하여 렌즈 왜곡특성을 구하지 않고, 초기화과정에서 측정된 렌즈의 왜곡특성을 참조함으로써 렌즈의 왜곡 계산 시간을 없앨 수 있다. 하지만 공간 상의 특징점들이 모두 하나의 평면상에 존재하는 경우에는 f와 T_z가 서로 연관되어 잘 분리가 되지 않기 때문에 T_z/f를 왜곡특성 참조의 인덱스로 사용하였다. 이 경우 정지된 카메라에 대해서만 적용이 가능하다. 이러한 카메라 동작의 제약성을 해결하기 위하여 Collinearity 특성을 이용한 렌즈의 왜곡특성 계산 방법을 제안하였다. 이 방법에서도 렌즈의 왜곡특성 계산을 위해서 비선형 최적화 방법이 사용되지만 계산량이 적어서 실제 실시간 적용에 적합하다.

영상으로부터 추출된 카메라변수는 영상에서의 잡음과 추출된 교점에서의 오차 때문에 실제값과 차이가 있게 된다. 이러한 원인으로 해서 실제 카메라가 정지상태에 있을 경우에도 카메라변수가 조금씩 변하게 되고 이 영향은 결국 생성된 그래픽 배경의 떨림으로 나타나게 된다. 이러한 영향을 줄이기 위해서는 무엇보다도 영상으로부터의 교점을 정확하게 찾아내는 것이 중요하며, 실제 실시간 처리에서는 많은 계산량을 요구하는 방법은 적용하기가 어렵다. 본 논문에서 제안한 교점필터의 방법으로는 대략 half-pixel정도의 정확도로 교점을 찾아낼 수가 있다. 이러한 정확도에서 생기게 되는 떨림을 없애기 위해서 본 논문에서는 구해진 카메라변수에 대한 Temporal Filtering과정을 취했다. 이러한 필터링 과정은 필연적으로 시간 지연을 발생시키며, 5개의 프로세서에 의한 병렬 처리에서 발생하는 시간 지연까지 더해져서 실제 시스템의 결과에는 8프레임의 시간 지연이 발생한다(필터길이를 5로 할 경우). 이러한 시간 지연을 보상하기 위해서는 영상에서 추출된 진행자에 대해서도 같은 시간 지연을 가지게 해야 하므로 별도의 시간 지연을 위한 하드웨어장치가 요구되게 된다. 이러한 시간 지연을 줄이기 위해서는 필터링 부분에서 실시간 적용에 적합하고 보다 정확하게 특징점을 추출할 수 있는 방법이 요구되며, 또한 시간 지연을 줄이기 위한 효과적인 병렬처리 방법이 필요하다.

기계장치에 의한 카메라 움직임 측정에서는 카메라 한 대마다 별도의 기계장치가 설치되어야 하는데, 영상으로부터 직접 카메라의 움직임을 알아내는 방법에서는 여러 대의 카메라에 대해서도 적용이 가능하다. 본 논문에서 사용된 방법을 여러 대의 카메라로부터 들어오는 영상에 대해서 적용하기 위해서는 우선 초기화 과정에서 각 카메라의 렌즈에 대한 특성이 미리 측정되어야 하며, 카메라가 바뀌는 순간에는 특징점의 인식 과정이 불변량을 이용한 초기인식 과정에 의해서 이루어지므로 인식이 어려워지는 경우 즉, 최대 줌인 되거나, 최대 줌아웃되어 있는 장면으로의 전환이 불가능하다는 제한점이 따르게 되는데, 이것은 영상을 이용한 카메라 움직임 추적 방법에서 해결해야될 문제이다.

참 고 문 헌

[1] Laurent Blondé, Matthias Buck, Ricardo Galli, Wolfgan Niem, Yakup Paker, Wolfgang Schmidt and Graham Thomas, "A Virtual Studio for Live Broadcasting : The Mona Lisan Project," *IEEE Multimedia*, 1996, pp. 18-29.

[2] Masaki Hayashi, "Image Compositing Based on Virtual Cameras," IEEE Multimedia, 1998, pp 36-48.

