Calibration-Free Augmented Reality in Perspective

Yongduek Seo and Ki Sang Hong, Member, IEEE

Abstract—This paper deals with video-based augmented reality and proposes an algorithm for augmenting a real video sequence with views of graphics objects without metric calibration of the video camera by representing the motion of the video camera in projective space. A virtual camera, by which views of graphics objects are generated, is attached to a 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 the 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 the theoretical and experimental results of our application of nonmetric vision to augmented reality.

Index Terms—Augmented reality, Euclidean geometry, virtual camera, perspective projection, projective geometry, calibration-free method.

1 INTRODUCTION

This paper deals with video-based augmented reality or enhanced reality whose major subject is making views of a 3D graphic model and mixing them into a video in real time [1], [2]. The virtual objects used in augmented reality are generally 3D graphic 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 [3], [4], assembly, maintenance and repair [5], [6], simulating light conditions of an outdoor building [7], etc. All of these systems enrich reality or real video with computer-generated views of graphical objects, cooperating with computer vision techniques: estimating the camera parameters in real-time [8], [9], [10], [11], resolving occlusion between a virtual object and a real object [12], [13], and correcting the registered location of a graphic object dynamically [3].

Camera calibration is a prerequisite for embedding virtual objects into video frames because the geometric relationship among physical objects, virtual objects, and the camera needs to be established to obtain correct views of the model object. Fig. 1 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^C_V denotes the Euclidean displacement between two coordinate systems O_C and O_V. The transformation between real camera O_C and virtual camera O_V is represented by the nonsingular 3D projective transformation T^V_C and the transformation between O_V and O_V by T^V_V. For physically correct video augmentation, all these coordinate transformations should be known a priori and the usual transformation T^V_C 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_V 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 by the graphics camera for generating graphics views. 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 such cases, some reference points or patterns whose Euclidean geometry is known must be seen in the video [8], [9], [10], [11]. 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 [14], [15], [16] or a calibration-free augmented reality system like [17]. For example, Faugeras applied a self-calibration method for fixed internal parameters to the augmentation of real video sequences [18]. Self-calibration provides not only calibration parameters but also the Euclidean motion of the real camera. However, the convergence issue of the methods due to a nonlinear formulation may constrain practical applications and more research is needed for real-time implementation. Seo et al. inserted a real object into a perspective video sequence using projective motion and image-based rendering technique [19]. Chen et al. utilized a user-specified cuboid structure for augmenting a single image with computer-generated graphics [20]. We are informed that the work of Chen et al. has been done independently of this paper.

The work of Kutulakos and Vallino [17] was innovative in 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, the method 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 an explicit Euclidean camera calibration procedure. Indeed, much research on nonmetric vision applications uses not Euclidean but projective or affine information: human and robot interface and interactive grasping with uncalibrated stereo vision [21], visual control or visual guided tasks [22], [23], navigation [24], [25], object recognition [26], etc. This paper applies the functionalities of nonmetric 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 is specified in two selected video images in order to insert the virtual world into the real world (embedding). This establishes the bridge between the real and virtual world. The view of the world coordinate system in a video image provides the corresponding virtual camera matrix, an imaginary camera attached to the real camera, through which the graphics views are generated. Embedding is accomplished by specifying five basis points, or unit vertices, of the world coordinate system shown in Fig. 6, in the first selected video image and four in the second. The relationship of the world coordinate frame to the virtual camera, as well as its computation algorithm, is 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 $T_{w}^{v}$ in Fig. 1. Accordingly, the relationship $T_{c}^{v}$ between the virtual camera and the real camera is, in our case, 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 library (rendering). Thus, we can generate perspective views of graphic objects, as presented in Section 5. The motion and change of the internal parameters, like focal length of the real video camera, are represented, in this paper, by projective camera matrices. The information of the real camera given in projective space induces the external Euclidean motion and internal parameter changes of the virtual camera. Hence, the virtual information comes to emulate the information of the real camera. Projective structure and motion is a formulation of structure from motion algorithm by projective geometry, applied when the internal calibration information about the camera is unknown and only image matches like points or lines are given [27], [28], [29]. Projective structure and motion can be obtained by first computing the fundamental matrix or the multilinear tensor like a trifocal tensor followed by estimating the structure and motion [30], [31], [32], [25], [23]. Possible approaches for multiple views are the factorization method for projective structure [33], [34] or the framework of affinely or projectively reduced setting [32]. The reconstruction is up to an unknown three-dimensional projective transformation and we briefly review the theory of projective reconstruction in Section 2.1. This paper uses a two view algorithm for a real-time application like the system of Kutulakos and Vallino [17]. 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 in 3D space and change its internal parameters according to the change of rotation, translation.

