Weakly Calibrated Video-based Augmented Reality: Embedding and Rendering through Virtual Camera

Yongduek Seo
Microwave Applications Research Center, POSTECH, Republic of Korea
dragon@postech.ac.kr

Ki-Sang Hong
IIP Lab., EE Dept, Pohang University of Science and Technology (POSTECH),
hongks@postech.ac.kr

Abstract

Proposed is 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. 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 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. Real experiments show the performance of the algorithm.

1. Introduction

Video-based augmented reality mixes 3D graphics into a real video [1, 14]. Virtual objects are usually 3D graphics models and the images of them are rendered by graphics machines and overlayed on real video frames. Computer vision techniques have played a major role in augmented reality: estimating the camera parameters [11, 12, 18], resolving occlusion between a virtual object and a real object [3, 9], real-time camera calibration for a virtual studio [16], correcting the registered location of a graphic object dynamically [2], etc.

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. Recently, some methods that do not require explicit metric calibration have been proposed. Kutulakos and Vallino utilized affine object representation to augment orthographic video streams [13]. Faugeras applied a self-calibration method for fixed internal parameters to the augmentation of real video sequences [7], where self-calibration replaced the function of a calibration pattern. Seo and Hong inserted a real object into a perspective video sequence using a projective motion and image-based rendering technique [20]. Chen *et. al.* utilized a cuboid structure for augmenting a single image with computer graphics [4]. We inform that the work of Chen *et. al.* has been done independently of our work.

The study of Kutulakos and Vallino [13] was innovative; they augmented real video *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 textures 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 a Euclidean camera calibration. This paper has two main contributions by extending the work of [13]. First, full graphic functions, like shading by virtual lighting, texture, etc, can be utilized by virtue of Euclidean characteristic of the virtual camera. Second, the real camera is now modeled by a pin-hole camera performing perspective projection, a more general one than affine orthographic projection camera.

Our algorithm consists of two parts: *embedding* and *rendering*. The world coordinate frame for graphics objects are

0-7695-0846-4/00 $10.00

specified in two selected video images, as was done in [13], in order to insert the virtual world into the real world (embedding) in Section 4. 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 OpenGL™ library (rendering). Thus, we can generate perspective views of graphic objects, as shown in Section 5, having textures and shading by virtual light sources. The motion and change of the internal parameters like focal length of the real video camera is represented by projective camera matrices that can be obtained through a projective reconstruction algorithm [8, 10, 22].

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 our embedding algorithm. A method to compute a virtual camera and interface it with a graphics library are provided in Section 5. Section 6 shows the experimental results of our method, and Section 7 deals with the problem of different perspectives, which is inherent in our application of non-metric vision. The proposed method can be similary adopted to the case of affine projection camera. We briefly introduce the affine camera case in Section 8. Finally, concluding remarks are given in Section 9. A detailed version of this work can be found in [19, 21].

2. Preliminaries

A 2D image point is represented by a 3D homogeneous vector $\boldsymbol{x}$ with the third component being one. Further, a 3D space point is represented by a 4D homogeneous vector $\boldsymbol{X}$ with the fourth component being one:

$$\boldsymbol{x} = [u \quad v \quad 1]^{\mathrm{T}}, \quad \boldsymbol{X} = [X \quad Y \quad Z \quad 1]^{\mathrm{T}}. \quad (1)$$

The k-th video image is represented by $\mathcal{I}_k$. The 3×4 camera matrix obtained by projective reconstruction from real video images is denoted by $\boldsymbol{P}_k$.

