논문-99-4-2-01

비유클리드공간 정보를 사용하는 증강현실

서용덕*, 홍기상*

Augmented Reality Using Projective Information

Yongduek Seo* and Ki-Sang Hong*

요 약

가상의 삼 차원 컴퓨터 그래픽 영상을 실제 비디오 영상과 합성하는 증강현실을 구현하기위해서는 카메라의 촛점거리 같은 내부변수와 카메라가 어떻게 움직였는지를 나타내는 회전 및 직선 운동에 대한 이동정보가 반드시 필요하다. 따라서, 기존의 방법들은 미리 카메라의 내부변수를 계산해 둔 다음, 실제 영상에서 얻어지는 정보를 이용하여 카메라의 운동정보를 계산하거나, 실제 영상에 카메라 보정을 위한 삼 차원 보정 패턴이 보이도록 한 다음 영상에서 그 패턴의 형태를 분석하여 카메라의 내부변수와 운동정보를 동시에 계산하는 방법을 사용하였다. 이 논문에서는 실제영상에서 얻어지는 정합점들로부터 카메라 보정없이 구할 수 있는 투영기하공간 카메라 이동정보를 이용하여 증강현실을 구현하는 방법을 제안한다. 실제 카메라의 내부변수와 유클리드공간 이동정보를 대신하기 위하여 가상카메라를 정의하며, 가상카메라는 실제공간과 가상 그래픽 공간의 연결을 위하여 두 장의 영상에 사용자가 삽입하는 가상공간 좌표계의 기준점들의 영상좌표로부터 구해진다. 제안하는 방법은 카메라의 내부변수에 대한 정보를 따로 구할 필요가 없으며 컴퓨터 그래픽이 지원하는 모든 기능을 비유클리드공간의 정보로도 구현이 가능하다는 것을 보여준다.

Abstract

We propose an algorithm for augmenting a real video sequence with views of graphics ojbects without metric calibration of the video camera by representing the motion of the video camera in projective space. We define a virtual camera, through which views of graphics objects are generated, attached to the real camera by specifying image locations of the world coordinate system of the virtual world. The virtual camera is decomposed into calibration and motion components in order to make full use of graphics tools. The projective motion of the real camera recovered from image matches has a function of transferring the virtual camera and makes the virtual camera move according to the motion of the real camera. The virtual camera also follows the change of the internal parameters of the real camera. This paper shows theoretical and experimental results of our application of non-metric vision to augmented reality.

Index terms: augmented reality, Euclidean geometry, virtual camera, perspective projection, projective geometry, calibration-free method

1. Introduction

Making views of a 3D graphic model and mixing them into a video in real time has been the major subject of *augmented reality*[1]. The virtual objects used in augmented reality are generally 3D graphics models and the images of them are rendered by graphics machines and overlayed on real-video frames. Applications include image-guided or assisted surgery or its training [2,3], assembly, maintenance and repair [4,5], simulating light conditions of an outdoor building [6], etc. All of these systems enrich reality or real video

* 포항공과대학교 전자전기공학과 영상처리연구실
Pohang University of Sience and Technology

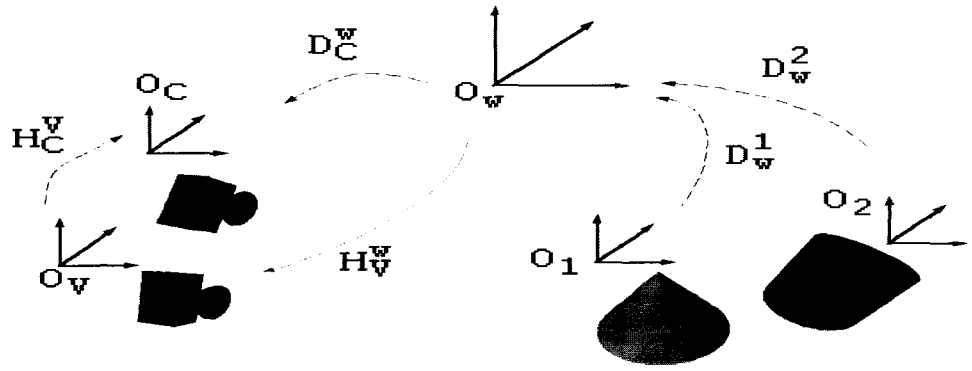

Fig 1. The relationship of various coordinate systems.

with computer-generated views of graphical objects, cooperating with computer vision techniques: estimating the camera parameters [7], resolving occlusion between a virtual object and a real object [8,9], and correcting the registered location of a graphic object dynamically [2].

Camera calibration is a prerequisite for embedding virtual objects into video frames because the geometric relationship among physical objects, virtual objects and the camera need be established to get correct views of the model object. Figure illustrates this relationship. In the figure, the coordinate system for the real camera is denoted by O_C and the coordinate system for the virtual camera looking at the graphics objects is denoted by O_V. D_V^X denotes the Euclidean displacement between two coordinate systems O_X and O_Y. The transformation between real camera O_C and virtual camera O_V is represented by the non-singular 3D projective transformation H_C^V and the transformation between O_W and O_V by H_V^W. For physically correct video augmentation, all these coordinate transformations should be known \emph{a priori}, and the usual transformation H_C^V between the physical video camera coordinate system O_C and the virtual camera coordinate system O_V is the identity transformation. The camera calibration step of the usual augmented reality systems estimates the coordinate transformation D_C^W between the world coordinate system O_W and the camera coordinate system O_C as well as the internal calibration parameters, e.g., focal length, of the real camera. The estimated parameters are then utilized for the virtual camera attached to the real one. If the camera changes its internal parameters, like focal length, and undergoes free three dimensional motion, it is necessary to calibrate the camera at every frame and in this case some reference points or patterns whose Euclidean geometry is known must be seen in the video. However, without those fiducial points, it is difficult to calibrate cameras for general video images and one should consider other ways like self-calibration methods [10,11,12] or a calibration-free augmented reality system like [13]. For example, Faugeras applied a self-calibration method for fixed internal parameters to the augmentation of real video sequences [14]. Selfcalibration provides not only calibration parameters but also the Euclidean motion of the real camera. However, the convergence issue of the methods due to a non-linear formulation may constrain practical applications.

The work of Kutulakos and Vallino [13] was innovative in the sense that they showed some results of video augmentation *without* Euclidean calibration of the video camera by applying affine object representation. However, their method could not be applied directly to the augmentation of general video sequences having perspective projection effects because they assumed an orthographic projection camera. In addition, it could not make use of the fundamental effects of computer graphics such as lighting, shading, and shadows because their affine coordinate representation does not obey the principle of computer graphics described in Euclidean geometry.

In this paper, motivated by the work of Kutulakos

and Vallino, we propose a method for augmenting a real video sequence captured by a *perspective* projection camera without the need for camera calibration procedure. There have been a lot of researches on non-metric vision applications: human and robot interface and interactive grasping with uncalibrated stereo vision [15], visual control or visual guided tasks [16,17], navigation [18,19], object recognition [20], etc. We apply the functionalities of non-metric computer vision to the augmentation of real video without giving up the fundamental features of computer graphics like lighting or shading.

