

Calibration-free Line-based Tracking for Video Augmentation

Bolan Jiang (presenter)
Computer Science Department
California State University, Sacramento
Riverside Hall 5038, 6000 J Street
Sacramento, CA95825

jiangb@ecs.csus.edu

Submit to the 2006 International Conference on Computer Graphics and Virtual Reality (CGVR'06)

ABSTRACT

This paper presents a calibration-free line-based tracking method to augment virtual objects into a video sequence. The presented method uses a cube on the first image to calibrate camera parameters. Then the 3D model for the cube is refined by adding 3D lines to its surface. For the remaining images in the sequence, the image lines are automatically detected and matched with their 3D correspondents. The matched line pairs are used for pose estimation. The experiments show that the presented method can achieve robust performance for video augmentation without any prior knowledge of 3D model and camera parameters.

Keywords

Augmented Reality, Tracking, Pose Estimation, Registration, Camera Calibration.

1. INTRODUCTION

Augmented Reality (AR) systems can enhance a user's perception by inserting virtual objects into the user's view of real world. For example, video-based AR systems insert virtual objects into a video sequence. To make virtual objects appear at accurate locations on a video image, internal camera parameters and camera pose need to be estimated accurately. This paper presents a calibration-free line-based tracking method for placing virtual objects into a video sequence for man-made environments

Though a wide variety of tracking technologies have been developed for video augmentation [1, 2], each technology has its limitation.

Model-based methods [3, 7, 13, 14, and 22] detect 2D features on the images and match them with 3D model features. The pose is estimated by minimizing the errors between the detected features and the projection of the corresponding 3D model features. Model-based methods can achieve high accuracy. However, they require a priori knowledge of 3D model and most existing model-based

methods also need to know the camera internal parameters beforehand.

Structure-and-motion methods [8, 16] track 2D features (points or lines) across the sequence and can estimate camera poses, internal camera parameters and 3D model simultaneously. However, structure-and-motion methods cannot work on the sequences with pure rotation motions.

Chen et al [5] developed a calibration-free method, which neither requires a priori knowledge of 3D model nor puts any constraints on camera motion. Their method uses a cube to estimate the internal camera parameters, the camera pose and the sizes of the cube. The user needs to specify six corners of the cube on the first image. For the remaining images, the corners are tracked automatically. The requirement for the clear corner features, which can be tracked reliably, limits the usage of their method. For example, the corners for the building, which is shown in Figure 1, can not be tracked reliably.



Figure 1 A building without clear corners

In this paper, we present a calibration-free line-based tracking method for video augmentation. The presented method first uses a cube structure on a single image to

estimate the internal camera parameters, the pose and the 3D model for the cube. Then 3D lines are added to the surface of the cube. For the remaining images of the sequence, the 2D image lines are detected and matched with the 3D model lines and the poses are automatically estimated from the matched line pairs.

One contribution of this paper is to integrate calibration-free and line-based tracking methods. The presented method does not need a priori knowledge of 3D model and does not put any constraints on camera motion either. The significant difference between the presented method and the calibration-free method in [5] is that the presented method uses line features rather than points. Line features are prominent in man-made environments. For example, most buildings provide numerous line features. Lines often remain in the view much longer than points, because lines can extend over an entire structure; lines are useful, even when partially occluded; and lines can be localized accurately because there are often many supporting pixels along a line.

Another contribution of the presented method is its robust line detection and matching process. A critical process in model-based tracking systems is to detect image features and to match them with features of a 3D scene model. This feature detection and matching process is often computationally intensive and prone to failure. One kind of tracking methods [14, 19] learns the patterns of point-like features beforehand and uses the learned patterns to detect and match features. However, it is hard to learn patterns for line features. Another kind of tracking methods can improve the speed for the feature detection and matching process by using temporal coherence [3, 7, and 13]. Feature search is thereby limited to small displacements around predicted positions. Large feature displacements, arising from rapid motions or long intervals, often exceed the search space and result in failures for feature detection and matching. The presented method computes affine transform between frames and uses the affine transform to predict line features. The lines can be detected and matched robustly for rapid motions due to the accurate predictions.