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

$$\boldsymbol{Q} = \rho \boldsymbol{K}[\boldsymbol{R}|\boldsymbol{t}] = [\boldsymbol{H}|\boldsymbol{h}], \quad (2)$$

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

$$\boldsymbol{K} = \begin{bmatrix} \alpha & 0 & u_0 \\ 0 & \beta & v_0 \\ 0 & 0 & 1 \end{bmatrix}. \quad (3)$$

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

$$\boldsymbol{H} = \rho \boldsymbol{KR}, \qquad \boldsymbol{h} = \rho \boldsymbol{Kt}. \quad (4)$$

Proposition 1 *Let $\boldsymbol{q}_1$, $\boldsymbol{q}_2$ and $\boldsymbol{q}_3$ be three rows, represented in column vector form, of $\boldsymbol{H}$ of a zero-skew camera $\boldsymbol{Q}$. Then*

$$(\boldsymbol{q}_1 \times \boldsymbol{q}_3) \cdot (\boldsymbol{q}_2 \times \boldsymbol{q}_3) = 0, \quad (5)$$

where $\times$ and $\cdot$ are the outer product and the inner product of three dimensional vectors, respectively [5].

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, for embedding graphic objects and rendering them. Here, we briefly introduce the method of projective reconstruction based on two views. Given matching points between two views, the fundamental matrix $\boldsymbol{F}$ satisfies the following:

$$\boldsymbol{x}'^{\mathrm{T}} \boldsymbol{F} \boldsymbol{x} = 0, \quad (6)$$

where $\boldsymbol{x}$ is an image point from one view and $\boldsymbol{x}'$ is from the other view. Then, two camera matrices are chosen as

$$\boldsymbol{P} = [\boldsymbol{I}|\boldsymbol{0}], \quad \boldsymbol{P}' = [[\boldsymbol{e}']_{\times} \boldsymbol{F} | \boldsymbol{e}'], \quad (7)$$

where $\boldsymbol{e}'$ is the epipole in the second image.

A projective 3D point $\boldsymbol{X}_i$ is reconstructed from its image match $(\boldsymbol{x}_i, \boldsymbol{x}'_i)$ by back-projection equations:

$$\lambda_i \boldsymbol{x}_i = \boldsymbol{P} \boldsymbol{X}_i, \quad \lambda'_i \boldsymbol{x}'_i = \boldsymbol{P}' \boldsymbol{X}_i. \quad (8)$$

Solving a linear least-square equation gives $\boldsymbol{X}_i$.

Conversely, the camera matrix $\boldsymbol{P}_k$ for the k-th image may be computed given matches $(\boldsymbol{x}_{ki}, \boldsymbol{X}_i)$ of image points and their corresponding 3D points using the projection equation

$$\lambda \boldsymbol{x}_{ki} = \boldsymbol{P}_k \boldsymbol{X}_i, \text{ for } \quad i = 1, ..., N. \quad (9)$$

Note that this reconstruction is up to a three dimensional projective transformation; projective camera matrices cannot be used in a graphics machine because they are not a Euclidean form of Equation (2), for this reason we employ a virtual camera.

2.2. Projective Plane Homography

Let us suppose that four points $\{\boldsymbol{X}_i\}_{i=1,...,4}$ are on the same plane $\boldsymbol{\Pi}$ and their image points in two different views are $\{\boldsymbol{x}_i\}_{i=1,...,4}$ and $\{\boldsymbol{x}'_i\}_{i=1,...,4}$. Then there exists a 3×3 projective plane homography $\boldsymbol{T}_{\Pi}$ such that

$$\boldsymbol{x}'_i \approx \boldsymbol{T}_{\Pi} \boldsymbol{x}_i. \quad (10)$$

When the fundamental matrix $\boldsymbol{F}$ is given, three point matches are enough to compute $\boldsymbol{T}_{\Pi}$ because the two epipoles $\boldsymbol{e}$ and $\boldsymbol{e}'$ obey Equation (10). Details can be found in [6].

3. Proposed Algorithm

This section gives a brief overview of our algorithm, to be followed by more detail in the next sections. Our algorithm is divided into two parts: embedding and rendering. 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 1, 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 like [13] or video images already captured in the case of the off-line procedure.

