2—1

Flexible Auto-Calibration and Its Application to Augmented Reality *

[1]Yongduek Seo† [2]Anders Heyden‡ [1]Ki-Sang Hong§
[1] Pohang University of Science and Technology (POSTECH), Korea
[2] MIG, Lund University, Sweden

Abstract

Practical problems are dealt with for augmented reality when the calibration is done through an auto-calibration. Projective motion of the camera is computed and upgraded to Euclidean via an auto-calibration algorithm. Video augmentation has two steps: embedding graphics objects into the real world using epipolar constraint and orthogonality constraint, and rendering the views of the virtual objects. This paper gives details of our approach and the results of real experiments.

1 Introduction

Under the name of **augmented reality** or **enhanced reality,** views of 3D graphics models have been mixed into the scenes of real video in order to have the effect of enhancing reality [1, 6]. The virtual objects used in augmented reality are generally 3D graphics models and the images of them are rendered by graphics machines and overlayed on real-video frames.

In this paper, we use an auto-calibration method for a single camera in the augmentation of real video. An example of the application of auto-calibration to video augmentation could be found in [2] where the calibration parameters of the camera was assumed to be constant with respect to time. In Section 2, we give some preliminary notations and a brief introduction to our method for projective reconstruction and auto-calibration. Since we apply a method of auto-calibration, we can compute the calibration and motion parameters. However, it does not provide the relationship between the camera coordinate system and the world coordinate system and thus we propose in Section 4 an image-based embedding method to compute the in-between transformation. Thus, we need not compute the 3D structure of the real scene. Two image locations in each of the images and one rotation angle are needed for embedding. The locations and directions of graphics objects can be specified with respect to the world coordinate system in modeling stage. The view of the graphics objects are then rendered in an SGI graphics machine and overlayed on the corresponding video image in Section 5. Real experimental results are presented in Section 6. When we have some estimation errors in the auto-calibration, we may have a different perspective view from what we intended in the embedding step. In Section 7 we investigate the effect of the estimation errors and non-zero skewness of estimated cameras. Finally, a concluding remark is given in Section 8.

*This work is partly supported by Korea Science and Engineering Foundation (KOSEF)

†dragon@postech.ac.kr
‡heyden@maths.lth.se
§hongks@postech.ac.kr

2 Auto-Calibration from Projective Reconstruction

A 2D image point is represented by a 3D homogeneous vector $\boldsymbol{x} = [u, v, 1]^{\mathrm{T}}$ with the third component being one. Also a 3D space point is represented by a 4D homogeneous vector $\boldsymbol{X}$. 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 $\boldsymbol{P}_k = [\tilde{\boldsymbol{P}}_k | \boldsymbol{p}_k]$, where $\tilde{\boldsymbol{P}}_k$ is the first 3×3 term of $\boldsymbol{P}_k$ and $\boldsymbol{p}_k$ is the last column of $\boldsymbol{P}_k$. The vector $\boldsymbol{X}_i^{\mathcal{P}}$ represents the projective 3D location of an image point $\boldsymbol{x}_{ki}$ and satisfies $\boldsymbol{x}_i \sim \boldsymbol{P}_k \boldsymbol{X}_i^{\mathcal{P}}$, where the notation $\sim$ denotes equality up to scale.

The real camera is modeled by a 3×4 matrix, denoted by $\boldsymbol{P}_k^{\mathcal{E}}$, which is decomposed into a calibration part and Euclidean motion part $\boldsymbol{P}_k^{\mathcal{E}} = \boldsymbol{K}_k[\boldsymbol{R}_k | \boldsymbol{t}_k]$, where $\boldsymbol{R}_k$ and $\boldsymbol{t}_k$ are rotation and translation, respectively, and $\boldsymbol{K}_k$ is a 3×3 calibration matrix of the form:

$$\boldsymbol{K}_k = \begin{bmatrix} \gamma f_k & f_k s_k & u_k \\ 0 & f_k & v_k \\ 0 & 0 & 1 \end{bmatrix}. \tag{1}$$

The effective focal length at time k of the real camera is f_k, and γ represents the aspect ratio. The image coordinate (u_k, v_k) denotes the principal point in image space and s_k is the skew factor of the camera. In this paper, we assume that the skew s_k of the real camera is always zero and call such a camera a **zero-skew** camera.