Our algorithm consists of two parts: embedding and rendering. The world coordinate frame for graphics objects are specified in two selected video images in order to insert the virtual world into the real world (embedding). The view of the world coordinate system from a video image provides a virtual camera matrix, attached to the real camera, through which the graphics view is generated. We have to specify five basis points (Figure 5) of the frame in the first selected video image and four in the second. The relationship of the world coordinate frame to the virtual camera and the computation algorithm for the relationship are presented in Section 4. In short, our method directly specifies the relationship between the world coordinate system and the *virtual* camera coordinate system in image space. This relationship is denoted by D_V^W in Figure 1. Therefore, the relationship H_C^V between the virtual camera and the real camera is no longer the identity transformation but a fixed general 3D projective transformation that is unknown.

The virtual camera is modeled as a pin-hole camera with zero skew and graphics images are synthesized using the components of the virtual camera by an SGI graphics computer with the OpenGL^{TM} library (rendering). Thus, we can generate perspective views of graphic objects, as shown in Section 5. The motion and change of the internal parmaters like focal length of the real video camera is represented by projective camera matrices that can be obtained through a projective reconstruction algorithm [21,22,23]. There has been a lot of research for projective structure and motion estimation. They can be obtained by first computing the fundamental matrix or the multi-linear tensor like trifocal tensor followed by estimating the structure and motion [24,25,26,19,17]. Possible approaches for multiple views are the factorization method for projective structure [27] or the framework of affinely or projectively reduced setting [26]. Note that the reconstruction is up to a three dimensional projective transformation and finding the transformation is related to self-calibration methods. In this paper, we briefly review the theory of projective reconstruction in Section 2.1. We use in this paper a two view algorithm for a real-time application like the system of Kutulakos and Vallino [13]. The components of the virtual camera are determined by the locations of the basis points of the world coordinate frame in a video image transferred by the recovered projective structure and motion, which allows the virtual camera to move and change its internal parameters according to the change of rotation, translation and internal parameters of the real video camera. Details of this are given in Section 5.1.

The embedding procedure is similar to the method of Kutulakos and Vallino [13], but our method is different in many respects as can be seen from the comparison in Table 1. We specify five points in the first image and four in the second which define the virtual camera of the perspective projection model. In contrast, they specified four locations in the first and the second images and the camera model is an orthographic camera which cannot

Table 1. Comparison of video augmentation algorithms.

	Affine Orthographic [13]	Non-Metric Perspective
No. of init. points	(4,4)	(5,4)
3D info. used	affine	projective
Estimation difficulty	X	XX
Distortion in graphics	affine	projective
Perspectivity	impossible	possible
Shading \& shadow	impossible	possible
Euclidean pattern	not required	not required

generate a perspective view. The 3D motion and structure information utilized are affine and projective, respectively. Therefore, there are corresponding intrinsic distortions in synthesized graphics views. Estimating projective reconstruction from perspective views is more difficult than affine reconstruction from orthographic views. In contrast with the orthographic system, which cannot generate a perspective graphics view due to its intrinsic property, our uncalibrated method can synthesize such a view. Finally, the most distinctive feature, from the viewpoint of practical application, is in the usage of well developed graphics tools. The orthographic system does not decompose the affine camera into intrinsic calibration components and extrinsic components, which makes it impossible for the system to generate shading and shadow effects in graphics views. However, we decomposed the virtual camera into the two components and we can specify lightings and material properties to make shaded objects as well as their shadows.

Section 2presents some preliminaries and notations. Section 3 gives a brief overview of our algorithm. Details are given in the following sections. Section 4 deals with how to define the world coordinate frame of two selected video images and how to compute their corresponding virtual cameras. A method to compute virtual cameras for other video images and interface with the graphics library is provided in Section 5. Section 6 shows the experimental results of our method. Section 7 describes a relationship between real camera and virtual camera, and deals with the problem of different perspectives, which is inherent in our application of non-metric vision. Our method of embedding the world coordinate system might seem as if it were a method of self-calibration. However, note that the inserted coordinate frame is not a real frame but a virtual frame specified interactively, and it makes the virtual camera mimic the action of the real camera. Finally, concluding remarks are given in Section 8.

2. Preliminaries

A 2D image point is represented by a 3D homogeneous vector x with the third component being one. Also a 3D space point is represented by a 4D homogeneous vector X with the fourth component being one:

$$x=[u\ v\ 1]^T,\quad X=[X\ Y\ Z\ 1]^T. \tag{1}$$

The k-th video image is represented by I_k. The 3×4 camera matrix obtained by projective reconstruction from real video images is denoted by P_k.

A 3×4 virtual camera matrix is denoted by Q, which is decomposed into a calibration part and Euclidean motion part as follows:

$$Q = \rho K[R \mid t] \tag{2}$$
$$= [H \mid h], \tag{3}$$

where ρ is a non-zero scale, R and t are rotation and translation, respectively, and K is a 3×3 calibration matrix of the form:

$$K=\begin{bmatrix} \alpha & 0 & u_0 \\ 0 & \beta & v_0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{4}$$

We call this kind of camera a zero-skew camera. The 3×3 matrix H and 3D vector h are defined to be equal to KR and Kt, respectively, up to a scalar ρ:

$$H = \rho KR, \tag{5}$$
$$= \rho Kt. \tag{6}$$

A zero-skew camera satisfies the following proposition [28]:

Proposition 1 *Let q_1, q_2 and q_3 be three rows of $H$of a zero-skew camera Q represented in column vector form. Then*

$$(q_1\times q_3)\cdot(q_2\times q_3)=0, \tag{7}$$

where $\times$ and $\cdot$ are the outer product and the inner product of three dimensional vectors, respectively.

Notice that Equation (7) satisfies regardless of the scale factor ρ. The proof is not difficult and this proposition is useful in the computation of the matrix of

a virtual camera as shown in Section 4.2.

2.1 Projective Reconstruction

We use the results of projective reconstruction, in order to transfer the virtual camera according to the motion of the real video camera, for embedding graphic objects and rendering them. In this section, we briefly review a method for projective motion and structure estimation that has been developed in computer vision society.

Given matching points between two views, we can compute the fundamental matrix F using the equation [22,29,30]:

$$x_t^2 F x_1 = 0, \tag{8}$$

where x_1 is an image point from one view and x_2 is from the other view. Then, two camera matrices are chosen as

$$P_1 = [I \mid 0], \quad P_2 = [[e_2]_\times F \mid e_2], \tag{9}$$

where e_2 is the epipole in the second image (the left null of F, $F^T e_2 = 0$) and $[e_2]_\times$ is the 3×3 skew matrix such that $[e_2]_\times x = e_2 \times x$ [24].

A projective 3D point X_i is reconstructed from its image match(x_{1i}, x_{2i}) by back-projection equations:

$$x_{1i} \approx P_i X_i, \quad x_{2i} \approx P_2 X_i, \tag{10}$$

where the notation $\approx$ denotes equality up to scale. Solving a linear least-square equation gives X_i.

Conversely, the camera matrix P_k for the k-th image may be computed given matches (x_{ki}, X_i) of image points and their corresponding 3D points using the projection equation

$$x_{ki} \approx P_k X_i, \quad \text{for } i = 1, \cdots, N, \tag{11}$$

or more specifically

$$u_{ki} = \frac{P_k^1 X_i}{P_k^3 X_i}, \quad u_{ki} = \frac{P_k^2 X_i}{P_k^3 X_i}, \quad \text{for } i + 1, \ldots, N, \tag{12}$$

where $x_{ki} = [u_{ki}, v_{ki}, 1]^T$, P_k^n is the n-th row of the camera matrix P_k, and N is the number of the 2D-3D correspondences.