(3) Estimate the projective motion, $\boldsymbol{P}$ and $\boldsymbol{P}'$, and structure $\{\boldsymbol{X}_i\}_{i=1}^N$ for the two images using point correspondences as described in Section 2.1.

(4) Specify the locations $\{(\boldsymbol{x}_i^b, \boldsymbol{x}_i^{b'})\}$ 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 $\boldsymbol{X}_i^b$, $i = 0, ..., 4$, of the five specified basis points $\{(\boldsymbol{x}_i^b, \boldsymbol{x}_i^{b'})\}$ using $\boldsymbol{P}$ and $\boldsymbol{P}'$.

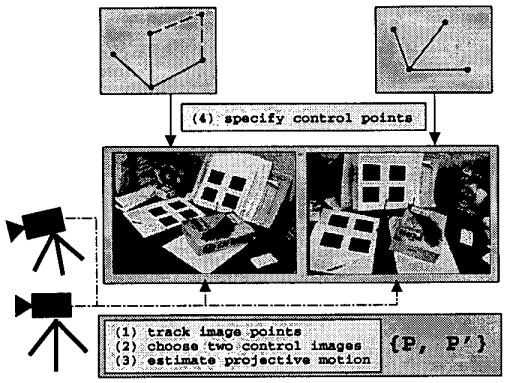

Figure 1. Embedding procedure.

The rendering steps, shown in Figure 2, are as follows:

(6) (a) For k-th input video image $\mathcal{I}_k$, detect image corner points and compute $\boldsymbol{P}_k$.

(b) Transfer or project the 3D points, $\{\boldsymbol{X}_i^b\}_{i=0}^4$ onto the k-th video image $\mathcal{I}_k$ using the projection equation $\boldsymbol{x}_{ki}^b \approx \boldsymbol{P}_k \boldsymbol{X}_i^b$, $i = 0, ..., 4$.

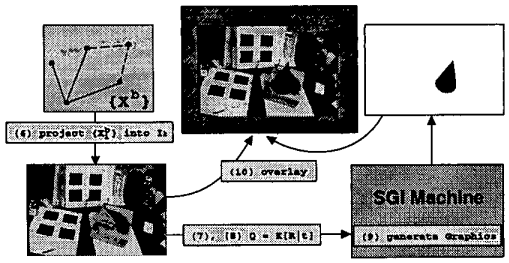

Figure 2. Rendering procedure.

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

(8) Decompose $\boldsymbol{Q}_k$: $\boldsymbol{Q}_k = \rho_k \boldsymbol{K}_k[\boldsymbol{R}_k|\boldsymbol{t}_k]$.

(9) (a) Set virtual camera in graphics machine using $\boldsymbol{K}_k$, $\boldsymbol{R}_k$ and $\boldsymbol{t}_k$.

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

(10) The graphics view $\mathcal{I}_k^G$, synthesized from the graphics machine, is overlayed on $\mathcal{I}_k$.

4. Embedding

Our first step is to find image corner points and compute projective camera matrices, $\boldsymbol{P}$ and $\boldsymbol{P}'$, of the two control views and the 3D projective coordinates $\{\boldsymbol{X}_i\}_{i=1}^N$.

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, Figure 3 shows two control images $\mathcal{I}_0$ and $\mathcal{I}_{270}$ 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 $\{\boldsymbol{E}_0, \boldsymbol{E}_1, \boldsymbol{E}_2, \boldsymbol{E}_3, \boldsymbol{E}_4\}$ of the coordinate frame.

2. With the help of epipolar geometry, we choose four image locations of the coordinates $\{\boldsymbol{E}_0, \boldsymbol{E}_1, \boldsymbol{E}_2, \boldsymbol{E}_3\}$ in the second control image $\mathcal{V}'$.