![Image](https://example.com/image.png)

**Fig. 1.** The relationship of various coordinate systems.
and internal parameters of the real video camera. Details are given in Section 5.1.

The embedding procedure is similar to the method of Kutulakos and Vallino [17], 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 defines the virtual camera of the perspective projection model. In contrast, they specified four locations in the first and the second images. The camera model is an orthographic camera which cannot 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 to 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 to enable us to specify lighting and material properties to make shaded objects and their shadows as well.

Section 2 presents 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 in 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 the real camera and the virtual camera and deals with the problem of different perspectives, which is inherent in our application of nonmetric 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, making 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:

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

The \( k \)th video image is represented by \( I_k \). The \( 3 \times 4 \) camera matrix obtained by projective reconstruction from real video images is denoted by \( P_k \).

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

\[
Q = \rho K[R|t]
\]

\[
= [H|h],
\]

where \( \rho \) is a nonzero scale, \( R \) and \( t \) are \( 3 \times 3 \) rotation matrix, and 3D translation vector, respectively, and \( K \) is a \( 3 \times 3 \) calibration matrix of the form:

\[
K = \begin{bmatrix}
\alpha & 0 & u_0 \\
0 & \beta & v_0 \\
0 & 0 & 1
\end{bmatrix}.
\]

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

\[
H = \rho KR,
\]

\[
h = \rho Kt.
\]

A zero-skew camera satisfies the following proposition [35]:

**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,

\[
(q_1 \times q_3) \cdot (q_2 \times q_3) = 0,
\]
where $\times$ and $\cdot$ are the outer product and the inner product of three-dimensional vectors, respectively.

Notice that (7) satisfies regardless of the scale factor $\rho$. The proof is not difficult and this proposition is useful in the computation of the matrix of a virtual camera as presented 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. In this section, we briefly review the concepts of projective motion and structure estimation that have been developed in computer vision. Details on theory and estimation methods may be found in literature on projective reconstruction or image tensors such as [28], [29], [27], [36], [30], [32], [23].

Given matching points between two views, the fundamental matrix $F$ can be computed from the equation

$$x^TFx = 0,$$

where $x$ is an image point from one view and $x'$ is from the other view. Fig. 2 illustrates the epipolar geometry between two views. The epipole is the image location of the focal point of other cameras, that is, the epipole $e$ ($e'$) is the image location of $C'$ ($C$), the focal point of the second (first) camera in the first (second) image. Equation (8) induces a line coordinate vector $l'_x$ with respect to $x$ given by

$$l'_x = Fx$$

since it satisfies the line equation $x^Tl'_x = 0$. The line $l'_x$ is called the epipolar line of the point $x$. Geometrically, epipolar line $l'_x$ is the projection of the ray $\overline{CX}$ onto the image $I'$ by the focal point $C'$. Notice that the corresponding point $x'$ is on its corresponding epipolar line in Fig. 2. Notice also that every epipolar line passes through the epipole.

Given the fundamental matrix and the epipole $e'$, two projective camera matrices are chosen as

$$P = [I|0], \quad P' = [\hat{e}'|F\hat{e}'],$$

where $e'$ is the epipole in the second image and $[e']_x$ is the $3 \times 3$ skew matrix such that $[e']_x x = e' \times x$. The epipole $e'$ is given as the left null of $F$: $F^T\hat{e}' = 0$.

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

$$x_{1i} \approx P_1X_i, \quad x_{2i} \approx P_2X_i,$$

where the notation $\approx$ denotes equality up to scale. That is, if two nonzero vectors or matrices have the relationship $u \approx v$, then there is a nonzero scale $\lambda$ such that $u = \lambda v$. Solving a linear least-square equation gives $X_i$.