Notice that this reconstruction is up to a three dimensional projective transformation. That is, there is a 4×4 non-singular matrix $T_{4\times4}^P$ such that

$$P_k T_{4\times4}^P \approx P_k^\varepsilon \tag{13}$$

for all k, where P_k^ε is the true camera matrix that can be decomposed into its calibration part K_k^ε and Euclidean motion parts R_k^ε and t_k^ε:

$$P_k^\varepsilon = K_k^\varepsilon [R_k^\varepsilon \mid t_k^\varepsilon]. \tag{14}$$

Note that K_k^ε is an upper triangular matrix representing the internal calibration parameters of the real camera, e.g. focal length, and R_k^ε is the relative direction and t_k^ε is the relative location of the camera coordinate system with respect to the world coordinate system. As soon as the transformation $T_{4\times4}^P$ is known, the projective reconstruction may be upgraded to a Euclidean version. Estimating the projective transformation $T_{4\times4}^P$ has been a subject of self-calibration[10,11,12,31]. Note that the Euclidean camera matrices P_k^ε can be directly used for graphics rendering using a graphics machine, but projective camera matrices cannot be used because they are not of the mform of Equation (14), which is the reason we define a virtual camera. Section 7explains the relationship between the real camera and virtual camera.

2.2 Projective Plane Homography

Let us suppose that four points $\{X_i\}_{i=1,\ldots,4}$ are on the same plane Π and their image points in two different views are $\{x_i\}_{i=1,\ldots,4}$ and $\{x_i\}_{i=1,\ldots,4}$. Then there exists a 3×3 projective plane homography T_Π such that

$$x_i' \approx T_{\Pi} x_i \qquad (15)$$

as illustrated in Figure 2. We can compute T_{Π} using more than four point matches with a least-square method. When the fundamental matrix F is given, three point matches are enough because the two epipoles e and e' obey Equation (15)[29].

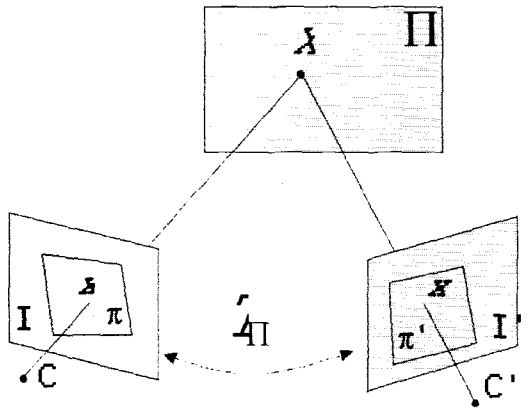

Fig. 2. Plane homography T_{Π} : $\pi \rightarrow \pi'$

3. Overview

This section gives a brief overview of our algorithm. The details of the algorithm are given in the following sections. Our algorithm can be divided into two parts: embedding and rendering.

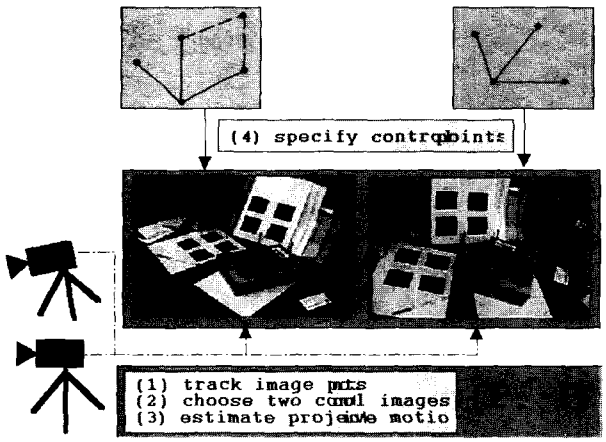

Fig. 3. Overview of our method:embedding procedure.

The steps of tracking feature points and estimating the projective motion of the video camera are included in the embedding part. First, the embedding steps, graphically shown in Figure 3, are as follows:

(1) Track feature points in the video sequence.
(2) Choose two control images onto which the graphics world coordinate frame will be inserted. The two images may be from the cameras of a real-time system or video images already captured in the case of the off-line procedure.
(3) Estimate the projective motion, P and P', and structure $\{X_i\}_{N}^{i=1}$ for the two images using point correspondences as described in Section 2.1.
(4) Specify the locations $\{(x_i^b, x_i^{b'})\}$ of the basis points of the world coordinate frame in each of the control images, five in the first and four in the second.
(5) Compute the 3D projective coordinates X_i^b, $i=0,\ldots,4$, of the five specified basis points $\{(x_i^b, x_i^{b'})\}$ using P and P'.

Second, the rendering steps, shown in Figure 4, are as follows:

(6) (a) For k-th input video image i_k, detect image corner points and compute P_k, the k-th projective camera matrix.
(b) Transfer or project the 3D points, $\{X_i^b\}_{i=0}^{4}$ onto k-th video image i_k using the projection equation $x_{ki}^b = P_k X_i^b, i=0,\ldots,4$.

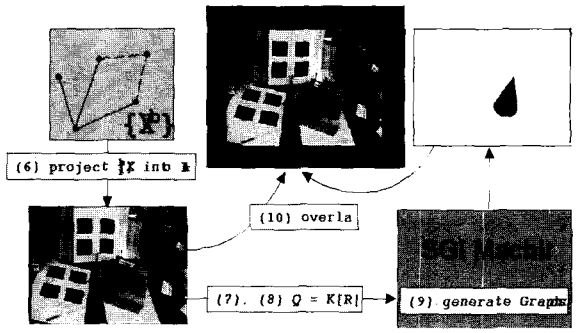

Fig. 4. Overview of our method:rendering procedure.

(7) Compute the corresponding virtual camera Q_k using the image points $\{x_{ki}^{b}\}_{i=0}^{4}$, according to the method in Section 4.2.

(8) Decompose Q_k: $Q_k = \rho_k K_k [R_k \mid t_k]$

(9) (a) Set virtual camera in graphics machine using K_k, R_k and t_k.

(b) Locate graphics objects with respect to the world coordinate system and define their characteristics like color, specularity and lighting conditions.

(10) The graphics view i_k^G, synthesized from the graphics machine, is overlayed on i_k.

4. Embedding

The first thing we have to do is to find feature matching points and compute projective camera matrices, P and P', of the two control views and the 3D projective coordinates $\{X_i\}_{i=1}^{N}$. Our projective reconstruction method estimates the fundamental matrix using image matches from the two control views. Then, two projective camera matrices, P and P', for the control views are given from Equation (9). Projective 3D coordinates $\{X_i\}_{i=1}^{N}$ of the image matches are computed using Equation (10).

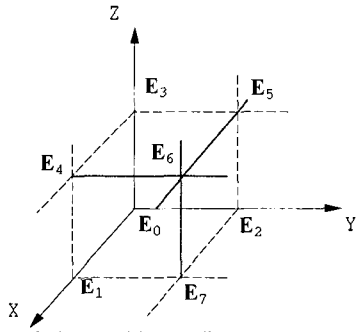

Fig. 5. Vertices of the world coordinate system

Video augmentation is accomplished by first specifying the graphic world coordinate system into the control images denoted by ν and ν'. For example, Figure 6 shows two control images τ_0 and τ_{270} from 274 video images used in our experiments.