The rest of this paper is organized as follows: Section 2 introduces the notation used in this paper; section 3 describes the initialization process and section 4 describes line-based tracking process. The experiment results are presented in section 5.

2. Notation

In this paper, we assume a pinhole camera model. A 3D point P is represented by the homogeneous coordinate $[X, Y, Z, 1]^T$ and a 2D image point p is represented

as $[u, v, 1]^T$. A 3D point and its corresponding projection on the image is related as

$$\lambda p = K[R_{3 \times 3} | t_{3 \times 1}]P \quad (1)$$

and

$$K = \begin{bmatrix} f_u & s & c_x \\ 0 & f_v & c_y \\ 0 & 0 & 1 \end{bmatrix} \quad (2)$$

where K is the matrix consisting of the internal camera parameters, R and t are rotation matrix and translation vector of the camera pose with respect to the world coordinate system, and λ is the depth. As in [5], we also assume $f_u = f_v = f$ and $s = 0$. And the internal camera parameters are assumed kept same for the whole sequence

In order to insert virtual objects into a video image, we need to know the internal camera parameters, which include focal length f and image center $[c_x, c_y]^T$, and the pose consisting of translation and rotation.

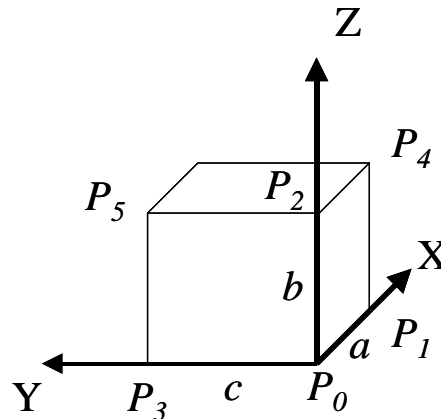


Figure 2 A cube with edge lengths a, b, c .

3. Initialization

The presented tracking method has two stages: initialization and automatic tracking. In the initialization stage, the user manually specifies six image points that form two adjacent faces of a cube. The geometry for these six points $\{P_0, P_1, P_2, P_3, P_4, P_5\}$ is illustrated in Figure 2. Then the internal camera parameters and the camera pose can be estimated based on the geometry constraints among these six points as in [5]. The parameters for the cube $\{a, b, c\}$, which are the lengths for the edges of the cube, can also be recovered. The estimated the focal length, the camera pose and the parameters for the cube are up to a scale factor. The

scale factor is assumed one here, which will not affect the visual effect for augmentation. A world coordinate is defined as following: the origin is at the point P_0 , the X axis is aligned with the edge P_0P_1 , the Y axis is aligned with the edge P_0P_3 and the Z axis is aligned with P_0P_2 . The 3D coordinates for the selected six points can be inferred from the values of a, b, c .

After the 3D coordinates for six corners are estimated, we also know the 3D coordinates for the lines that align with the edges of the cube. And those 3D lines can be used to estimate poses in the automatic tracking process. In order to increase the stability for the tracking process, other 3D lines are added to the two surfaces of the cube. To add a line to a surface s , first the user specifies two end points p_i and p_{i+1} on the image. Suppose the camera center is located at the point O, then the camera center O and the point p_i form a line l_i . The 3D point P_i , which corresponds to the 2D point p_i , can be estimated by intersecting the line l_i and the surface s . The 3D point P_{i+1} corresponding to the 2D point p_{i+1} can be estimated in a similar way.

4. Automatic Line-based tracking

After the initialization stage, the poses for the remaining images of the sequence are automatically estimated. The process to estimate pose for a video image is as following: 1) the image lines are detected and matched around the lines predicted by affine transform; 2) The pose is estimated based on the detected lines; 3) The estimated pose is used to detect new lines on the current image.