2.1 Projective Reconstruction and Auto-Calibration

We compute projective reconstruction using the factorization method of Heyden *et. al* [3] followed by a projective bundle adjustment. In result, we have the projective camera matrices $\boldsymbol{P}_k$ and structures $\boldsymbol{X}_i^{\mathcal{P}}$ that satisfy the projection equation $\boldsymbol{x}_{ik} = \boldsymbol{P}_k \boldsymbol{X}_i^{\mathcal{P}}$.

To recover Euclidean motion, we assume at first that the locations of the principal points are the image center. Then,

from the dual absolute conic projection equation, $\omega_k \sim P_k \Omega P_k^{\mathrm{T}}$, we can compute linearly the matrix Ω using the equations

$$p_k^1 \Omega p_k^{2\,\mathrm{T}} = p_k^1 \Omega p_k^{3\,\mathrm{T}} = p_k^2 \Omega p_k^{3\,\mathrm{T}} = 0, \qquad (2)$$

where p_k^m is the m-th row of P_k. The matrix $T_{4\times4}^{\mathcal{P}}$ is given by $T_{4\times4}^{\mathcal{P}} = V \tilde{D}^{\frac{1}{2}}$ after computing the SVD of $\Omega = VDV^{\mathrm{T}}$, setting the smallest singular value in D to zero, giving $\tilde{D}$, and finally setting the $(4,4)$-th element of $T_{4\times4}^{\mathcal{P}}$ to 1. After computing $T_{4\times4}^{\mathcal{P}}$, the principal points are now re-estimated by decomposing $P_k T_{4\times4}^{\mathcal{P}}$. Then, using the newly estimated principal points, we compute $T_{4\times4}^{\mathcal{P}}$ again. Eventually, this iteration refines the rank 3 condition of the matrix Ω and also the calibration parameters and motion parameters. Due to space limit, details can be found in [9].

Euclidean cameras are then computed by decomposing $P_k T_{4\times4}^{\mathcal{P}}$ via QR or Cholesky decomposition, and Euclidean structure $X_i^{\mathcal{E}}$ is obtained by normalizing the coordinates of $X_i^{\mathcal{E}} = (T_{4\times4}^{\mathcal{P}})^{-1} X_i^{\mathcal{P}}$ so that the last coordinate becomes 1. In computing the Euclidean quantities, there are two sign ambiguities because we have image matches only and the computed results are based on algebraic equations. Let us suppose that $X_i^{\mathcal{E}} = [X, Y, Z, 1]^{\mathrm{T}}$. Then we know that $x_{ki} \sim P_k^{\mathcal{E}} X_i^{\mathcal{E}} = [\tilde{P}_k^{\mathcal{E}} | p_k^{\mathcal{E}}] X_i^{\mathcal{E}}$. However, we can also find that $\hat{X}_i^{\mathcal{E}} = [-X, -Y, -Z, 1]^{\mathrm{T}}$ satisfies

$$x_{ki} \sim \hat{P}_k^{\mathcal{E}} \hat{X}_i^{\mathcal{E}} \sim [\tilde{P}_k^{\mathcal{E}} | -p_k^{\mathcal{E}}] \hat{X}_i^{\mathcal{E}} \sim [-\tilde{P}_k^{\mathcal{E}} | p_k^{\mathcal{E}}] \hat{X}_i^{\mathcal{E}}. \qquad (3)$$

That is, we have the same image point x_{ki} despite the different signs. This phenomenon occurs because they are equivalent algebraically [12, 5]. Therefore, we have to consider it physically in practice and the signs must be determined according to the model of the coordinate system. In particular, this problem must be considered for the process of graphics programming in order to get correct views of graphics objects.

Finally we have Euclidean motion and calibration: $P_k^{\mathcal{E}} = K_k [R_k | t_k]$. Our approach may give non-zero skews for each of the calibration matrices. However, those non-zero skews did not bring about a noticeable change in the real experiments. Section 7 gives a validation on this topic. On the other hand, our approach also reduces the computation time and complexity.

In Section 5, the calibration matrix will be used to define the **perspective viewing volume** of the graphics camera in a graphics machine and the motion part R_k and t_k will be used to locate graphics objects relatively in front of the camera.