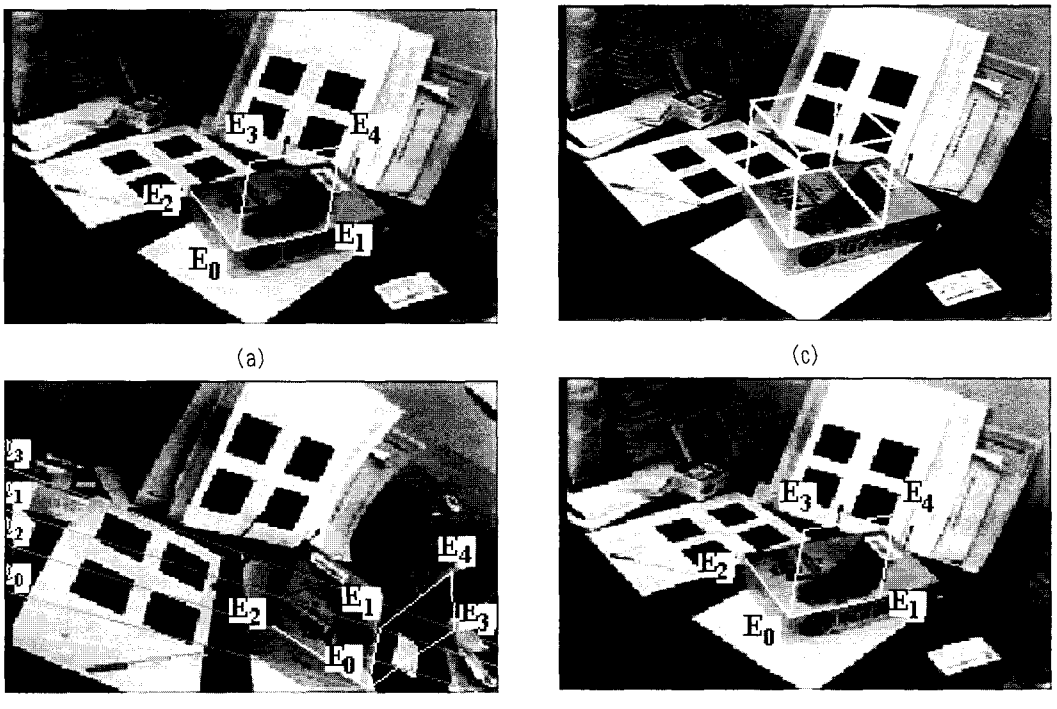

Fig. 6. (a) Five points are specified embedding in , and (b) four points on the corresponding epipolar lines in . Specified image points determine its corresponding virtual camera from which the images of the world coordinate frame are drawn in (c) and (d).

The embedding procedure consists of two steps:

1. We insert the world coordinate system of the graphics frame into the first control image ν by specifying the image locations of the five basis coordinates $\{E_0, E_1, E_2 E_3, E_4\}$ of the coordinate frame.
2. With the help of the epipolar geometry, we choose four image locations of the coordinates $\{E_0, E_1, E_2 E_3\}$ in the second control image ν'.

Here $E_0=[0,0,0,1]^T$ is the origin of the world coordinate frame and the others are the vertices (or basis points) of the unit cube, as shown in Figure 5, and they are defined as $E_1=[1,0,0,1]^T$, $E_2=[0,1,0,1]^T$, $E_3=[0,0,1,1]^T$, $E_4=[1,0,1,1]^T$ and so on.

4.1 Embedding via Epipolar Geometry)

First, *five* locations $\{x_i\}_{i=0}^4$ of the vertices are specified in the first control image ν. This determines a look of the world coordinate frame in the video image. Then we have five epipolar lines in the second image ν' using the fundamental matrix between the two control images from Equation (8). Now, we choose *four* image locations $\{x_i^{b'}\}_{i=0}^3$ of the vertices on the corresponding epipolar lines l_i':

$$l_i' = F x_i \quad i=0,\ldots,3. \tag{16}$$

However, we do not need to specify the last one, $x_4^{b'}$, because it can be determined by the plane homography T_Π which is determined by the equations $x_i^{b'} \approx T_\Pi x_i^b$, $i=0,1,3$ and $e' \approx T_{\Pi e}$ as given in Section 2.1.

Finally, we compute the projective 3D coordinates $\{X_i^b\}_{i=0}^4$ of them in order to compute the locations of the basis points--vertices--in the other video images. This enables the virtual camera to move according to the motion of the real video camera, which is explained in Section 5.1.

Figure 6shows an example where five image locations in the first control image ν_1 and four in the second ν_2 are selected. Using the image coordinates, we can compute the matrix of the corresponding virtual camera as given in Section 4.2. Then the image of the world coordinate system may be drawn in the video image, as shown in the lower part of the figure.

4.2 Virtual Camera Computation

In principle we need to estimate the Euclidean camera parameters of the real video camera in order to render a graphics view according to the motion of the video camera. However, in our case, the camera is not calibrated and we cannot use the projective camera matrices directly for graphics rendering. Therefore, our approach is to define a virtual camera that looks at the inserted world coordinate frame.

Our goal is to make the matrix of the virtual camera generate a view of the world coordinate frame that is the same as what we figured in the control images in the embedding step. Let us denote a virtual camera Q by a 3 × 4 matrix

$$Q=[a_1\ a_2\ a_3\ a_4] \tag{17}$$

where a_k is the k-th column of the matrix. The image location of a vertex E_i is denoted by $x_i^b=[u_i^b\ v_i^b\ 1]^T$. Using the relationship

$$\lambda_i x_i^b = Q E_i \tag{18}$$

we have

$$a_4 - \lambda_0 x_0^b \tag{19}$$

$$a_1 = \lambda_i x_i^b - \lambda_0 x_0^b, \quad \text{for } i\ IN\ \{1,2,3\}. \tag{20}$$

Since Q can be defined up to a global scale, we fix it temporarily: $\lambda_0=1$. Then, Q is of the form:

$$Q=[\lambda_1 x_1^b - x_0^b \quad \lambda_2 x_w^b - x_0^b \quad \lambda_3 x_3^b - x_0^b \quad x_0^b] \tag{21}$$

Using the fifth point, we have

$$\lambda_4 x_4^b = Q E_4 = \lambda_1 x_1^b + \lambda_3 x_3^b - x_0^b, \tag{22}$$

and components of x_4^b are

$$u_4^b = \frac{\lambda_1 u_1^b + \lambda_3 u_3^b - u_0^b}{\lambda_1 + \lambda_3 - 1} \quad (23)$$

$$v_4^b = \frac{\lambda_1 v_1^b + \lambda_3 v_3^b - v_0^b}{\lambda_1 + \lambda_3 - 1} \quad (24)$$

Using these two equations, we find the following equation to compute λ_1 and λ_3:

$$\begin{bmatrix} u_4^b - u_1^b & u_4^b - u_3^b \\ v_4^b - v_1^b & v_4^b - v_3^b \end{bmatrix} \begin{bmatrix} \lambda_1 \\ \lambda_3 \end{bmatrix} = \begin{bmatrix} u_4^b - u_0^b \\ v_4^b - v_0^b \end{bmatrix}. \quad (25)$$

