Non-Metric Augmented Reality by Virtual Camera Method under Affine Projection Model*

Yongduek Seo Ki Sang Hong
IIP Lab. Pohang University of Science & Technology (POSTECH)

Abstract

An algorithm is developed for augmenting a real video/scene with virtual graphics objects without computing Euclidean information. For this, we design a method of specifying the virtual camera which performs Euclidean orthographic projection in recovered affine space. In addition, our method has the capability of generating views of objects shaded by virtual light sources, which was unavailable previously. Our novel formulation and experimental results are presented.

1 Introduction

Making views of a 3D graphic model according to the motion of camera and mixing them with a real scene video in real time have been the major subject of *augmented reality* [2]. The virtual objects are generally 3D graphics models and their rendered views by graphics machines are overlayed on real-video frames. Applications include image-guided or assisted surgery or its training [3, 9], assembly, maintenance and repair [28, 26], simulating light conditions of an outdoor building [6], virtual studio systems for commercial/public broadcasting [18], etc. These systems enrich reality with computer-generated views, cooperating with computer vision techniques: estimating the camera parameters [22], resolving occlusion between a virtual object and a real object [5, 10], and correcting the registered location of a graphic object dynamically [3]. An extensive survey can be found in [17]. Among the techniques, camera calibration has been a prerequisite for embedding virtual objects into video frames because the geometric relationship among physical objects, virtual objects and the camera needs be established to get correct results.

On the other hand, there have been a lot of researches on non-metric vision applications: visual control or visual guided tasks [11, 29], navigation [4, 21], object recognition [8], etc. In particular, developments in computer vision technology have brought forth weakly-calibrated methods for augmenting real video with graphics objects without Euclidean information [14, 7]. Especially, the orthographic projection camera model and affine 3D representation together with world coordinate specification by user-interface have been the core of Kutulakos and Vallino's first innovative work of the un-calibrated augmented reality system [14]. They successfully augmented orthographic projection images with graphics objects using their real time system *without* explicit camera calibration and Euclidean reconstruction. However, the system had a limitation that it could not generate a shaded view of graphics objects nor specify a lighting source because theories and packages of computer graphics are written on the basis of *Euclidean geometry*. Indeed, this is one of the critical reasons that previous augmented reality systems have relied on the calibration of their cameras. Recently, Chen *et. al.* [7] proposed a method, independently of this paper, for overlaying graphics on the images. A cuboid, specified through a user-interface, is utilized to connect real camera to computer graphics world. However, they only dealt with a fixed (not moving) camera. In this paper, we extend the work of Kutulakos and Vallino [14], applying 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.

There are two kinds of orthographic cameras: affine camera and weak-perspective camera. The weak-perspective camera retains the features of the Euclidean projection model; it consists of rotation, translation and scaling. On the contrary, the affine camera assumes affine geometry and affine ambient space, which means that direct application of the affine representation to Euclidean computer graphics may cause a deficiency.

We attach a virtual camera of the weak-perspective model to the real affine camera so that the virtual camera moves in the same way as does the real camera. Our method *decomposes* the virtual camera into Euclidean motion components and internal calibration components, which enables us to make full use of a well-developed computer graphics package and to synthesize graphics objects that are shaded and colored by a lighting source. This paper is an extension our non-metric AR method to orthographic camera model. Details on Implementation and experiments for perspective camera model can be found in [24].

Section 2 provides some notations and preliminaries of camera models and a brief review of affine structure from motion algorithm. Section 3 presents a short description of our algorithm. Section 4 illustrates how the virtual world coordinate system is embedded into the recovered affine space. Also it gives the method of computing a weak-perspective virtual camera by specifying four basis points in two control images. Section 5 shows how a corresponding virtual camera is computed, given a video image, and graphics views are rendered using an SGI graphics machine . Section 6 illustrates our experimental results. Finally, concluding remarks are given in Section 7.

*This work is supported by KOSEF and BK-21.

2 Preliminaries

Every vector is denoted by a column vector. We represent an image point by $\boldsymbol{x} = [u, v]^\mathsf{T}$ and a 3D point by $\boldsymbol{X} = [X, Y, Z, 1]$. A 3D point $\boldsymbol{X}$ projects to the image point $\boldsymbol{x}$:

$$\boldsymbol{x} = \boldsymbol{K}_{2\times2}\boldsymbol{I}_{2\times4}\boldsymbol{D}_{4\times4}\boldsymbol{X} \quad (1)$$

$$= \begin{bmatrix} f_X & 0 \\ 0 & f_Y \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{R} & \boldsymbol{t} \\ \boldsymbol{0}_3^\mathsf{T} & 1 \end{bmatrix} \boldsymbol{X}, \quad (2)$$

where $\boldsymbol{R} = [\boldsymbol{r}_1\, \boldsymbol{r}_2\, \boldsymbol{r}_3]^\mathsf{T}$, $\boldsymbol{r}_i^\mathsf{T}$ being i-th row vector, and $\boldsymbol{t} = [t_X, t_Y, t_Z]^\mathsf{T}$ denote the transformation from the world coordinate system to the camera coordinate system. Notice that $\boldsymbol{X}$ is represented in the world coordinate system. Thus, we have the following [16]:

Definition 1 *A weak perspective camera $\boldsymbol{P}_{\mathrm{wp}}$ is defined by a 2×4 matrix as follows:*

$$\boldsymbol{P}_{\mathrm{wp}} = \begin{bmatrix} f_X & 0 \\ 0 & f_Y \end{bmatrix} \begin{bmatrix} \boldsymbol{r}_1^\mathsf{T} & t_X \\ \boldsymbol{r}_2^\mathsf{T} & t_Y \end{bmatrix} \quad (3)$$

$$= \begin{bmatrix} \boldsymbol{b}_1^\mathsf{T} & t'_X \\ \boldsymbol{b}_2^\mathsf{T} & t'_Y \end{bmatrix} \quad (4)$$

$$= \begin{bmatrix} \boldsymbol{a}_1 & \boldsymbol{a}_2 & \boldsymbol{a}_3 & \boldsymbol{a}_4 \end{bmatrix}. \quad (5)$$

Notice that $\boldsymbol{a}_i$ is the i-th column of $\boldsymbol{P}_{\mathrm{wp}}$, and $\boldsymbol{b}_1^\mathsf{T} = f_X \boldsymbol{r}_1^\mathsf{T}$ and $\boldsymbol{b}_2^\mathsf{T} = f_Y \boldsymbol{r}_2^\mathsf{T}$ satisfy the **orthogonality constraint:**

$$\boldsymbol{b}_1 \cdot \boldsymbol{b}_2 = 0, \quad (6)$$

where $\cdot$ represents the operation of the inner product. Equation (1) can now be simply written as $\boldsymbol{x} = \boldsymbol{P}_{\mathrm{wp}}\boldsymbol{X}$, where $\boldsymbol{X}$ is represented in the world coordinate system.

2.1 Affine Structure from Motion

Affine camera model is a generalization of the weak-perspective camera model and may be defined as follows [16].

Definition 2 *An affine camera is defined by a general 2×4 matrix of rank 2 and denoted by $\boldsymbol{P}_A$.*

Replacing $\boldsymbol{K}_{2\times2}$ with a non-singular 2×2 matrix and $\boldsymbol{D}$ with a matrix of general 3D affine transformation in equations (1) and (2), we obtain the general form of the affine camera $\boldsymbol{P}_A$.

Since the recovery of affine structure and motion is not main topic of this paper, here we briefly introduce affine structure from motion algorithms. It is well known that given point matches from two orthographic or scaled orthographic views, affine structure of a scene can be found [13]. Affine epipolar geometry has been studied [25] and affine structure and motion can be obtained through factorization methods [27, 23]. Details can also be found in various literatures [15, 19, 20, 12, 1]. Let us suppose that we have point matches from input images and let $\boldsymbol{x}_{ki}$ be the i-th point of k-th image. Then from the theory of affine structure from motion, we can estimate the k-th *affine* camera matrix $\boldsymbol{P}_{A,k}$ for each of the input images and the i-th *affine* coordinate vector $\boldsymbol{X}_{A}$ that satisfy the following projection equation:

$$\boldsymbol{x}_{ki} = \boldsymbol{P}_{A,k}\boldsymbol{X}_{A,i} \quad \text{for} \quad i = 1, ..., N, \quad k = 1, ..., K, \quad (7)$$

where N is the number of point correspondences and K is the number of input images. Note that this reconstruction is up to an unknown affine transformation [13].

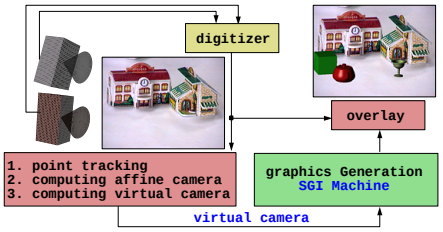


Figure 1: Overall configuration. Point matches are obtained for the recovery of affine motion and structure. The basis points specified in two control video images yield our virtual weak-perspective cameras, through which a shaded graphics objects by a lighting source can be rendered.