The camera pose is represented as

$$\vec{e} = (x, y, z, \dot{x}, \dot{y}, \dot{z}, \Delta\theta, \Delta\phi, \Delta\varphi, \dot{\theta}, \dot{\phi}, \dot{\varphi}), \quad (3)$$

where (x, y, z) represent position and $(\Delta\theta, \Delta\phi, \Delta\varphi)$ are the Euler angles representing the incremental rotation relative to the previous image. The global rotation for the previous image is represented as a quaternion $\alpha = (\alpha_w, \alpha_x, \alpha_y, \alpha_z)$.

4.1 2D Line Detection and Matching

To detect lines on the current image, the first step is to compute affine transform between previous and current image by the Fast Fourier transform (FFT) as in [18]. A point $p_i = (x_i, y_i, 1)$ on the previous image is predicted by the affine transform to the point $p_i^0 = (x_i^0, y_i^0, 1)$ on the current image as followings:

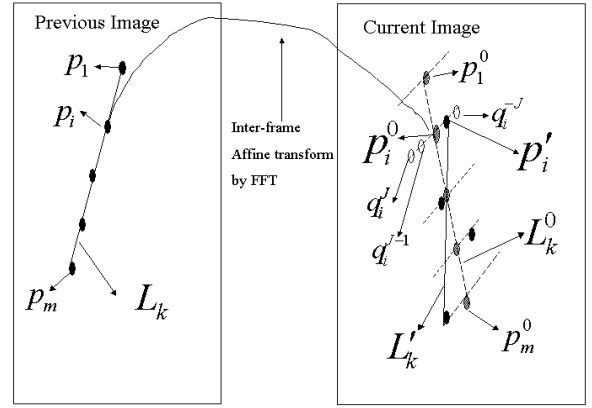


Figure 3 – 2D line detection and matching. The lines are searched around the lines predicted by affine transform

$$\begin{aligned} x_i^0 &= s \cos \theta (x_i - c_x) + s \sin \theta (y_i - c_y) - t_x + c_x \\ y_i^0 &= -s \sin \theta (x_i - c_x) + s \cos \theta (y_i - c_y) - t_y + c_y \end{aligned} \quad (4)$$

where s, θ, t_x, t_y are the affine parameters computed by FFT and $[c_x, c_y]^T$ is the image center.

As shown in Figure 3, a list of points $p_i (i = 1, \dots, m)$ on the line L_k , which are detected on the previous image, are predicted on the current image using equation (4). The predicted points are represented as $p_i^0 (i = 1, \dots, m)$, and the two predicted end points p_1^0 and p_m^0 form a predicted line L_k^0 . For each point p_i^0 on the predicted line L_k^0 , its corresponding point p_i' is searched in a 1D interval $\{q_i^j, j \in [-J, J]\}$. The center of the search interval is at p_i^0 and its direction is the normal to the predicted line L_k^0 . The search criterion is as follows:

$$p_i' = q_i^{j^*} = \arg \max_{q_i^j, j \in [-J, J]} |G_i^j| \quad (5)$$

subject to $\alpha < \frac{G_i^j}{G_i} < \beta$ and $|G_i^j| > \eta$.

G_i^j is the gradient in the direction of the normal to the predicted line for point q_i^j on the current image. G_i is the gradient for p_i in the direction of the normal to the line L_k on the previous image. The directional gradient is computed by the convolution masks defined in [4].



Figure 4 – Snapshot for initialization. (a) The cube (black) specified by the user (b) The manually specified 2D lines (black) on the surface of the cube (white). The white cube is the 3D model inserted into the image. (c) The recovered 3D model for the cube (gray) and the lines (blue) on the surface.

$\alpha (=0.5)$ and $\beta (=5)$ are used to ensure the similarity between the two gradients. η is the gradient threshold used to choose the pixels with a significant intensity discontinuity. The search distance J is chosen as 3 pixels in our implementation. The detected points are fit into a straight line L'_k by Hough transform.

Because the lines detected on the previous image are already matched with 3D model lines, the 2D-3D correspondences for the lines on the current image can be inferred from the 2D-2D correspondences.

4.2 Pose Estimation

After the image lines are detected and matched with the 3D model lines, the 3D model lines are projected on the image based on the estimated pose for the previous image. The projection errors between the image and model lines are minimized to refine the camera pose.