[3] Christopher Coelho, Aaron Heller,

Joseph L. Mundy, David A. Forsyth and Andrew Zisserman, "An Experimental Evaluation of Projective Invariants," in Geometric Invariance in Computer Vision, edited by Joseph L.Mundy and Andrew Zisserman, MIT Press, 1992, pp87-104.

[4] Roger Y. Tsai , "A Versatile Camera Calibratin Technique for High Accuracy 3-D Maching Vision Metrology Using Off-the-shelf TV Cameras and Lenses," *IEEE Journal of Robotics and Automation*, 3(4), Aug. 1987.

[5] Reg G. Willson, "Modeling and Calibration of Automated Zoom Lenses," CMU-RI-TR-94-03 Ph.D. Thesis, 1994.

[6] Keith Jack, *Video Demystified 2nd Edition A Handbook for the Digital Engineer*, Hightext Interactive, 1996, pp. 58-61, 137-138.

[7] Megxiang Li and Jean-Marc Lavest, "Some Aspects of Zoom-Lens Camera Calibration," *IEEE Trans. on PAMI*, 1996.

[8] Megxiang Li, "Camera Calibration of the KTH Head-Eye System," TRITA-NA-9407, KTH, Sweden, 1996.

[9] Shishir Shah and J. K. Aggarwal, "A Simple Calibration Procedure for Fish-Eye (High Distortion) Lens Camera," *IEEE Conference on Robotics and Automation*, San Diego, May. 1994, Vol. 4.

[10] Reimar K.Lenz and Roger Y. Tsai, "Techniques for Calibration of the Scale Factor and Image Center for High Accuracy 3-D Maching Vision Metrology," *IEEE Transaction on PAMI*, Vol.10, No.5, Sep. 1988.

[11] Reg G. Willson, Steven A. Shafer, "What is the Center of the Image," *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, June 1993, New York.

[12] Andrew Wojdala, "Chanllenges of Virtual Set Technology," *IEEE Multimedia*, 1998, pp 50-57.

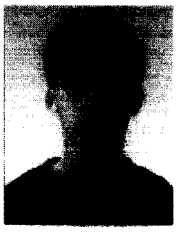
[13] CYBERSET, Orad, http://www.orad.co.il.

[14] 3DK: The Virtual Studio, GMD, http://viswiz.gmd.de/DML/vst/vst.html.

[15] ELSET : Accom, http://www.studio.sgi.com /Features/VirtualSets/accom.html.

[16] E \& S Mindset, Evans \& Sutherland, http: //www.es.com/Products/DStudio/index.html.

저 자 소 개

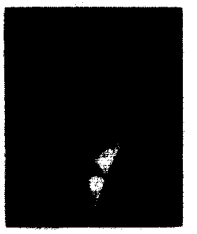


朴 星 禹(正會員)

1997년 전북대학교 전자공학과 학사. 1999년 포항공과대학교 대학원 전자전기공학과 석사. 1999년 ~ 현재 삼성 데이터 시스템(SDS). 주관심분야는 영상 처리, Virtual Studio

徐 龍 德(正會員)

1992년 경북대학교 전자공과 학사. 1994년 포항공과대학교 대학원 전자전기공학과 석사. 1994년~현재 포항공과대학교 대학원 전자전기공학과 박사 과정. 1999년 4월 ~ 5월 스웨덴 Lund Institute of Technology Mathmatical Image Group 방문 연구원. 주관심분야는 실시간 컴퓨터 시각, 카메라 자동 캘리브레이션, augmented reality



洪 起 祥(正會員)

1977년 서울대학교 전자공학사. 1979년 한국과학기술원 전기 및 전자공학과 석사. 1984년 한국과학기술원 전기및전자공학과 박사. 1984년~1989년 한국 에너지 연구소 선임 연구원. 1986년 ~ 현재 포항공과대학교 부교수. 1988년 ~ 1989년 Carnegie-Mellon 대학교 방문교수. 주관심분야는 합성개구 레이더 영상처리, 영상처리 및 컴퓨터 시각, 가상현실, 패턴 인식 등