3 Algorithm Overview

Our algorithm consists of three steps:

1. Affine structure recovery: We compute the affine camera matrices and affine structure from the image correspondences of two views.
2. Embedding: We insert the graphics world coordinate system into the recovered affine space by specifying four basis points in two *control* images using affine epipolar geometry. Virtual graphics objects are placed with respect to the world coordinate system.
3. Rendering: As the real camera moves, the affine camera is computed, the virtual camera is updated, and the views of graphics objects are rendered and overlayed on the real video images.

Figure 1 briefly shows our method. As the real camera moves, point correspondences are obtained. After choosing two control images, we recover their corresponding affine cameras $\{\boldsymbol{P}_A, \boldsymbol{P}'_A\}$ and 3D affine coordinates $\{\boldsymbol{X}_{A,i}\}_{i=1}^N$ using matches $\{(\boldsymbol{x}_{A,i}, \boldsymbol{x}'_{A,i})\}_{i=1}^N$ from the two control images. *Two control images* may be given from a stereo-rig for a real-time augmented reality system or they may be chosen from the video sequence for off-line video augmentation. The two images of Figure 2 are an example of control images

Then, we specify four basis points in two control images and compute the corresponding virtual camera, which is used for generating a view of graphics objects through an SGI graphics machine. As mentioned, the virtual camera enables us to utilize the full functionality of our graphics package. That is, we can define various characteristics of light sources including their colors and positions.

4 Embedding Algorithm

The first thing we have to do for augmentation is to define the weak-perspective virtual camera attached to the real video camera by specifying some basis points of the graphics world coordinate system. This section provides an image-based embedding method for inserting the world coordinate system by specifying its $3\frac{1}{2}$ basis points in the first control image and *three* basis points in the second. Note that in [14] they specified *four* points in both of the control images using only affine epipolar constraint.

4.1 Embedding in the First Control Image

We first specify *three* image locations $\{\boldsymbol{x}_0^{\mathrm{b}}, \boldsymbol{x}_1^{\mathrm{b}}, \boldsymbol{x}_2^{\mathrm{b}}\}$ of the basis points $\{\boldsymbol{E}_0, \boldsymbol{E}_1, \boldsymbol{E}_2\}$ in the first image. Specifically, $\boldsymbol{E}_0 = [0,0,0,1]^{\mathsf{T}}$ is the origin and $\boldsymbol{E}_1 = [1,0,0,1]^{\mathsf{T}}$, $\boldsymbol{E}_2 = [0,1,0,1]^{\mathsf{T}}$ and $\boldsymbol{E}_3 = [0,0,1,1]^{\mathsf{T}}$ are three basis points on the x, y and z axes, respectively, of the world coordinate system. Using the notation of Equation (5), each column $\boldsymbol{a}_i$ of the virtual camera $\boldsymbol{P}_{\mathrm{wp}}$ is given from the relationship $\boldsymbol{x}_k^{\mathrm{b}} = \boldsymbol{P}_{\mathrm{wp}}\boldsymbol{E}_k$ as follows:

$$\boldsymbol{a}_4 = \boldsymbol{x}_0^{\mathrm{b}}, \tag{8}$$

$$\boldsymbol{a}_j = \boldsymbol{x}_j^{\mathrm{b}} - \boldsymbol{x}_0^{\mathrm{b}} \quad \text{for } j = 1,2. \tag{9}$$

The third column, $\boldsymbol{a}_3$, is still left unknown. If an arbitrary image point is given for $\boldsymbol{E}_3$, it will make our virtual camera a general affine camera like [14]. In order to force the virtual camera to become a weak-perspective camera, we utilize the *orthogonality constraint* of Equation (6). Now we use the notation of Equation (4). Since we have determined $\boldsymbol{a}_1$ and $\boldsymbol{a}_2$, we may write $\boldsymbol{b}_1$ and $\boldsymbol{b}_2$ as

$$\boldsymbol{b}_1 = [a_{11}, a_{12}, a_{13}]^{\mathsf{T}} \quad \text{and} \quad \boldsymbol{b}_2 = [a_{21}, a_{22}, a_{23}]^{\mathsf{T}}, \tag{10}$$

where $\boldsymbol{a}_{ki}$ is the k-th element of $\boldsymbol{a}_i$. Then from the orthogonality constraint, we have

$$0 = \boldsymbol{b}_1 \cdot \boldsymbol{b}_2 = a_{11}a_{21} + a_{12}a_{22} + a_{13}a_{23}, \tag{11}$$

which gives a constraint equation for the elements of the third column $\boldsymbol{a}_3 = [a_{13}, a_{23}]^{\mathsf{T}}$:

$$a_{23} = -\frac{a_{11}a_{21} + a_{12}a_{22}}{a_{13}}. \tag{12}$$

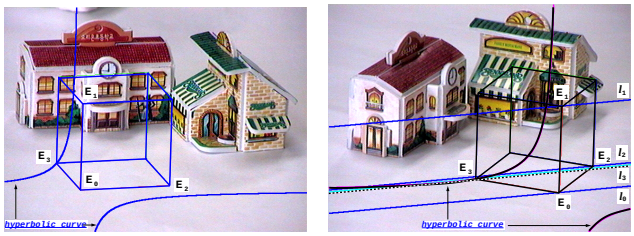
Since $\boldsymbol{x}_3^{\mathrm{b}} = \boldsymbol{P}_{\mathrm{wp}}\boldsymbol{E}_3 = \boldsymbol{a}_3 + \boldsymbol{x}_0^{\mathrm{b}}$, the coordinates of the last image point $\boldsymbol{x}_3^{\mathrm{b}} = [u_3^{\mathrm{b}}, v_3^{\mathrm{b}}]^{\mathsf{T}}$ must satisfy the hyperbolic equation

$$v_3^{\mathrm{b}} - v_0^{\mathrm{b}} = \frac{a_{11}a_{21} + a_{12}a_{22}}{u_3^{\mathrm{b}} - u_0^{\mathrm{b}}}. \tag{13}$$

Finally, we choose an appropriate value of u_3^{b} on the hyperbola and compute $\boldsymbol{P}_{\mathrm{wp}}$ to complete the embedding step for the first image:

$$\boldsymbol{P}_{\mathrm{wp}} = \begin{bmatrix} \boldsymbol{x}_1^{\mathrm{b}} - \boldsymbol{x}_0^{\mathrm{b}} & \boldsymbol{x}_2^{\mathrm{b}} - \boldsymbol{x}_0^{\mathrm{b}} & \boldsymbol{x}_3^{\mathrm{b}} - \boldsymbol{x}_0^{\mathrm{b}} & \boldsymbol{x}_0^{\mathrm{b}} \end{bmatrix}. \tag{14}$$

Figure 2 shows an example of the result of our embedding procedure. The Figure 2(a) is the first image where three image locations were specified as explained. The three specified image points provided a hyperbolic equation, which is also shown in the figure. Finally, the u coordinate of the last image point $\boldsymbol{x}_3^{\mathrm{b}}$ was selected. The wire-cube is the projection of a graphics unit cube through the computed virtual camera.



(a) (b)

Figure 2: Embedding step. (a) (First control image) Three basis points $\{\boldsymbol{E}_0, \boldsymbol{E}_1, \boldsymbol{E}_2\}$ were selected. In selecting these points, we tried to place the basis frame on the same floor that the toy houses existed. Then, $\boldsymbol{E}_3$ was chosen on the corresponding hyperbola. (b) (Second control image) On the corresponding epipolar lines, three basis points were selected. But, the last basis point $\boldsymbol{E}_3$ was computed automatically as the intersection of its epipolar line and the hyperbola.

4.2 Embedding in the Second Control Image

In the second image, we also have to specify the image locations of the basis points of the world coordinate system. In the embedding step for the first image, we are provided with just one constraint for building up a weak-perspective virtual camera. However, in the second image, we have an additional constraint – affine epipolar geometry.

Let $\boldsymbol{P}_A = [\boldsymbol{c}_1, \boldsymbol{c}_2, \boldsymbol{c}_3, \boldsymbol{c}_4]^{\mathsf{T}}$ and $\boldsymbol{P}'_A = [\boldsymbol{c}'_1, \boldsymbol{c}'_2, \boldsymbol{c}'_3, \boldsymbol{c}'_4]^{\mathsf{T}}$ be two affine camera matrices for the two control images. For an image point $\boldsymbol{x}$, its epipolar line has the equation [25]

$$\boldsymbol{x}'(\zeta) = (\boldsymbol{G}(\boldsymbol{x} - \boldsymbol{c}_4) + \boldsymbol{c}'_4) + \zeta\,(\boldsymbol{c}'_3 - \boldsymbol{G}\boldsymbol{c}_3)\,, \tag{15}$$

where $\boldsymbol{G} = [\boldsymbol{c}'_1, \boldsymbol{c}'_2][\boldsymbol{c}_1, \boldsymbol{c}_2]^{-1}$ is a 2×2 matrix and ζ is a scalar parameter. This equation implies that a corresponding point $\boldsymbol{x}'$ of $\boldsymbol{x}$ must be on its epipolar line. The four lines in Figure 2(b) are the epipolar lines of the four points $\{\boldsymbol{x}_0^{\mathrm{b}}, \boldsymbol{x}_1^{\mathrm{b}}, \boldsymbol{x}_2^{\mathrm{b}}, \boldsymbol{x}_3^{\mathrm{b}}\}$ specified in Section 4.1.