Now we compute the last scale factor λ_2 using Proposition 1. Since we know λ_1 and λ_3, the three vectors q_1, q_2 and q_3 may be written as:

$$H = [\, a_1 \;\; a_2 \;\; a_3 \,] = \begin{bmatrix} q_1^T \\ q_2^T \\ q_3^T \end{bmatrix} = \begin{bmatrix} a & \lambda_2 u_2^b & d \\ b & \lambda_2 v_2^b - u_0^b & e \\ c & \lambda_2 - 1 & f \end{bmatrix} \quad (26)$$

From Equation (7), we have a quadric equation for λ_2:

$$A\lambda_2^2 + B\lambda_2 + C = 0, \quad (27)$$

where A, B and C are the coefficients composed of the elements of the matrix H

$$A = (u_2^b - d)(v_2^b f - e) + (u_2^b c - a)(v_2^b - b), \quad (28)$$

$$B = (u_2^b f - d)(e - f v_0^b) + (v_2^b f - e)(d - f u_0^b) + (u_2^b c - a)(b - c v_0^b) + (v_2^b c - b)(a - c u_0^b) \quad (29)$$

$$C = (e - f v_c^b)(d - f u_0^b + (b - c v_0^b)(a - c u_0^b) + (dc - af)(ec - bf). \quad (30)$$

Given two solutions of the equation, we choose the larger positive one. This choice is due to our implementation of the graphics program.

Note that all the λ_k's should be positive, which indicates that one should be careful in choosing the control points because the values of λ_k's are determined by the image locations of the vertices chosen in the process of embedding. Also there is a possibility of obtaining no real solutions for Equation (27). However, such a case does not occur in practice unless the control points make a very odd configuration. If an image point of a vertex has a negative λ it means geometrically that the point is behind the retinal plane of the virtual camera. In this case, although the point is projected to the desired location algebraically, we can not obtain correct views through the graphics library.

4.3 Projection by the Virtual Camera

During the procedure of embedding, we specify some vertices of the world coordinate frame. To facilitate this procedure, we need to see the results of our embedding. Given a virtual camera Q we compute the image locations $x_i^b \approx QE_i$ of the vertices $\{E_i\}_{i=5,6,7}$ and examine our embedding result. An example of the result of embedding is shown in the bottom half of Figure 6.

4.4 Virtual Camera Decomposition

To use the virtual camera matrix in graphics, we should decompose it into three parts K, R and t as shown in equation (2). Since we know from Equation (2) that

$$HH^T = \rho^2 KK^T, \quad (31)$$

we first compute the scale factor $\rho = \| q_3 \|$ of the matrix Q, and then K is computed using Cholesky decomposition [32]. Finally, R and t are computed using equations (5) and (6).

4.5 Embedding Graphics Objects

The location and pose of a 3D graphics object are defined with respect to the world coordinate system which is now embedded in camera coordinate system. In this way, the characteristics of the graphics objects, such as color, specularity of surfaces, etc, can be defined as usual graphics modeling does. The light sources can also be configured with respect to the embedded world coordinate system.

5. Rendering

Rendering for k-th video image consists of two steps:

1. Determination of the k-th virtual camera.
2. Generation of corresponding view of graphic objects and overlaying it on the video image.

5.1 Transfer of Virtual Camera

When k-th video image is entered, corresponding virtual camera Q_k is computed as follows.

1. Find image matches $\{x_i\}_{i=1}^{N}$ in the k-th video image.
2. Compute the projective camera matrix P_k using the recovered 3D projective coordinates $\{X_i\}_{i=1}^{N}$ of the image matches. Equation (10) will give P_k.
3. Project the projective 3D basis points $\{X_i^b\}_{i=0}^{4}$ into the k-th video image by P_k to compute $\{x_{ki}^b\}_{i=0}^{4}$:

$$x_{ki}^b \approx P_k X_i^b, \quad \text{for } i=0, \ldots, 4 \tag{32}$$

4. Compute the k-th virtual camera Q_k using the image points $\{x_{ki}^b\}_{i=0}^{4}$ through the procedure of Section 4.2.
5. Decompose Q_k into K_k, R_k and t_k as in Section 4.4.

Note that the components, K_k, R_k and t_k, of the virtual camera varies during the sequence as the real camera moves in space and changes its internal parameters like focal length or zoom.

5.2 Graphics Rendering

We decomposed Q_k into three parts

$$Q_k = \rho_k K [\, R_k \mid t_k \,]. \tag{33}$$

Now, the scale factor ρ_k is no longer useful in graphics rendering. Figure 7shows the three stages of

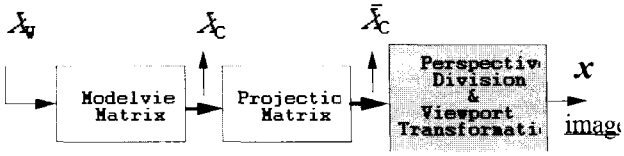

Fig. 7. Stages of coordinate transformation in a graphics system

coordinate transformation in a graphics machine for making a view of a graphics object through the virtual camera Q_k. The rotation R_k and translation t_k define the modeling transformation and Q_k gives the perspective viewing volume. The modelview matrix M_k in Figure 7is defined by the rotation R_k and t_k as follows:

$$M_k = \begin{bmatrix} R_k & t_k \\ 0_3^{\mathrm{T}} & 1 \end{bmatrix}. \tag{34}$$

This transformation converts a 3D point X_w defined with respect to the world coordinate system O_c to the point X_c with respect to the camera coordinate system O_c. After that, a projection matrix is applied to yield clip coordinates $\bar{X}_c$. The internal calibration matrix K_k determines this matrix and defines the corresponding viewing volume. Any part of the graphic objects outside this volume are clipped so that they are not drawn in the final scene. Finally, the perspective division and viewport transformation are performed to yield the final graphic image[33]. The generated view is then overlayed on the corresponding video image i_k, which gives the effect of video augmentation.

6. Experiments

We implemented and tested our method. Figure 8 shows a result of video augmentation. This video sequence was captured by a hand-held video camera and composed of 274 frames in which 32 corner points of the rectangles were tracked to estimate the projective motion of the video camera. We removed the effect of motion blur in the measurement of corner points by

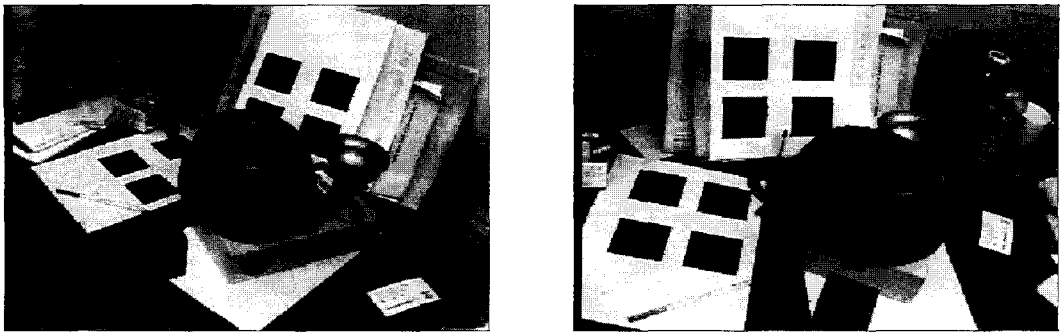

Fig. 8. A result of video augmentation. The 1st and the 200-th augmented images are shown in which a teapot, a cup and a cube were inserted. Notice the specularity of the teapot and the cup, as well as the shading effect on the objects.