Here $\boldsymbol{E}_0 = [0, 0, 0, 1]^{\mathrm{T}}$ is the origin of the world coordinate frame, and the others are the vertices (or basis points) of the unit cube of the world coordinate system, and they are defined as $\boldsymbol{E}_1 = [1, 0, 0, 1]^{\mathrm{T}}$, $\boldsymbol{E}_2 = [0, 1, 0, 1]^{\mathrm{T}}$, $\boldsymbol{E}_3 = [0, 0, 1, 1]^{\mathrm{T}}$, $\boldsymbol{E}_4 = [1, 0, 1, 1]^{\mathrm{T}}$ and so on.

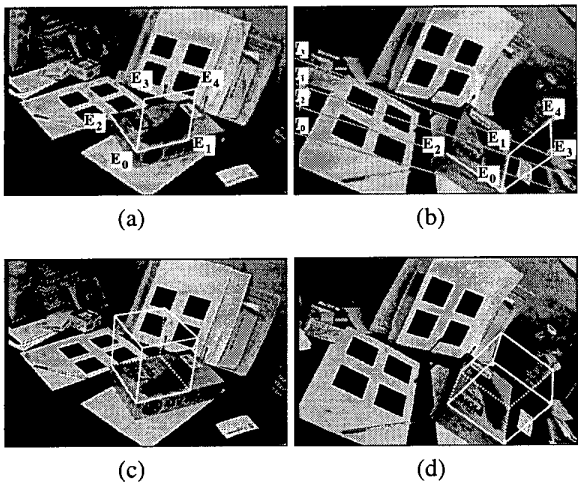

Figure 3. (a) Five points are specified for embedding in $\mathcal{V}$, and (b) four points on the corresponding epipolar lines in $\mathcal{V}'$. Specified image points determine its corresponding virtual camera from which the images of the world coordinate frame are drawn in (c) and (d).

4.1. Embedding via Epipolar Geometry

First, *five* locations $\{\boldsymbol{x}_i^b\}_{i=0}^4$ of the vertices are specified in the first control image $\mathcal{V}$. This determines the look 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 Equation (6). Now, we choose *four* image locations $\{\boldsymbol{x}_i^{b\prime}\}_{i=0}^3$ of the vertices on the corresponding epipolar lines $\boldsymbol{l}_i'$:

$$\boldsymbol{l}_i' = \boldsymbol{F}\boldsymbol{x}_i \quad i = 0, ..., 3. \tag{11}$$

However, we do not need to specify the last one, $\boldsymbol{x}_4^{b\prime}$, because it can be determined by the plane homography $\boldsymbol{T}_\Pi$ determined by the equations $\boldsymbol{x}_i^{b\prime} \approx \boldsymbol{T}_\Pi \boldsymbol{x}_i^b$, $i = 0, 1, 3$ and $\boldsymbol{e}' \approx \boldsymbol{T}_\Pi \boldsymbol{e}$ as given in Section 2.1.

Finally, we compute the projective 3D coordinates $\{\boldsymbol{X}_i^b\}_{i=0}^4$ 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, as explained in Section 5.1.

Figure 3 gives an example where five image locations in the first control image $\mathcal{V}_1$ and four in the second $\mathcal{V}_2$ are selected. Using those 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 into 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, 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 $\boldsymbol{Q}$ by a 3×4 matrix

$$\boldsymbol{Q} = [\boldsymbol{a}_1\ \boldsymbol{a}_2\ \boldsymbol{a}_3\ \boldsymbol{a}_4], \tag{12}$$

where $\boldsymbol{a}_k$ is the k-th column of the matrix. The image location of a vertex $\boldsymbol{E}_i$ is denoted by $\boldsymbol{x}_i^b = [u_i^b\ v_i^b\ 1]^{\mathrm{T}}$. Using the relationship

$$\lambda_i \boldsymbol{x}_i^b = \boldsymbol{Q}\boldsymbol{E}_i \tag{13}$$

we have

$$\boldsymbol{a}_4 = \lambda_0 \boldsymbol{x}_0^b \tag{14}$$

$$\boldsymbol{a}_i = \lambda_i \boldsymbol{x}_i^b - \lambda_0 \boldsymbol{x}_0^b, \text{ for } i \in \{1, 2, 3\}. \tag{15}$$