What we now have to do is to choose *three* image locations $\{\boldsymbol{x}_0^{\mathrm{b}'}, \boldsymbol{x}_1^{\mathrm{b}'}, \boldsymbol{x}_2^{\mathrm{b}'}\}$ on their epipolar lines $\{\boldsymbol{l}_0, \boldsymbol{l}_1, \boldsymbol{l}_2\}$. As soon as the three points are determined, they also provide a hyperbolic curve in the second image as shown in Figure 2(b). At this time, we need not choose $u_3^{\mathrm{b}'}$ because the last point $\boldsymbol{x}_3^{\mathrm{b}'}$ is determined as the intersection point of the hyperbola and the epipolar line of $\boldsymbol{x}_3^{\mathrm{b}}$. When there are two intersection points, we just need to select one of them. If there is no intersection, then we have to adjust the locations of the three specified pairs in order to get at least one.

Computing $\boldsymbol{P}'_{\mathrm{wp}}$ with the four image locations of the basis points of the world coordinate system finishes our embedding step (Equation (14)). Figure 2 also shows a wire-cube, which is the image of the unit cube through the computed weak-perspective virtual camera.

4.3 Affine Structure for the Control Points

We compute the affine coordinates $\{\boldsymbol{X}_i^{\mathrm{b}}\}_{i=0}^3$ of the four pairs of the control points $\{(\boldsymbol{x}_i^{\mathrm{b}}, \boldsymbol{x}_i^{\mathrm{b}'})\}_{i=0}^3$ in order to enable the virtual camera to move according to the motion of the real camera. In other words, the affine coordinates form an imaginary link between the real camera and the virtual camera.

Let ζ_i be the selected parameter of Equation (15) for the i-th pair $(\boldsymbol{x}_i^{\mathrm{b}}, \boldsymbol{x}_i^{\mathrm{b}'})$. Then, corresponding 3D affine coordinates $\boldsymbol{X}_i^{\mathrm{b}} = [X_i^{\mathrm{b}}, Y_i^{\mathrm{b}}, Z_i^{\mathrm{b}}, 1]^{\mathsf{T}}$ are given as follows:

$$Z_i^{\mathrm{b}} = \zeta_i \tag{16}$$

$$\begin{bmatrix} X_i^{\mathrm{b}} & Y_i^{\mathrm{b}} \end{bmatrix}^{\mathsf{T}} = \boldsymbol{G}^{-1}\left(\boldsymbol{x}_i^{\mathrm{b}} - \boldsymbol{c}_4 - Z_i \boldsymbol{c}_3\right). \tag{17}$$

The weak-perspective virtual camera at an arbitrary time k will be determined in Section 5.1 using the affine coordinates $\{\boldsymbol{X}_i^{\mathrm{b}}\}_{i=0}^3$.

4.4 Virtual Camera Decomposition

To use the matrix of the virtual camera in graphics rendering, we decompose it into three parts – $\boldsymbol{K}$, $\boldsymbol{R}$ and $\boldsymbol{t}$ – as in equations (1) and (2). We will use the notations of equations (3) - (5).

The scale components f_X and f_Y are given as follows:

$$f_X = \|\boldsymbol{b}_1\| \quad \text{and} \quad f_Y = \|\boldsymbol{b}_2\|, \tag{18}$$

which determine the calibration matrix $\boldsymbol{K}$. Then we have an orthonormal pair

$$\boldsymbol{r}_1 = \boldsymbol{b}_1 / f_X \quad \text{and} \quad \boldsymbol{r}_2 = \boldsymbol{b}_2 / f_Y, \tag{19}$$

which define a rotation matrix $\boldsymbol{R} = [\boldsymbol{r}_1, \boldsymbol{r}_2, \boldsymbol{r}_1 \times \boldsymbol{r}_2]^{\mathsf{T}}$. What is left is the translation vector $\boldsymbol{t} = [t_X, t_Y, t_Z]^{\mathsf{T}}$. The first two elements, t_X and t_Y, of $\boldsymbol{t}$ are given by $\boldsymbol{K}^{-1}\boldsymbol{a}_4$ but the last element t_Z can be determined at our disposal. Note that algebraically speaking, a view through an orthographic camera has nothing to do with the depth component. However, in a physical situation, one strict constraint is that every object must be in front of the camera, and according to our camera model we choose a positive depth t_Z in practice so that every graphics object lie in front of the virtual camera.