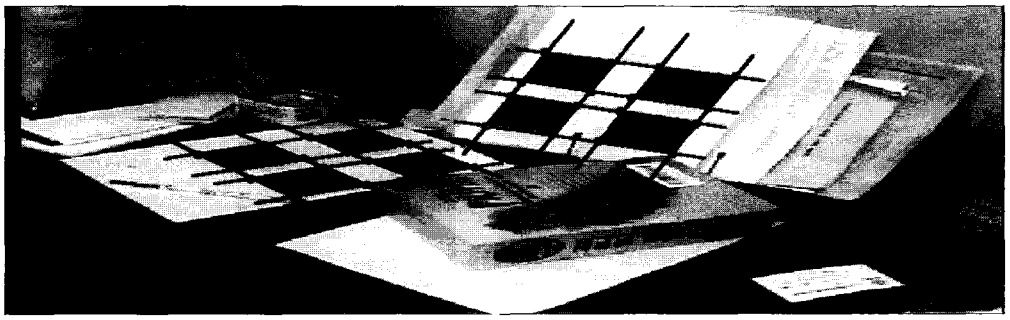

Fig. 9. Lines tracked and their intersection points. A total of 32 intersection points are detected in each of the 546 fields

dividing each image into two even and odd fields. Thus, we processed 546 fields to track the lines of the rectangles. Corner points were found by computing the intersections of the tracked lines.

Figure 9 shows tracked lines and their intersection points. The size of each field was 720 × 243. The intersection points were utilized in the estimation of the projective motion estimation. First, two fields were selected as the control images as shown in Figure 6, and then the fundamental matrix were computed using the algorithm of [30]. Projective camera matrices for the two views were then computed, the 3D projective coordinates were computed for the 32 intersection points, and finally the projective camera matrices for the whole fields were estimated by the reprojection method in the sequence, as explained in Section 2.1 Figure 10shows the performance of our projective motion estimation using the tracked intersection points. It depicts the graphs of the maximum and RMS of reprojection errors $E_{ki} = \left\| x_i - \frac{P_k X_i}{P_k^3 X_i} \right\|_2$, where x_i is the i-th intersection point measured in the k-th field, P_k^3 is the third row of P_k, and $\| \cdot \|_2$ is the Euclidean norm. The maximum reprojection error was below a 1.4 pixel error during the sequence, implying that the intersection points were detected accurately. The values of the skew

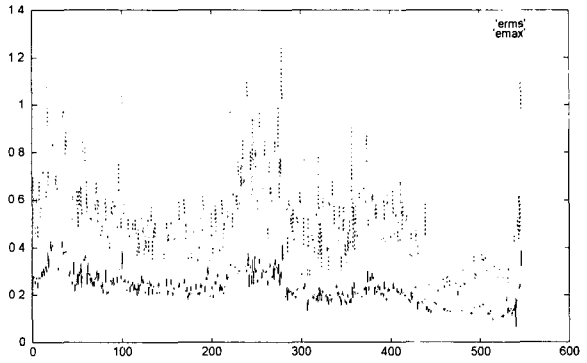

Fig. 10. Evaluation of our projective motion estimation. Maximum and RMS reprojection errors are plotted. The axis of abscissa denotes the field index of the video images and the axis of ordinate the magnitude of errors in the unit of pixel.

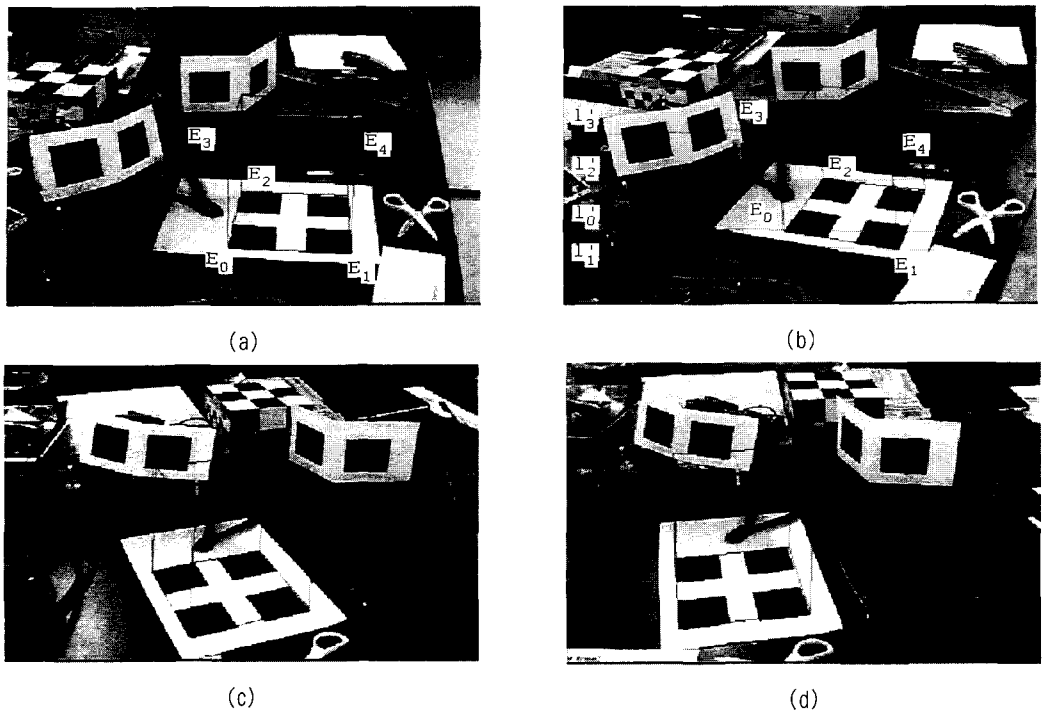

Fig. 11. Embedding for another video sequence. (a) and (b) show two control images on which five vertices and four epipolar lines are illustrated. (c) and (d) give the view of the unit-cube of the world coordinate system at different view points.