3 Overview of the Video Augmentation Algorithm

This section gives a brief overview of our two step procedure, embedding and rendering. First, the embedding steps, shown in Figure 1, are as follows:

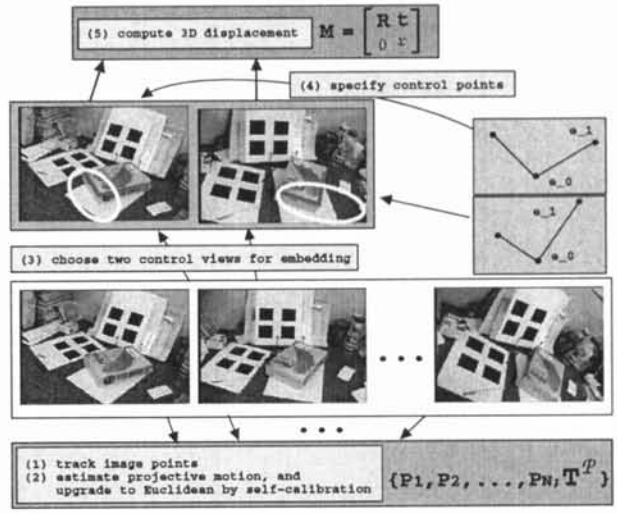

Figure 1: Overview of video augmentation: embedding procedure.

(1) Track feature points in the video sequence.

(2) Estimate the projective motion of the video camera $\{P_k\}$ using the point correspondences and do auto-calibration.

(3) Among the video images, choose **two control images** onto which the graphics world coordinate system will be embedded.

(4) Specify the locations of **two vertices E_0 and E_1** of the world coordinate system in each of the control images using the epipolar constraint. Then select rotation angle θ of the coordinate system.

(5) Compute the world-to-camera coordinate transformation M using the result of the previous step.

Second, the rendering steps, shown in Figure 2, are as follows:

(6) Compute the Euclidean camera for the k-th video image $\mathcal{I}_k$.

(7) (a) Set the graphics camera in the SGI graphics machine using $C_k = P_k^{\mathcal{E}} M$.

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

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

4 Embedding

Video augmentation is accomplished *by specifying the graphic world coordinate system* in two selected video images - two control images $\mathcal{V}_1$ and $\mathcal{V}_2$. For example, Figure

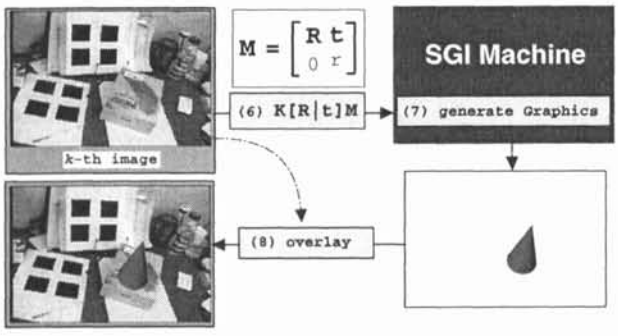

Figure 2: Overview of video augmentation: rendering procedure.

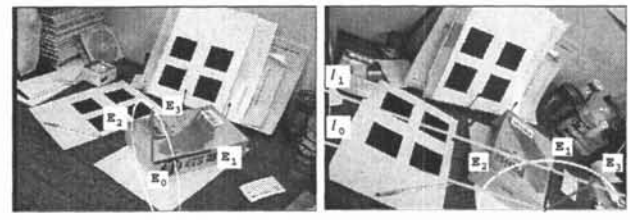

Figure 3: Embedding result. The two lines l_0 and l_1 show the epipolar lines corresponding to the image locations of E_0 and E_1, respectively. The circular arcs show the possible traces of the image location of the vertex E_2. We determined the rotation angle θ so that the Y-axis might be arranged with the edge of the book in the control images.

3 shows two selected images $\mathcal{I}_0$ and $\mathcal{I}_{270}$ from 274 video images. The embedding procedure consists of three steps:

1. We insert the world coordinate system of the graphics world into the first control image $\mathcal{V}_1$ by specifying the image locations of the two vertices $\boldsymbol{E}_0$ and $\boldsymbol{E}_1$ of the world coordinate system.
2. With the help of the epipolar geometry, we choose the two corresponding image locations of the vertices $\boldsymbol{E}_0$ and $\boldsymbol{E}_1$ in the second control image $\mathcal{V}_2$.
3. Finally, using the orthogonality of the coordinate system, we select the rotation angle θ, which determines the similarity transformation (rotation, translation and scaling) between the graphics world coordinate system and the real coordinate system.

Here $\boldsymbol{E}_0 = [0,0,0,1]^\top$ is the origin of the world coordinate system and $\boldsymbol{E}_1 = [1,0,0,1]^\top$ is the basis point on the X-axis.