The calibration part $\boldsymbol{K}$ will be utilized in setting the viewing volume of the virtual camera in graphics machine and the motion parts $\boldsymbol{R}$ and $\boldsymbol{t}$ are for modelview transformation. The details are given in Section 5.2.

4.5 Projection by the Virtual Camera

During the procedure of embedding, we specify basis points of the world coordinate system. To facilitate this procedure, we need to see the results of our embedding. Given a virtual camera $\boldsymbol{P}_{\mathrm{wp}}$ we compute the image locations $\boldsymbol{x}_i^{\mathrm{b}} = \boldsymbol{P}_{\mathrm{wp}}\boldsymbol{E}_i$ of the vertices of the unit cube and examine our embedding result. An example (the wire-cube) is shown in Figure 2.

4.6 Placing Graphics Objects

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

5 Rendering Algorithm

Graphics rendering for the k-th video image consists of two steps:

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

5.1 Transferring the Virtual Camera

As the video camera moves, its affine camera matrix changes and, accordingly, its virtual camera must be computed or transferred to the k-th video image. The procedure is as follows:

1. Find the image matches $\{\boldsymbol{x}_{A,ki}\}_{i=1}^N$ in the k-th video image.
2. Compute the affine camera matrix $\boldsymbol{P}_{A,k}$ using the affine re-projection equation:

   $$\boldsymbol{x}_{A,ki} = \boldsymbol{P}_{A,k}\boldsymbol{X}_{A,i},\ i = 1, ..., N. \tag{20}$$

   Note that the affine coordinates $\boldsymbol{X}_{A,i}$ were already obtained from two control images using affine structure from motion algorithm.
3. Project the 3D affine coordinates $\{\boldsymbol{X}_i^{\mathrm{b}}\}_{i=0}^3$ of the control points into the k-th image through $\boldsymbol{P}_{A,k}$:

   $$\boldsymbol{x}_i^{\mathrm{b}} = \boldsymbol{P}_{A,k}\boldsymbol{X}_i^{\mathrm{b}},\ \ i = 0, ..., 3. \tag{21}$$

4. Compute the virtual camera $\boldsymbol{P}_{\mathrm{wp},k}$ using the projection $\{\boldsymbol{x}_i^{\mathrm{b}}\}_{i=0}^3$ of the control points (Equation (14)).
5. Decompose $\boldsymbol{P}_{\mathrm{wp},k}$ into $\boldsymbol{K}_k$, $\boldsymbol{R}_k$ and $\boldsymbol{t}_k$ as in Section 4.4.

5.2 Graphics Rendering

Figure 3 shows the three stages of coordinate transformation in our SGI graphics machine for making a view of graphics objects through the virtual camera $\boldsymbol{P}_{\mathrm{wp},k}$. The rotation $\boldsymbol{R}_k$ and translation $\boldsymbol{t}_k$ define the modeling transformation and $\boldsymbol{K}_k$ gives the orthographic viewing volume. The modelview matrix $\boldsymbol{M}_k$ for modeling transformation is given as $\boldsymbol{M}_k = \begin{bmatrix} \boldsymbol{R}_k & \boldsymbol{t}_k \\ \boldsymbol{0}_3^{\mathsf{T}} & 1 \end{bmatrix}$, which converts a 3D point $\boldsymbol{X}_W$, defined

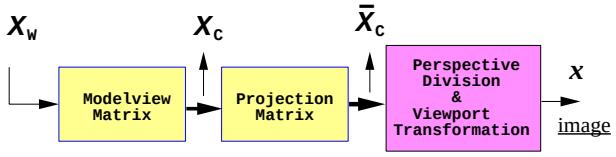
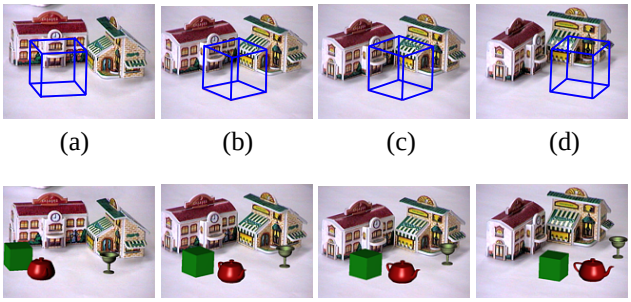


Figure 3: Stages of coordinate transformation in a graphics machine.