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

$$\boldsymbol{Q} = [\lambda_1 \boldsymbol{x}_1^b - \boldsymbol{x}_0^b | \lambda_2 \boldsymbol{x}_2^b - \boldsymbol{x}_0^b | \lambda_3 \boldsymbol{x}_3^b - \boldsymbol{x}_0^b | \boldsymbol{x}_0^b]. \tag{16}$$

Using the fifth point $\boldsymbol{x}_4^b$, we have

$$\lambda_4 \boldsymbol{x}_4^b = \boldsymbol{Q}\boldsymbol{E}_4 = \lambda_1 \boldsymbol{x}_1^b + \lambda_3 \boldsymbol{x}_3^b - \boldsymbol{x}_0^b, \tag{17}$$

which gives two equations to compute λ_1 and λ_3:

$$\begin{bmatrix} u_4^b - u_1^b & u_4^b - u_3^b \\ v_4^b - v_1^b & v_4^b - v_3^b \end{bmatrix} \begin{bmatrix} \lambda_1 \\ \lambda_3 \end{bmatrix} = \begin{bmatrix} u_4^b - u_0^b \\ v_4^b - v_0^b \end{bmatrix}. \tag{18}$$

Now, we compute the last scale λ_2 using Proposition 1. Since we know λ_1 and λ_3, the three vectors $\boldsymbol{q}_1$, $\boldsymbol{q}_2$ and $\boldsymbol{q}_3$ may be written as:

$$\boldsymbol{H} = [\boldsymbol{a}_1\ \boldsymbol{a}_2\ \boldsymbol{a}_3] = \begin{bmatrix} \boldsymbol{q}_1^{\mathrm{T}} \\ \boldsymbol{q}_2^{\mathrm{T}} \\ \boldsymbol{q}_3^{\mathrm{T}} \end{bmatrix} = \begin{bmatrix} a & \lambda_2 u_2^b - u_0^b & d \\ b & \lambda_2 v_2^b - u_0^b & e \\ c & \lambda_2 - 1 & f \end{bmatrix}. \tag{19}$$

From Equation (5), we have a quadric equation for λ_2:

$$A\lambda_2^2 + B\lambda_2 + C = 0, \tag{20}$$

where A, B and C are the coefficients composed of the elements of the matrix $\boldsymbol{H}$. 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. There is also the possibility of obtaining no real solutions for Equation (20). 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 focal 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 $\boldsymbol{Q}$ we compute the image locations $\boldsymbol{x}_i^b \approx \boldsymbol{Q}\boldsymbol{E}_i$ of the vertices $\{\boldsymbol{E}_i\}_{i=5,6,7}$ and examine our embedding result. An example of the result of embedding is shown in the bottom half of Figure 3.

4.4. Virtual Camera Decomposition

To use the virtual camera matrix in graphics, we should decompose it into three parts $\boldsymbol{K}$, $\boldsymbol{R}$ and $\boldsymbol{t}$ as shown in equation (2). Equation (2) gives $\boldsymbol{H}\boldsymbol{H}^{\mathrm{T}} = \rho^2\boldsymbol{K}\boldsymbol{K}^{\mathrm{T}}$, and we first compute the scale factor $\rho = ||\boldsymbol{q}_3||$ of the matrix $\boldsymbol{Q}$, and then $\boldsymbol{K}$ is computed using Cholesky decomposition [17]. Finally, $\boldsymbol{R}$ and $\boldsymbol{t}$ are computed by removing out $\boldsymbol{K}$ and ρ from $\boldsymbol{H}$ and $\boldsymbol{h}$.

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 graphics modeling usually 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 $\boldsymbol{Q}_k$ is computed as follows.