Embedding via Epipolar Geometry: First, two image locations, $\boldsymbol{x}_0$ and $\boldsymbol{x}_1$ of the vertices are specified in the first control image $\mathcal{V}_1$. Then we have two epipolar lines in the second control image $\mathcal{V}_2$ from the epipolar constraint between the two control images. Now we choose two image locations $\boldsymbol{x}'_0$ and $\boldsymbol{x}'_1$ of the vertices on the corresponding epipolar lines $\boldsymbol{l}_0$ and $\boldsymbol{l}_1$ which are given by $\boldsymbol{l}_i = \boldsymbol{F}\boldsymbol{x}_i$.

This procedure allows us to determine the direction of the X-axis of the graphics world coordinate system in the camera coordinate system and the scale ratio, ρ, between the two coordinate systems. First, we compute the 3D coordinates $\boldsymbol{Y}_0$ and $\boldsymbol{Y}_1$ of the two specified image pairs $(\boldsymbol{x}_0, \boldsymbol{x}'_0)$ and $(\boldsymbol{x}_1, \boldsymbol{x}'_1)$ using the relationship: $\boldsymbol{x}_k = \boldsymbol{P}^{\mathcal{E}}\boldsymbol{Y}_k$, and $\boldsymbol{x}'_k = \boldsymbol{P}^{\mathcal{E}'}\boldsymbol{Y}_k$, for $k = 0, 1$, where $\boldsymbol{P}^{\mathcal{E}}$ and $\boldsymbol{P}^{\mathcal{E}'}$ are the Euclidean camera matrices corresponding to $\mathcal{V}_1$ and $\mathcal{V}_2$, respectively. Notice that $\boldsymbol{Y}_k$ is the 3D coordinate in the camera coordinate system of the vertex $\boldsymbol{E}_k$. The 4×4 similarity transformation $\boldsymbol{M}$ between the two coordinate systems defined by $\boldsymbol{Y}_k = \boldsymbol{M}\boldsymbol{E}_k$ is denoted as follows

$$\boldsymbol{M} = \begin{bmatrix} \boldsymbol{R} & \boldsymbol{t} \\ \boldsymbol{0} & 1/\rho \end{bmatrix}, \quad \text{where} \quad \boldsymbol{R} = \begin{bmatrix} \boldsymbol{r}_1 & \boldsymbol{r}_2 & \boldsymbol{r}_3 \end{bmatrix}. \quad (4)$$

We can find that $\rho = \|\boldsymbol{Y}_1 - \boldsymbol{Y}_0\|$, $\boldsymbol{r}_1 = \frac{1}{\rho}(\boldsymbol{Y}_1 - \boldsymbol{Y}_0)$, and $\boldsymbol{t} = \frac{1}{\rho}\boldsymbol{Y}_0$.

Embedding through Orthogonality: What left are the two columns of the rotation matrix $\boldsymbol{R}$. Note that the last degree of freedom implies the the rotation about the X-axis because we determined the origin and the direction of X-axis. Let us temporarily determine the rotation matrix by

$$\boldsymbol{r}_2 = \frac{[r_{21}, -r_{11}, 0]^\top}{\sqrt{r_{21}^2 + r_{11}^2}} \quad \text{and} \quad \boldsymbol{r}_3 = \boldsymbol{r}_1 \times \boldsymbol{r}_2, \quad (5)$$

where r_{i1} is the i-th element of $\boldsymbol{r}_1$. Let us denote this rotation matrix by $\hat{\boldsymbol{R}}$. Then the coordinates of $\boldsymbol{Y}_2$ and $\boldsymbol{Y}_3$ can be determined by the rotation angle θ about the X-axis

$$\boldsymbol{Y}_k(\theta) = \rho\left(\hat{\boldsymbol{R}}\boldsymbol{R}^X(\theta)\boldsymbol{E}_k + \boldsymbol{t}\right), \quad (6)$$

where

$$\boldsymbol{R}^X(\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{bmatrix}. \quad (7)$$

Here we determine the angle θ by examining the image location of $\boldsymbol{Y}_2$ which is given by $\boldsymbol{x}_2(\theta) = \boldsymbol{P}^{\mathcal{E}}\boldsymbol{Y}_2(\theta)$ and $\boldsymbol{x}'_2(\theta) = \boldsymbol{P}^{\mathcal{E}'}\boldsymbol{Y}_2(\theta)$.

Figure 3 shows an example of embedding procedure.

5 Rendering

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

1. Determination of the k-th graphics camera.
2. Setting graphics machine, rendering the corresponding graphics view, and overlaying it on the video image.