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

$$x_{ki} \approx P_kX_i, \quad \text{for } i = 1, \ldots, N,$$

or, more specifically,

$$u_{ki} = \frac{P_{1i}X_i}{P_{2i}X_i}, \quad v_{ki} = \frac{P_{3i}X_i}{P_{2i}X_i}, \quad \text{for } i = 1, \ldots, N,$$

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

The projective structure $X_i$ and motion $P_k$ can be taken as simply a 4D vector and a $3 \times 4$ matrix yielding the $i$th input image coordinate $x_{ki}$ in the $k$th image:

$$x_{ki} \approx P_kX_i.$$  

This reconstruction is up to a nonsingular three-dimensional projective transformation $T$ of size $4 \times 4$ since $P'_k = P_kT$ and $X'_i = T^{-1}X_i$ yield the same result, which underlies the specialized term projective structure and motion.

The relationship between the Euclidean motion and projective motion of the camera is described by the projective transformation $T_{4 \times 4}$, which upgrades the projective camera matrix to Euclidean camera matrix:

$$P_kT_{4 \times 4} \approx P_k^e$$

Fig. 2. Epipolar geometry between two views. Geometrically, the epipole $e'$ is the projection of $C$ onto the image $I'$ and the epipolar line $l'_x$ is the projection of the ray $\overline{CX}$. 

\[V\]
for all $k$, where $P^e_k$ is the true camera matrix that can be decomposed into its calibration part $K^e_k$ and Euclidean motion parts $R^e_k$, rotation, and $t^e_k$, translation:

$$P^e_k = K^e_k[R^e_k|t^e_k].$$

Note that $K^e_k$ is an upper triangular matrix representing the internal calibration parameters of the real camera, e.g., focal length, and $R^e_k$ is the relative direction and $t^e_k$ is the relative location of the camera coordinate system with respect to the world coordinate system. As soon as the transformation $T_{4 \times 4}$ is known, the projective reconstruction may be upgraded to a Euclidean version. Estimating the projective transformation $T_{4 \times 4}$ has been a subject of self-calibration and can be found in [14], [15], [16], [37]. Note that the Euclidean camera matrices $P^e_k$ can be directly used for graphics rendering using a graphics machine, but projective camera matrices cannot be used because they are not of the form of (16), which is the reason we define a virtual camera. Section 7 explains the relationship between the real camera and virtual camera.

### 2.2 Projective Plane Homography

Let us suppose that four points $\{X_i\}_{i=1}^4$ are on the same plane $\Pi$ and their image points in two different views are $\{x_i\}_{i=1}^4$ and $\{x'_i\}_{i=1}^4$. Then, there exists a $3 \times 3$ projective plane homography $T_{\Pi}$ such that

$$x'_i \approx T_{\Pi}x_i,$$

as illustrated in Fig. 3. We can compute $T_{\Pi}$ using at least 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 (17) [36].

### 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. The steps of tracking image corner points and estimating the projective motion of the video camera are included in the embedding part. First, the embedding steps, graphically shown in Fig. 4, are as follows:

1. Track image corner points in the video sequence. Fig. 10 shows an example of the tracked image corner points.
2. Choose two control images onto which the graphics world-coordinate frame will be inserted. The two images may be from the two 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 $\{X_i\}_N$ for the two images using point correspondences, as described in Section 2.1.
4. Specify the locations $\{(x'_i, x'_i)\}$ of the basis points of the world coordinate frame in each of the control images, five in the first and four in the second.

![Fig. 3. Plane homography $T_{\Pi} : \pi \rightarrow \pi'$](image)

![Fig. 4. Overview of our method: embedding procedure.](image)
5. Compute the 3D projective coordinates $X_i^b$, $i = 0, \ldots, 4$, of the five specified basis points $\{(x^i, x^i')\}$ using $P$ and $P'$.

Second, the rendering steps, shown in Fig. 5, 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 $x_i^k \approx P_k X_i^b$, $i = 0, \ldots, 4$.

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

8. Decompose $Q_k$: $Q_k = \rho_k K_k[R_k|t_k]$.

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, such as 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 step is to find matches of image corner 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 (10). Projective 3D coordinates $\{X_i\}_{i=1}^N$ of the image matches are computed using (11).

Video augmentation is accomplished by first specifying the graphic world-coordinate system into the control images denoted by $\mathcal{V}$ and $\mathcal{V}'$. For example, Fig. 7 shows two control images, $I_0$ and $I_{270}$, from 274 video images used in our experiments.