A 3D model line, which is represented by its two end points V_{i1} and V_{i2} , is projected on the image as

$$\begin{bmatrix} m_{ix} \\ m_{iy} \\ m_{iz} \end{bmatrix} = K^{-T} R^{-1} \left((V_{i1} - \begin{bmatrix} x \\ y \\ z \end{bmatrix}) \times (V_{i2} - \begin{bmatrix} x \\ y \\ z \end{bmatrix}) \right) \quad (6)$$

where K is the matrix consisting of the internal camera parameters, (x, y, z) is the camera position and R is the rotation matrix computed from the current estimates for the incremental rotation $(\Delta\theta, \Delta\phi, \Delta\varphi)$ and global orientation α . The projection error for a pair of detected and

projected lines is defined as the sum of the squared distances from the detected end points to the projected line as follows:

$$E_i^2 = (x_{i1}m_{ix} + y_{i1}m_{iy} + m_{iz})^2 / (m_{ix}^2 + m_{iy}^2) + (x_{i2}m_{ix} + y_{i2}m_{iy} + m_{iz})^2 / (m_{ix}^2 + m_{iy}^2) \quad (7)$$

Though lines can be predicted more accurately by affine transformation, it is still possible to have outliers. For example, there may be more than one line inside the neighborhood around a predicted line. The outliers are detected by a Least Median Square method [23]. Groups of three line pairs (m, n, k) are randomly drawn from the N matched line pairs. The chosen lines are used to estimate pose by minimizing $E_m^2 + E_n^2 + E_k^2$. Then the projection errors for all the pairs are computed. The group with the minimum median error is selected. The robust standard deviation estimate is given by

$$\sigma = 1.4826 [1 + 5/(N - 3)] \sqrt{M_J} \quad (8)$$

where $M_J = \text{med}_{i=1, \dots, N} r_i^2$, $r_i^2, i = 1, \dots, N$ are the projection

errors corresponding to the selected group. The pose is estimated by minimizing the following objective function

$$\hat{e} = \underset{\hat{e}}{\text{argmin}} \sum_{i=1}^N w_i E_i^2, \quad (9)$$

where $w_i = \begin{cases} 1 & \text{if } r_i^2 < (2.5\sigma)^2 \\ 0 & \text{otherwise} \end{cases}$.

This nonlinear optimization problem can be solved with the Levenberg-Marquardt algorithm [17]. The pose estimated for previous image is used as the initial value in the nonlinear optimization process.



Figure 5 – Snapshots for the line detection. Black lines are the lines detected on the current image. (a) 411th frame: The white lines detected on the previous image are far from their correspondents on the current image due to large image displacements. The lines (black) on the current image are detected accurately due the accurate prediction (yellow lines) by affine transform. (b) 983rd frame: Some lines were lost due to the change of viewpoint. (c) 1023rd frame: The lost lines reappeared and were detected accurately.

After pose estimation, the incremental rotation angles are integrated into the global rotation represented by the quaternion. And the incremental rotation angles are set to zeros.

The detected lines with zero weight are treated as outliers and will not be used for the line detection and matching process for the next image.

4.3 Reappearing Line Detection

When the user moves around, some previously detected lines may be lost and reappear later. Finding reappearing lines is particularly critical as many factors can cause the loss of currently tracked lines. Unless new lines are constantly sought, the system will eventually lose the set of starting lines, causing pose loss.

After pose estimation, the 3D model lines are projected on the image based on the current pose estimate. The hidden lines are removed by rendering the model and lines in OpenGL. For each projected line without a 2D correspondent, a new line search is performed in its neighborhood. For each point on a projected line, a 1D interval $[-J, J]$ centered at the point and perpendicular to the projected line is searched. The point with the maximum gradient in the direction perpendicular to the predicted line is selected as a line point. All the selected points are fit into a straight line by Hough transform.