(a) (b) (c) (d)

(e) (f) (g) (h)

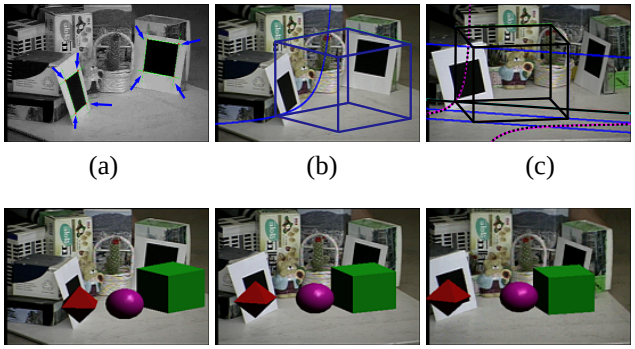
Figure 4: Experiment 1. (a)-(d) The unit cube was embedded in Figure 2 of Section 4. (e)-(h) Then the cube was utilized as the world coordinate system of the graphics world in order to put the three graphics objects.

with respect to the world coordinate system, to the point $\boldsymbol{X}_C$ with respect to the camera coordinate system. After that, a projection matrix is applied to yield clip coordinates $\bar{\boldsymbol{X}}_C$ by removing any part of the graphic objects outside the viewing volume which is a rectangular parallelepiped defined by $\boldsymbol{K}_k$.

6 Experiments

We implemented and tested our method and Figure 4 shows the result of the augmented images. The wire-cubes in Figure 4(a) – (d) are the result of the embedding shown in Figure 2 of Section 4. With respect to the embedded world coordinate system, we inserted three graphics objects and the images 4(e) – (h) show the results.

Figure 5 shows another augmentation result. The video sequence consisted of 251 images of 720×486 size. We selected eight lines in the first image of the sequence and the lines were then tracked automatically in the sequence. Eight intersection points (indicated by arrows in Figure 5(a)) were then used for affine structure and motion computation using the algorithm of Seitz and Dyer [23]. As the two control images for this sequence, we have chosen the first and the last images . In the embedding step, basis points were specified using the affine epipolar geometry (epipolar lines) and the orthogonality constraint (hyperbolas) as illustrated in Figure 5(b) and Figure 5(c). Graphics objects were located with respect to the embedded world coordinate system. In addition, we made the sphere move linearly in the y direction. Three frames of the augmentation result are shown in Figure 5(d) – (f).



(a) (b) (c)

(d) (e) (f)

Figure 5: Experiment 2. (a) The eight corner points indicated by arrows were computed as intersection points of the tracked edges. (b) This is the first control image showing the unit cube of the world coordinate system and the hyperbola generated for embedding. (c) Epipolar lines and the hyperbola are shown together with the unit cube. (d)-(f) Augmentation result. The sphere was designed to move in the direction of y-axis of the embedded world coordinate system.

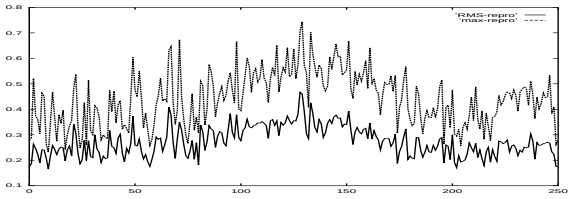


Figure 6: Re-projection error in pixel unit vs. frame number. The video consisted of 251 frames and our affine reconstruction showed that the maximum re-projection error was below a 0.8 pixel error during the sequence and the rms error was below a 0.5 pixel error.

Figure 6 shows a plot of the rms ϵ_k and maximum η_k of re-projection error with respect to the frame numbers in the pixel unit for the computation of the affine structure and camera. The maximum re-projection error for the whole sequence was below a 0.8 pixel error. Figure 7 shows the orthogonality $(\boldsymbol{b}_1 \cdot \boldsymbol{b}_2)$ of the virtual camera computed at each time step. The maximum was below 0.005 – about $0.3°$ deviation from $90°$. Figure 8 illustrates another experimental result.

7 Conclusion

We proposed a method for un-calibrated augmented reality of an orthographic projection camera. In order to make full use of computer graphics package, we designed a weak-perspective virtual camera for which we applied the orthogonality constraint. The image-based embedding enabled us to generate the intrinsic and extrinsic motion of the virtual camera, without metric camera calibration, in the same way as the real camera moves in 3D space and changes its internal parameters.

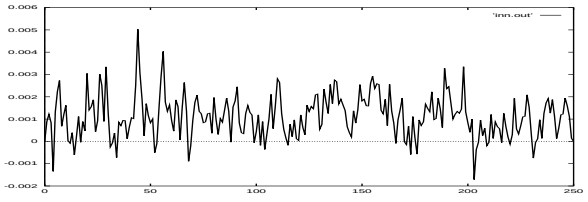


Figure 7: Orthogonality test plot. During the sequence, the maximum value of $\boldsymbol{b}_1 \cdot \boldsymbol{b}_2$ was below 0.005, which means that the angle between the two rows of the rotation matrix showed at most $0.3°$ deviation from the ideal value, $90°$.