The embedding procedure consists of two steps:

1. We insert the world coordinate system of the graphics frame into the first control image $\mathcal{V}$ 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 $\mathcal{V}'$.

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 Fig. 6, 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 soon.

4.1 Embedding via Epipolar Geometry

First, five locations $\{x_i\}_{i=0}^4$ of the vertices are specified in the first control image $\mathcal{V}$. This determines the appearance of the world coordinate frame in the video image. Then, we have five epipolar lines in the second image $\mathcal{V}'$ using the fundamental matrix between the two control images.
from (8). Now, we choose four image locations $f_{xxb_i} = g_i \hat{e}_0$ of the vertices on the corresponding epipolar lines $l_i = Fx_i, \quad i = 0, \ldots, 3$. However, we do not need to specify the last one, $f_{xxb_4}$, because it can be determined by the plane homography $T_{xxb_0}$, which is determined by the equations $x_i = T_{xxb_0}x_i, \quad i = 0, 1, 3$ and $e' \approx T_{xxb_0}e$, as given in Section 2.1.

Finally, we compute the projective 3D coordinates $f_{XXb_i} = g_{i\hat{e}_0}$ 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.

Fig. 7 gives an example where five image locations in the first control image $V$ and four in the second $V_0$ 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 procedure, 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 perceives the embedded world coordinate frame.

Our goal is to have the matrix of the virtual camera generate a view of the world coordinate frame that matches what we figured in the control images in the embedding step. Let us denote a virtual camera $Q$ by a $3 \times 4$ matrix

$$Q = [a_1, a_2, a_3, a_4],$$

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

$$\lambda_i x_i = Q E_i,$$

we have

$$a_4 = \lambda_0 x_0,$$

$$a_i = \lambda_i x_i - \lambda_0 x_0, \quad \text{for } i \in \{1, 2, 3\}.$$  

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

$$Q = [\lambda_1 x_1 - x_0, \lambda_2 x_2 - x_0, \lambda_3 x_3 - x_0, x_0].$$

Using the fifth point, we have

$$\lambda_4 x_4 = QE_4 = \lambda_1 x_1 + \lambda_3 x_3 - x_0,$$

and components of $x_4$ are

$$u_4 = \frac{\lambda_1 u_1 + \lambda_3 u_3 - u_0}{\lambda_1 + \lambda_3 - 1}.$$
Using these two equations, we find the following equation to compute $\lambda_1$ and $\lambda_3$:

$$\begin{bmatrix} a & b & c \\ b & d & e \\ c & e & f \end{bmatrix} \begin{bmatrix} x^b_i \\ y^b_i \\ z^b_i \end{bmatrix} = 0, \quad (29)$$

Next, we compute the last scale factor $\lambda_2$ using Proposition 1. Since we know $\lambda_1$ and $\lambda_3$, the three vectors $q_1, q_2, q_3$ may be written as:

$$H = [a \quad b \quad c] = \begin{bmatrix} q^T_1 \\ q^T_2 \\ q^T_3 \end{bmatrix} = \begin{bmatrix} a & b & c \\ \lambda_2 & \lambda_2 & \lambda_3 \\ \lambda_2 & \lambda_3 & \lambda_3 \end{bmatrix} \begin{bmatrix} x^b_0 \\ y^b_0 \\ z^b_0 \end{bmatrix}. \quad (28)$$

From (7), we can derive a quadric equation for $\lambda_2$:

$$AX^2 + BX + C = 0, \quad (26)$$

where $A, B, C$ are the coefficients of the elements of the matrix $H$:

$$A = (u^2 e - d)(v^2 f - e) + (u^2 c - a)(v^2 c - b), \quad (30)$$

$$B = (u^2 e - d)(e - f u^2_0) + (v^2 f - e)(d - f u^2_0) + (u^2 c - a)(b - c u^2_0) + (v^2 c - b)(a - c u^2_0), \quad (31)$$

$$C = (e - f u^2_0)(d - f u^2_0) + (b - c u^2_0)(a - c u^2_0) + (d - e f)(e c - b f). \quad (32)$$

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 of the $\lambda_3$s should be positive, which indicates that one should be careful in choosing the control points because the values of $\lambda_3$s are determined by the image locations of the vertices chosen in the process of embedding. Further, there is a possibility of obtaining no real solutions for (29). 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 $\lambda$, it means geometrically that the point is behind the retinal plane of the virtual camera. In this case, the point is projected to the desired location algebraically, we cannot obtain correct views through the graphics library.