1. Find image matches $\{\boldsymbol{x}_i\}_{i=1}^N$ in the k-th video image.
2. Compute the projective camera matrix $\boldsymbol{P}_k$ using the recovered 3D projective coordinates $\{\boldsymbol{X}_i\}_{i=1}^N$ of the image matches. Equation (8) will give $\boldsymbol{P}_k$.
3. Project the projective 3D basis points $\{\boldsymbol{X}_i^b\}_{i=0}^4$ into the k-th video image by $\boldsymbol{P}_k$ to compute $\{\boldsymbol{x}_{ki}^b\}_{i=0}^4$:
$$\boldsymbol{x}_{ki}^b \approx \boldsymbol{P}_k\boldsymbol{X}_i^b, \quad \text{for } i = 0, ..., 4, \tag{21}$$
4. Compute the k-th virtual camera $\boldsymbol{Q}_k$ using the image points $\{\boldsymbol{x}_{ki}^b\}_{i=0}^4$ through the procedure of Section 4.2.
5. Decompose $\boldsymbol{Q}_k$ into $\boldsymbol{K}_k$, $\boldsymbol{R}_k$ and $\boldsymbol{t}_k$ as in Section 4.4.

Note that the components, $\boldsymbol{K}_k$, $\boldsymbol{R}_k$ and $\boldsymbol{t}_k$, 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 $\boldsymbol{Q}_k$ into three parts

$$\boldsymbol{Q}_k = \rho_k\boldsymbol{K}_k[\boldsymbol{R}_k|\boldsymbol{t}_k]. \tag{22}$$

Now, the scale factor ρ_k is no longer useful in graphics rendering. The rotation $\boldsymbol{R}_k$ and translation $\boldsymbol{t}_k$ define the modeling transformation and $\boldsymbol{K}_k$ gives the perspective viewing volume. The generated view is then overlayed on the corresponding video image $\mathcal{I}_k$, which achieves the effect of video augmentation.

6. Experiments

We implemented and tested our method. Figure 4 shows the result of video augmentation. With a video sequence captured by a hand-held video camera, it 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 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 5 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

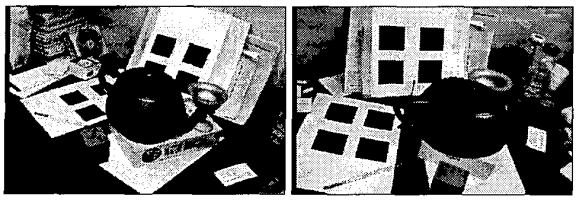

Figure 4. A result of video augmentation. Notice the specularity of the teapot and the cup, as well as the shading effect on each of the objects.

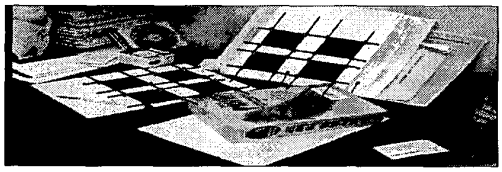

Figure 5. Lines tracked and their intersection points. A total of 32 intersection points are detected in each of the 546 fields.

Figure 3, and then the fundamental matrix were computed using the algorithm of [23]. 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 6 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 $E_{ki} = \left\| \boldsymbol{x}_{ki} - \frac{\boldsymbol{P}_k \boldsymbol{X}_i}{\boldsymbol{P}_k^3 \boldsymbol{X}_i} \right\|_2$, where $\boldsymbol{x}_{ki}$ is the i-th intersection point measured in the k-th field, $\boldsymbol{P}_k^3$ is the third row of $\boldsymbol{P}_k$, and $\| \cdot \|_2$ is the Euclidean norm. The maximum reprojection error was below a 1.4 pixel error during the sequence. 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 [15]. 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.