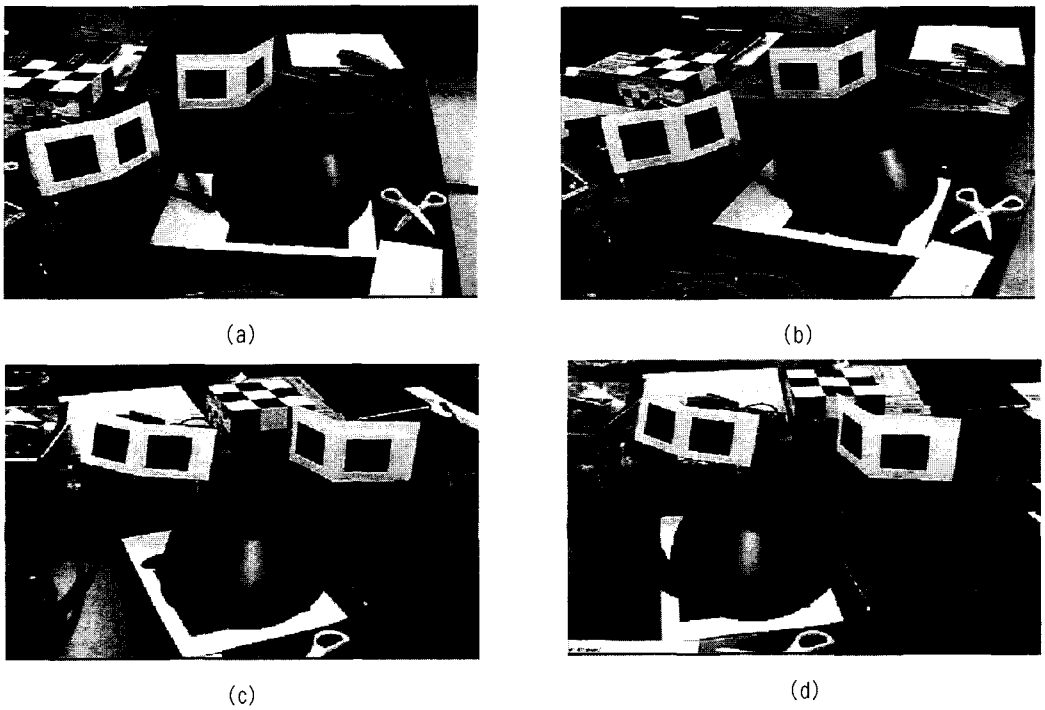

Fig. 12. Another video augmentation result. With respect to the embedded world coordinate system, shown in Figure 11, we augmented the real world with two virtual light sources and one red teapot. (a) and (b) correspond to the control images. (c) and (d) are 300-th augmented images, respectively.

components of the computed virtual camera matrices were below 10^{-12}, which means that the computation of the virtual camera matrices with transfer operation by projective motion and structure was very accurate in the sense that the skews were almost zero. We used an SGI graphics machine and a C-encoded 3D graphics program with the OpenGL library [33]. We defined a light source and three graphics objects with respect to the world coordinate system whose material properties were different from each other. Due to our system limit, the results were obtained through an off-line process.

Figures 11 and 12show another experiment. The video sequence was composed of 400 frames. We selected 0-th and 100-th video images as the two control images in which the locations of the five vertices of the world coordiate system were interactively chosen. Figure 11 (a) and (b) show the two control images, five vertices chosen, and four epipolar lines drawn for embedding. Figure 11(c) and (d) show the 300-th and 399-th images, on which the unit cube of the world coordinate system embedded was drawn. Figure 12 shows four augmented video images. They correspond to (a) 0-th, (b) 100-th, (c) 300-th, and (d) 399-th images, respectively. In this experiment, we used two light sources and one red teapot for the virtual world. [1]

7. Real Camera vs. Virtual Camera

Now, let us consider the relationship between the real camera and virtual camera. As explained in the previous sections, a virtual camera is defined and computed using the *manually specified* image coordinates of the vertices $\{E_i\}$ of the world coordinate system, while their projective coordinates $\{X_i^b\}$ are computed using the projective camera matrices $\{P_k\}$ of the *real camera*. We have the following relationship from the definition of our virtual camera:

$$x_i^b \approx QE_i. \tag{35}$$

The projective coordinates of the vertices are computed using the following equation:

$$x_i^b \approx PX_i^b. \tag{35}$$

We know that there exists an unknown 3D projective transformation $T_{4\times4}^P$ as mentioned in Section 2.1:

$$P_k\, T_{4\times4}^P \approx P_k^{\varepsilon}. \tag{37}$$

Let us suppose that we know $T_{4\times4}^P$. Then we have

$$\begin{aligned} x_i^b &\approx PX_i^b && (38)\\ &\approx PT_{4\times4}^P\, {T_{4\times4}^P}^{-1} X_i^b && (39)\\ &\approx P^{\varepsilon} X_i^{b^{\varepsilon}} && (40) \end{aligned}$$

where $P^{\varepsilon} = PT_{4\times4}^P$ and $X_i^{b^{\varepsilon}} = (T_{4\times4}^P)^{-1} X_i^b$. The 3D Euclidean coordinate $X_i^{b^{\varepsilon}}$ is a point represented in a Euclidean coordinate system of the real world. Then, *do the coordinates* $\{X_i^{b^{\varepsilon}}\}$ *constitute a rectangular coordinate system like* $\{E_i\}$ *as shown in Figure 5?* The answer is no. It is clear that there exists a non-singular transformation T between $\{X_i^{b^{\varepsilon}}\}$ and $\{E_i\}$:

$$E_i \approx TX_i^{b^{\varepsilon}}. \tag{41}$$

However, the transformation T cannot be confined to a group of Euclidean transformations because it is determined implicitly through the procedure of embedding, which does not guarantee the ortho-normality of the coordinate system composed of $\{X_i^{b^{\varepsilon}}\}$. Hence, the transformation is included in a more general class, the group of projective transformations, even though we cannot remove the possibility of the transformation being a Euclidean transformation by manual specification.

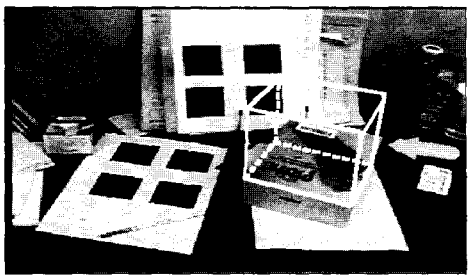

Fig. 13. Implicit projective distortion. The view of a rectangular cube through a virtual camera shows a different perspectivity compared to the views of real objects. In particular, the face indicated by arrow would not be seen if the rectangular cube was a real object.

Practical implication of the 3D projective transformation T results in a projectively distorted graphics view. Figure 13shows an example. We tried to place a virtual rectangular cube on the real box so that two edges of the cube coincided with those of the real box. If the cube had been a real object, the face indicated by the arrow, which was caused by our embedding, would not have been seen in the image.

8. Conclusion

We proposed a method for augmenting real video images with computer generated perspective views of graphic objects without explicit metric calibration of the video camera. There are two major steps: embedding and rendering. In order to embed the world coordinate system with respect to the camera coordinate system, we specified five vertices of the world coordinate frame in the first control video image and four in the second. Using the specified locations of the vertices, we defined a virtual camera attached to the video camera, and obtained effective motion of the virtual camera being the same as that of the real video camera. In the rendering procedure, we decomposed the virtual camera into three components: rotation, translation and calibration. This enabled us to make full use of the functions of 3D computer graphics for generating views of graphics model objects.

Our method is based on a projective reconstruction algorithm. Without calibration of the video camera, the recovery of the projective motion and structure of the real world provided us with the ability to define the virtual camera attached to the real camera and move the virtual camera according to the motion of the real camera. Also, the virtual camera followed the change of the internal parameters of the real camera. The projective distortion explained in Section 7 could be a limitation of our algorithm, even though we could make the augmented videos seem real. Real-time implementation is another issue, for which detecting and tracking feature points, sequential update of projective motion and rendering graphics objects should be done in real-time. Our method can be modified easily for use in a real-time system. We hope this research provides a link between computer vision and computer graphics.

참 고 문 헌

[1] R. T. Azuma, "A survey of augmented reality," *PRESENCE: Teleoperations and Virtual Environments*, vol. 6, pp. 355-385, Aug. 1997.