### 4.3 Projection by the Virtual Camera

During the embedding procedure, 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 $x^i_k$ of the vertices $\{E_j\}_{j=5,6,7}$ and examine our embedding result. An example of the result of embedding is shown in the bottom half of Fig. 7.

### 4.4 Virtual Camera Decomposition

To use the virtual camera matrix in graphics, we should decompose it into three parts $K, R, t$, as shown in (2). Since we know from (5) that

$$t \approx R^{-1} \hat{t} \begin{bmatrix} R_k \\ t_k \end{bmatrix} \begin{bmatrix} R_k & t_k \\ 0^T & 1 \end{bmatrix} \begin{bmatrix} \hat{t} \end{bmatrix} = \begin{bmatrix} R_k \hat{t} \\ t_k \end{bmatrix}, \quad (36)$$

we first compute the scale factor $\rho_q$ of the matrix $Q$ and then $K$ is computed using Cholesky decomposition [38]. Finally, $R$ and $t$ are computed using (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 the 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. Computation 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 the $k$th video image is entered, the corresponding virtual camera $Q_k$ is computed as follows:

1. Find image corner points $\{x_{ki}\}_{i=1}^N$ 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 (11) will give $P_k$.
3. Project the projective 3D basis points $\{X^i_k\}_{i=0}^4$ into the $k$th video image through $P_k$ to obtain $\{x^i_k\}_{i=0}^4$:

$$x^i_k \approx P_k X^i_k, \quad \text{for } i = 0, \ldots, 4. \quad (34)$$

4. Compute the $k$th virtual camera $Q_k$ using the image points $\{x^i_k\}_{i=0}^4$ 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 vary 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:

$$Q_k = \rho_q K_k [R_k | t_k]. \quad (35)$$

Now, the scale factor $\rho_q$ is no longer useful in graphics rendering. Fig. 8 shows the three stages of 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 $K_k$ gives the perspective viewing volume. The modelview matrix $M_k$ in Fig. 8 is defined by the rotation $R_k$ and $t_k$ as follows:

$$M_k = \begin{bmatrix} R_k & t_k \\ 0^T & 1 \end{bmatrix}. \quad (36)$$
This transformation converts a 3D point $X_W$ defined with respect to the world coordinate system $O_W$ to the point $X_C$ with respect to the camera coordinate system $O_C$. Afterward, a projection matrix is applied to yield clip coordinates $X_C$. The internal calibration matrix $K_k$ determines this matrix and defines the corresponding viewing volume. Any parts 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 [39].

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. Fig. 9 shows the result of video augmentation. This video sequence was captured by a hand-held video camera and is 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 interleaving in the measurement of corner points by dividing each image into 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. In tracking, a form of Kalman filter was utilized to facilitate the process of finding corner points.

Fig. 10 shows tracked lines and their intersection points. The size of each field was $720 \times 243$. The intersection points were utilized in the projective motion estimation. First, two fields were selected as the control images, as shown in Fig. 7, and then the fundamental matrix was computed using the algorithm of [40]. 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. Fig. 11 shows the performance of our projective motion estimation using the tracked intersection points. It depicts the graphs of the maximum and RMS of reprojection errors

$$E_{ki} = \left\| x_{ki} - P_k X_i \right\|_2,$$

where $x_{ki}$ 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, showing the accuracy of the detection of the intersection points. The values of the skew 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 [39]. We defined a light source and three graphics objects with respect to the world coordinate system whose material properties were different from each other. Finally, the two even and odd fields augmented by graphic views are recombined to yield an interlaced frame. Due to our system limit, the results were obtained through an off-line process.