The graphics camera of the k-th video image is given by $\boldsymbol{C}_k$:

$$\boldsymbol{C}_k = \boldsymbol{P}_k^{\mathcal{E}}\boldsymbol{M} = \boldsymbol{K}_k[\boldsymbol{R}_k|\boldsymbol{t}_k]\begin{bmatrix} \boldsymbol{R} & \boldsymbol{t} \\ \boldsymbol{0} & 1/\rho \end{bmatrix} = \boldsymbol{K}_k[\boldsymbol{R}_k^G|\boldsymbol{t}_k^G] \quad (8)$$

The zero-skew calibration matrix $\boldsymbol{K}_k$ gives the **perspective viewing volume**, and the rotation $\boldsymbol{R}_k^G$ and translation $\boldsymbol{t}_k^G$ define the **modeling transformation** [7]. The generated view is then overlayed on the corresponding video image $\mathcal{I}_k$, which gives the effect of video augmentation.

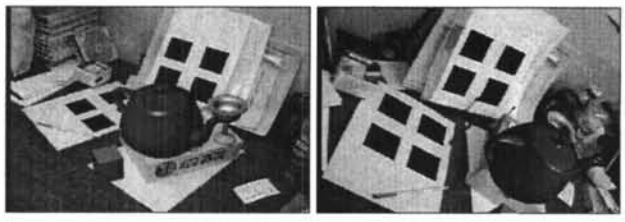

Figure 4: A result of video augmentation. A teapot, a cup and a cube were inserted.

6 Experiments

Figure 4 shows a result of video augmentation. This video sequence was captured by a hand-held video camera and composed of 274 frames (546 fields) in which lines of black rectangles were tracked to get 32 corner points and to estimate the projective motion of the video camera. The graphics world coordinate system was embedded into the camera coordinate system as described in Section 4. Finally, we defined a lighting source and three graphics objects with respect to the world coordinate system whose material properties were different from each other.

7 Discussion

When we estimate the Euclidean motion, the projective-to-Euclidean transformation and the calibration parameters will have some errors due to the noise in image matches. Among many kinds of errors, non-zero skews were dominant in our case due to the linear computation. The estimation error of the auto-calibration procedure brings about some problems as follows.

1. This is a fundamental problem. When the result of the auto-calibration comes to have different parameter values from the truth, the views of the embedded graphics object show a different perspective from the real camera.

2. When we have non-zero-skew calibration matrices, we may have the same effect as explained above. In addition, it causes the rotation matrix of the modelview matrix to be a non-orthogonal matrix. In order to get correct shading and color effects through a graphics system like the SGI machine, the rotation matrix should be orthogonal. However, the effect of non-zero skew values on the result of video augmentation was indistinguishable when the maximum skew was as much as 0.1, that is, for example, $f_k = 1000$, $f_k s_k = 100$, and $\gamma_k = 1$ in pixel unit.

A detail can be found in [8] as well as a comparison to non-metric augmented reality methods like [4, 11, 10].

8 Conclusion

We studied how to apply the result of auto-calibration to video augmentation. Investigated also were some practical problems in the application. The image-based embedding procedure through the epipolar geometry and orthogonality of the graphics world coordinate system was proposed. In the rendering procedure, the calibration and motion components were utilized to generate synthetic views. The resultant video clip showed that the graphics objects sticked well on the real object. Also we discussed on the effects of the auto-calibration error due to numerical computation and camera modeling.

References

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

[2] Olivier 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.

[3] Anders Heyden, Rikard Berthilsson, and Gunnar Sparr. An iterative factorization method for projective structure and motion from image sequences. *Image and Vision Computing*, 17(5), 1999.

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

[5] S. Laveau and O. Faugeras. Oriented porjective geometry for computer vision. In *Proc. ECCV'96*, 1996.

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

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

[8] Yongduek Seo. *Non-Metric Augmented Reality and Flexible Self-Calibration*. PhD thesis, Dept. of EE., Pohang University of Science and Technology (POSTECH), 2000.

[9] Yongduek Seo and Anders Heyden. Auto-calibration from the orthogonality constraint. In *Int. Conf. on Pattern Recognition, Barcellona, Spain*, 2000.

[10] Yongduek Seo and Ki-Sang Hong. Calibration-free augmented reality in perspective. *Accepted to IEEE Trans. Visualization and Computer Graphics*, 2000.

[11] Yongduek Seo and Ki Sang Hong. Weakly calibrated video-based augmented reality: embedding and rendering through virtual camera. In *IEEE and ACM International Symposium on Augmented Reality*, 2000.

[12] J. Stolfi. *Oriented Projective Geometry*. Academic Press, Inc., 1991.