Figure 7 shows another experiment. The video sequence

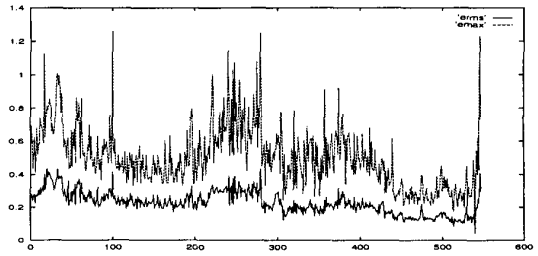

Figure 6. Evaluation of our projective motion estimation. Maximum and RMS reprojection errors are plotted.

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. In this experiment, we used two light sources and one red teapot for the virtual world.

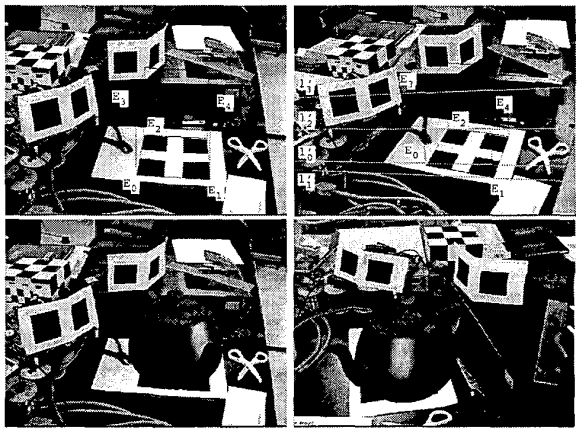

Figure 7. (Up) Embedding for another video sequence. Two control images on which five vertices and four epipolar lines are illustrated. Another video augmentation result. (Down) With respect to the embedded world coordinate system, graphic objects and light sources are located.

7. Real Camera vs. Virtual Camera

Now, let us consider the effects of using virtual camera on the results. 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*. It results in a projectively distorted graphics view. Figure 8 shows 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, caused by our embedding, would not have been seen in the image.

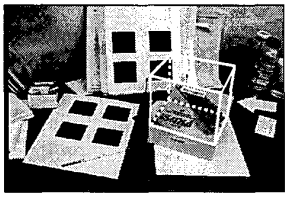

Figure 8. Implicit projective distortion. The view of a rectangular cube through a virtual camera shows a different perspectivity compared to the views of real objects. In particular, the face indicated by arrow would not be seen if the rectangular cube was a real object.

8. Affine Projection Camera Model

The method proposed here can be applied to the case of affine projection camera model; affine strcutre from motion algorithm should be utilized, virtual camera is then modeled to have the form of scaled orthographic (or weak-perspective) camera. In the embedding step, we come to have another pair of hyperbolic constraint curves in addition to three epipolar lines of the basis points. Note that four epipolar lines are utilized in [13]. By selecting the final fourth basis points on a hyperbolic constraint curve gives full matrix elements of the virtual weak-perspective camera, which will then be decomposed into Euclidean components as we did in the previous sections. This could be the direct extension of [13]. A detail on the method is omitted here but can be found in [19].

Figure 9 shows example results of video augmentation when the real camera is modeled as an affine projection camera as Kutulakos and Vallino did in [13], and the virtual camera is modeled by a weak-perspective camera.

9. 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. 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 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. We hope this research provides a link between computer vision and computer graphics.

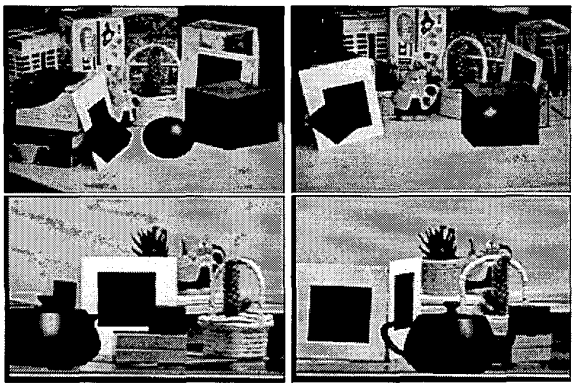

Figure 9. Augmented images when the real camera is assumed to be an affine projection camera. The sphere was desined to move to the left direction.