Figs. 12 and 13 show another experiment. The video sequence was composed of 400 frames. We selected the zeroth and 100th video images as the two control images in which the locations of the five vertices of the world
coordinate system were interactively chosen. Fig. 12a and Fig. 12b show the two control images, five vertices chosen, and four epipolar lines drawn for embedding. Fig. 12c and Fig. 12d show the 300th and 399th images, on which the unit cube of the world coordinate system embedded was drawn. Fig. 13 shows four augmented video images. They correspond to the (a) zeroth, (b) 100th, (c) 300th, and (d) 399th 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 the 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 \( \{X_i^\varepsilon\} \) 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:

\[
x_i^\varepsilon \approx QE_i.
\]  

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

\[
x_i^\varepsilon \approx PX_i^\varepsilon.
\]

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

\[
P_k T_{4\times4}^p \approx P_k^\varepsilon.
\]

Let us suppose that we know \( T_{4\times4}^p \). Then, we have

\[
x_i^\varepsilon \approx PX_i^\varepsilon
\]

\[
\approx PT_{4\times4}^p (T_{4\times4}^p)^{-1} X_i^\varepsilon
\]

\[
\approx P^\varepsilon X_i^\varepsilon.
\]

¹ Video clips can be found on our web site http://cafe.postech.ac.kr/~dragon/un-ar.
where \( P^e = PT_{4 \times 4}^p \) and \( X_i^e = (T_{4 \times 4}^p)^{-1}X_i^b \). The 3D Euclidean coordinate \( X_i^e \) is a point represented in a Euclidean coordinate system of the real world. Then, do the coordinates \( \{ X_i^e \} \) constitute a rectangular coordinate system like \( \{ E_i \} \) as shown in Fig. 6? The answer is no. It is clear that there exists a nonsingular transformation \( T \) between \( \{ X_i^e \} \) and \( \{ E_i \} \):

\[
E_i \approx TX_i^e.
\]

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 orthonormality of the coordinate system composed of \( \{ X_i^e \} \). Hence, the transformation is included in a more general class, the group of projective transformations, even though we cannot discount the possibility of the transformation being a Euclidean transformation by manual specification.

Practical implication of the 3D projective transformation \( T \) results in a projectively distorted graphics view. Fig. 14 gives 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 DISCUSSIONS AND 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 and move it 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 and physically correct.

Real-time implementation is another issue, for which detecting and tracking image corner points, sequential update of projective motion, and rendering graphics objects should be done in real-time. Our current hardware system used for the experiments is SGI Power Indigo 2, which has one 75 MHZ IP26 Processor of MIPS R8000 Processor CPU and MIPS R8010 FPU Chip, running IRIX 6.2 OS. As a system setup, computing fundamental matrix takes a few seconds, but the time of embedding procedure with user-interface, specifying five basis points, cannot be measured. Except for the graphics generation, which requires a high performance computer, it takes 50 milliseconds per frame during the sequence to find the locations of 32 corner points of the black rectangles. If the procedure of image processing is optimized for speed, our method can be modified easily.

![Fig. 13. Another video augmentation result. With respect to the embedded world coordinate system, shown in Fig. 12, we augmented the real world with two virtual light sources and one red teapot. (a) (I_0) and (b) (I_100) correspond to the two control images. (c) (I_300) and (d) (I_399) are the 300th and 399th augmented images, respectively.](image)

![Fig. 14. 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 the arrow would not be visible if the rectangular cube was a real object.](image)
for use in a real-time system since, for example, the computation of projective motion by a least-square method requires one singular value decomposition of 12 × 12 matrix and a few multiplications.

On the other hand, a usage of lenses with wide fields of view giving lots of radial distortion may make the system unstable since projective structure computation will consider it as a corner localization error, causing unstable registration of virtual objects.

To summarize, we propose a method for augmented reality without explicit Euclidean camera calibration by adopting the notion of virtual camera handled by the projective motion of the real camera and the embedded world coordinate system. We hope this research provides a link between computer vision and computer graphics.

ACKNOWLEDGMENTS

The authors would like to thank associate editor Lisa Avila for the kind communications. This work was partially supported by postdoctoral fellowships program from the Korea Science and Engineering Foundation (KOSEF).

REFERENCES

Yongduek Seo received the BS degree in electronic engineering from Kyungpook National University in 1992, and the MS 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, completed in Spring 2000. During April-May 1999, he was a guest researcher in the Mathematical Imaging Group, Lund Institute of Technology, Sweden. Currently, he is with the Microwave Applications Research Center (MARC) in POSTECH. His research interests include differential geometry, real-time computer vision, camera self-calibration, structure recovery from motion, and augmented reality.

Ki Sang Hong received the BS degree in electronic engineering from Seoul National University, Korea, in 1977, and the MS and PhD 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, Pennsylvania, as a visiting professor. His current research interests include computer vision and pattern recognition. He is a member of the IEEE.