(a) (b) (c)

Figure 8: Experiment 3. (a) A video image from the sequence (b), (c) Two augmented images.

References

[1] K. Åström, A. Heyden, F. Kahl, and M. Oskarsson. Structure and motion from lines under affine projections. In *Proc. ICCV*. 1999.

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

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

[4] P. Beardsley, I. Reid, A. Zisserman, and D. Murray. Active visual navigation using non-metric structure. In *Fifth Inter. Conf. Comp. Vision*, 1995.

[5] M.-O. Berger. Resolving occlusion in augmented reality: a contour based approach without 3D reconstruction. In *CVPR'97*, 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*, pages 90–94, 1996.

[7] Chu-Song Chen, Chi-Kuo Yu, and Yi-Ping Hung. New calibration-free approach for augmented reality based on parameterized cuboid structure. In *Proc. ICCV*. 1999.

[8] D.A. Forsyth, J.L. Mundy, A. Zisserman, and C.A. Rothwell. Using global consistency to recognise euclidean objects with an uncalibrated camera. In *CVPR*, June 1994.

[9] 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*, pages 3–12, April 1995.

[10] Stefan Grosskopf and Peter Neugebauer. The use of reality models in augmented reality applications. In *SMILE Workshop, in conjunction with ECCV'98*, June 1998.

[11] Gregory Hager. Calibration-free visual control using projective invariance. In *Fifth Inter. Conf. Comp. Vision*, June 1995.

[12] F. Kahl and A. Heyden. Structure and motion from points, lines and conics with affine cameras. In *Proc. 6th ECCV*, 1998.

[13] J.J. Koenderink and A.J. Van Doorn. Affine structure from motion. *J. of the Optical Society of America*, 8:377–385, 1991.

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

[15] J. M. Lawn and R. Cipolla. Robust egomotion estimation from affine motion-parallax. In J.-O. Eklundh, editor, *Proc. 3rd ECCV*, pages 205–210. Springer–Verlag, 1994.

[16] Joseph L. Mundy and Andrew Zisserman, editors. *Geometric Invariance in Computer Vision*. Artificial Intelligence series. MIT Press, 1992.

[17] Yuichi Ohta and Hideyuki Tamura, editors. *Mixed Reality - Merging real and virtual worlds*. Springer Verlag, 1999.

[18] S.W. Park, Yongduek. Seo, and Ki-Sang Hong. Real-time camera calibration for virtual studio. *Real-Time Imaging*, (6):433–448, Dec. 2000.

[19] L. Quan. Self-calibration of an affine camera from multiple views. *Int. Journal of Computer Vision*, 19(1):93–105, 1996.

[20] L. Quan and T. Kanade. Affine structure from line correspondences with uncalibrated affine cameras. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 19(8), August 1997.

[21] Luc Robert, Michel Buffa, and Martial Hébert. Weakly-calibrated stereo perception for rover navigation. In *Fifth Inter. Conf. Comp. Vision*, June 1995.

[22] Hagen Schumann, Silviu Burtescu, and Frank Siering. Applying augmented reality techniques in the field of interactive collaborative design. In *SMILE Workshop, in conjunction with ECCV'98*, June 1998.

[23] Steven M. Seitz and C.R. Dyer. Complete scene structure from four point correspondences. In *Proc. Int. Conf. on Computer Vision*, 1995.

[24] Yongduek. Seo and Ki-Sang Hong. Calibration-free augmented reality in perspective. *IEEE Trans. Visualization and Computer Graphics*, 6(4):346–359, Dec. 2000.

[25] Larry S. Shapiro, Andrew Zisserman, and Michael Brady. Motion from point matches using affine epipolar geometry. In *ECCV'94*, 1994.

[26] Rajeev Sharma and Jose Molineros. Computer vision-based AR for guiding manual assembly. *Presence: Teleoperators and Virtual Environments*, 6(3):292–317, June 1997.

[27] C. Tomasi and T. Kanade. Shape and motion from image streams under orthography: A factorization method. *International Journal of Computer Vision*, 9:137–154, 1992.

[28] 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*, 1(3):255–273, September 1995.

[29] C. Zeller and O. Faugeras. Applications of non-metric vision to some visual guided tasks. In *12th IAPR International Conference on Pattern Recognition*, October 9-13 1994.