Acknowledgements

This work was partially supported by postdoctoral fellowships program from Korea Science & Engineering Foundation (KOSEF).

References

[1] R. T. Azuma. A survey of augmented reality. *PRESENCE: Teleoperations and Virtual Environments*, 6(4):355–385, August 1997.

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

[3] M.-O. Berger. Resolving occlusion in augmented reality: a contour based approach without 3D reconstruction. In *CVPR'97*, 1997.

[4] C.-S. Chen, C.-K. Yu, and Y.-P. Hung. New calibration-free approach for augmented reality based on parameterized cuboid structure. In *Proc. 7th Int. Conf. on Computer Vision, Kerkyra, Greece*. IEEE Computer Society Press, 1999.

[5] O. Faugeras. *Three-Dimensional Computer Vision.* MIT Press, Cambridge, Mass, 1993.

[6] O. Faugeras. Stratification of three-dimensional vision: Projective, affine, and metric representations. *J. Opt. Soc. Am. A*, 12(3):465–484, Mar. 1995.

[7] O. Faugeras. From geometry to variational calculus: theory and applications of three-dimensional vision. In *IEEE and ATR Workshop on Computer Vision for Virtual Reality Based Human Communications*, January 1998.

[8] O. D. Faugeras. What can be seen in three dimensions with an uncalibrated stereo rig? In G. Sandini, editor, *Proc. 2nd European Conf. on Computer Vision, Santa Margherita Ligure, Italy*, pages 563–578. Springer-Verlag, 1992.

[9] S. Grosskopf and P. Neugebauer. The use of reality models in augmented reality applications. In *SMILE Workshop on 3D structure from multiple images of large-scale environments, in conjunction with ECCV'98*, June 1998.

[10] R. I. Hartley, R. Gupta, and T. Chang. Stereo from uncalibrated cameras. In *CVPR'92*, 1992.

[11] D. Koller, G. Klinker, E. Rose, D. Breen, R. Whitaker, and M. Tuceryan. Automated camera calibration and 3d egomotion estimation for augmented reality applications. In *7th International Conference on Computer Analysis of Images and Patterns (CAIP-97), Kiel, Germany*, pages 199–206, September 1997.

[12] D. Koller, G. Klinker, E. Rose, D. Breen, R. Whitaker, and M. Tuceryan. Real-time vision-based camera tracking for augmented reality applications. In *ACM Symposium on Virtual Reality Software and Technology (VRST-97), Lausanne, Switzerland*, pages 87–94, September 1997.

[13] K. N. Kutulakos and J. Vallino. Calibration-free augmented reality. *IEEE Trans. Visualization and Computer Graphics*, 4(1):1–20, 1998.

[14] Y. Ohta and H. Tamura, editors. *Mixed Reality - Merging real and virtual worlds.* Springer Verlag, 1999.

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

[16] S.-W. Park, Y. Seo, and K. S. Hong. Real-time camera calibration for virtual studio. *Accepted to J. Real-Time Imaging, To be published*, 2000.

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

[18] 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*, June 1998.

[19] Y. Seo. *Non-metric Augmented Reality and Flexible Auto-Calibration.* PhD thesis, EE Department, Pohang University of Science and Technology (POSTECH), 2000.

[20] Y. Seo, M. Ahn, and K. S. Hong. Video augmentation by image-based rendering under the perspective camera model. In *International Conference on Pattern Recognition*, pages 1694–1696, Aug. 1998.

[21] Y. Seo and K.-S. Hong. Calibration-free augmented reality in perspective. Accepted to IEEE Trans. Visualization and Computer Graphics, 2000.

[22] 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.

[23] Z. Zhang. Determining the epipolar geometry and its uncertainty: A review. Technical Report 2079, INRIA, France, July 1996.