[2] M. Bajura and U. Neumann, "Dynamic registration correction in video-based augmented reality systems," *Computer Graphics and Applications*, pp. 52-60, 1995.

[3] W. Grimson, G. Ettinger, S. White, P. Gleason, T. Lozano-Perez, W. Wells, and R. Kikinis, "Evaluating and validating an automated registration system for enhanced reality visualization in srugery," in *Computer Vision, Virtual Reality and Robotics in Medicine '95(CVRMed'95)*, pp. 3-12, Apr. 1995.

[4] M. Tuceryan, D. Greer, R. Whitaker, D. Breen, C. Crampton, E. Rose, and K. Ahlers, "Calibration requirements and procedures for a monitor-based augmented reality system," *IEEE Transactions on Visualization and Computer Graphics*, vol. 1, pp. 255-273, Sep. 1995.

[5] R. Sharma and J. Molineros, "Computer vision-based AR for guiding manual assembly," *Presence: Teleoperators and Virtual Environments*, vol. 6, pp. 292-317, Jun. 1997.

[6] M. O. Berger, G. Simon, S. Petitjean, and B. Wrobel-Dautcourt, "Mixing synthesis and video images of outdoor environments: Application to the bridges of paris," in *ICPR'96*, pp. 90-94, 1996.

[7] H. Schumann, S. Burtescu, and F. Siering, "Applying augmented reality techniques in the field of interactive collaborative design," in *SMILE Workshop on 3D structure from multiple images of large-scale environments, in conjunction with ECCV'98*, Jun. 1998.

[8] M. O. Berger, "Resolving occlusion in augmented reality: a contour based approach without 3D recon-

struction," in *CVPR'97*, 1997.

[9] S. Grosskopf and P. Neugebauer, "The use of reality models in augmented reality application," in *SMILE Workshop on 3D s*ructure from multiple images of large-scale environments, in conjunction iwth ECCV'98*, Jun. 1998.

[10] A. Heyden and K. Astrom, "Euclidean reconstruction from image sequences with varying and unknown focal length and principal point," in *Proc. CVPR'97*, 1997.

[11] A. Heyden and K. Astrom, "Minimal conditions on intrinsic parameters for euclidean reconstruction," in *Proc. Asian Conference on Computer Vision, Hong Kong*, 1998.

[12] M. Pollefeys and L. V. Gool, "Self-calibration and metric reconstruction in spite of varying and unknown internal camera parameters," in *Proc. Int. Conf. on Computer Vision*, 1998.

[13] K. N. Kutulakos and J. Vallino, "Calibration-free augmented reality," *IEEE Trans. Visualization and Computer Graphics*, vol. 4, no. 1, pp. 1-20, 1998.

[14] O. Faugeras, "From geometry to variational calculus: theory and applications of threedimensional vision," in *IEEE and ATR Workshop on Computer Vision for Virtual Reality Based Human Communications*, Jan. 1998.

[15] R. Cipolla and N. Hollinghurst, "Human-robot interface by pointing with uncalibrated stereo vision," *Image and Vision Computing*, vol. 14, no. 3, pp. 171-178, 1996.

[16] G. Hager, "Calibration-free visual control using projective invariance," in *Fifth Inter. Conf. Comp. Vision*, Jun. 1995.

[17] C. Zeller and O. Faugeras, "Applications of non-metric vision to some visual guided tasks," in *12th IAPR International Conference on Pattern Recongnition*, Oct. 9-13 1994.

[18] P. Beardsley, I. Reid, A. Zisserman, and D. Murray, "Active visual navigation using nonmetric structure," in *Fifth Inter. Conf. Comp. Vision*, 1995.

[19] L. Robert, M. Buffa, and M. Hebert, "Weakly-calibrated stereo perception for rover navigation," in *Fifth Inter. Conf. Comp. Vision*, Jun. 1995.

[20] D. Forsy, J. Mundy, A. Zisserman, and C. Rothwell, "Using global consistency to recognise euclidean objects with an uncalibrated camera," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, Jun. 1994.

[21] A. Shashua, "Projective depth: A Geometric invariant for 3D reconstruction from two perspective/ orthographic views and for visual recognition," in *Fourth Inter. Conf. Comp. Vision*, 1993.

[22] O. Faugeras, "What can be seen in three dimensions with an uncalibrated stereo rig?," in *ECCV'92*, 1992.

[23] R. I. Hartley, R. Gupta, and T. Chang, "Stereo from uncalibated cameras," in *CVPR'92*, 1992.

[24] Q. T. Luong and T. Vieville, "Canonic representations for the geometries of multiples projective views," in *Lecture Notes in Computer Science, Vol. 800 Jan-Olof Eklundh(Ed.) Computer Vision-ECCV'94*, Springer-Verlag, 1994.

[25] R. I. Hartley, "A linear method for reconstruction from lines and points," in *Fifth Inter. Conf. Comp. Vision*, 1995.

[26] A. Heyden, "Reconstruction from image sequences by means of relative depths," in *Fifth Inter. Conf. Comp. Vision*, Jun. 1995.

[27] B. Triggs, "Factorization methods for projective structure and motion," in *Int. Conf. Computer Vision & Pattern Recognition*, (San Francisco), 1996.

[28] O. Faugeras, *Three Dimensional Computer Vision: A Geometric Approach. MIT Press, 1993.*

[29] O. Faugeras, "Stratification of three-dimensional vision: Projective, affine, and metric representations," *J. Opt. Soc. Am. A*, vol. 12, pp. 465-484, Mar. 1995.

[30] Z. Zhang, "Determining the epipolar geometry and its uncertainty: A review," Tech. Rep. 2079, INRIA, France, Jul. 1996.

[31] B. Triggs, "Autocalibration and the absolute quadric," in *Int. Conf. Computer Vision & Pattern Recognition, 1997.*

[32] W. Press, S. Teukolsky, W. Vetterling, and B. Flannery, Numerical Recipes in C: *The Art of Scientiic Computing*. Cambridge University Press, 1992.

[33] OpenGL ARB, *OpenGL Programming Guide*. Addison Wesley, 1993.

저 자 소 개

Yongduek Seo

Yongduek Seo received the B.S. degree in electronic engineering from Kyungpook National University, in 1992, and the M.S. degree in electronic and electrical engineering from Pohang University of Science and Technology (POSTECH), in 1994. He started PhD studies in electronic and electrical engineering at POSTECH, in 1994, and anticipates completion in Spring 2000. During April-May 1999 he was a guest researcher in Mathematical Imaging Group, Lund Institute of Technology, Sweden. His interests include real-time computer vision, camera self-calibration, structure recovery from motion, and augmented reality.

Ki-Sang Hong

Ki-Sang Hong received the B.S. degree in electronic engineering from Seoul National University, Korea, in 1977, and the M.S. and Ph.D degrees in electrical and electronic engineering from KAIST (Korea Advanced Institute of Science and Technology), in 1979 and 1984, respectively. During 1984-1986, he was a researcher in the Korea Atomic Energy Research Institute. In 1986, he joined POSTECH (Pohang University of Science and Technology), Korea, where he is currently an associate professor of electronic and electrical engineering. During 1988-1989, he worked in the Robotics Institute at Carnegie Mellon University, Pittsburgh,USA, as a visiting professor. His current research interests include computer vision and pattern recognition.