5. Experiments

To test the presented tracking method, a video sequence was captured when a user was walking on a parking lot and looking at the buildings besides the parking lot. The video sequence was captured by a CCD video camera (Sony XC-

999 with 6mm lens) and an Osprey frame grabber. The image size is 320x240 pixels. The capture speed was about 20frames/second and about 2600 frames were captured. The tracking algorithm was run on a PC with a Pentium 4 1.6GHz CPU. The performance for the presented tracking algorithm is about 1.5frames/second. Currently, the affine transforms are computed using Matlab, which runs about 2frames/seconds. The performance for line detection and pose estimation (without affine transform computation) is about 6frames/second.

In the initialization stage, the user manually selects six corners of a cube on the first image, which is the black cube shown in Figure 4(a). The internal camera parameters, 3D geometry for the cube and the pose were estimated based the cube. The 2D lines on the surfaces of the cube, which are the black lines shown in Figure 4(b), were specified manually by users. The 3D coordinates for those lines were estimated as described in section 3. The 3D model for the cube and lines on its surface are shown in Figure 4(c). Due to the walking people and moving vehicles, it is hard to detect the lines accurately on the lower part of the building. So the lines on the lower part of the building were not included in the 3D line set.

After the initialization, the lines were detected on each image and matched with their corresponding 3D model lines automatically. The pose was estimated by the matched line pairs as described in section 4. As shown in the Figure 5(a), the white lines are the lines detected on the previous image, which are far from their correspondents (black lines) on the current image due to the large image displacements. On the other hand, the yellow lines that are predicted by affine transform are close to the correct locations. So it can improve accuracy for line detection and

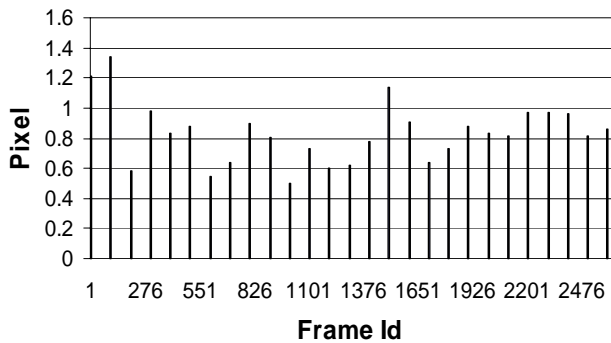


Figure 6 – The average re-projection errors on the key frames

matching by using affine transform for prediction. As shown in Figure 5(b), some lines were lost due to the change of viewpoint. Those lost lines reappeared in Figure 5(c) and were detected accurately.

To evaluate the tracking performance, 26 key frames were chosen evenly from the sequence. The 2D image lines are detected and matched manually with the 3D model lines on each key frame. The average of the re-projection errors as defined in equation (7) was computed for each key frame. As shown in Figure 6, the maximum average error is below 1.4 pixels.

To analyze the visual effect of the presented method, the 3D wire-frame model of the cube and a sculpture were inserted into the video sequence based on the estimated poses. Several snapshots are shown in Figure 7, in which the 3D models in white color are aligned accurately with the images.

6. Conclusion

This paper presents a calibration-free line-based tracking method for video augmentation. The presented method does not require a priori knowledge of 3D model or camera parameters. In the initialization stage, a cube is manually specified to calibrate camera parameters and to build 3D models of the lines on the surface of the cube. Those 3D model lines are automatically detected on the remaining images to estimate poses. The affine transforms between images are used to predict 2D lines, which can improve the robustness of line detection and matching process. The experiments show that the presented tracking system can achieve robust performance even with large image motions. And the virtual objects can be placed accurately into an uncalibrated video sequence.

7. REFERENCES

- [1] R. Azuma, Y. Baillet, R. Behringer, S. Feiner, S. Julier, B. MacIntyre, "Recent Advances in Augmented Reality", *IEEE Computer Graphics & Application* 21, 6 (Nov/Dec 2001).
- [2] R. T. Azuma, "A Survey of Augmented Reality", *Presence: Teleoperators and Virtual Environments*, vol. 6, 1997, pp355-385.
- [3] R. Behringer, J. Park, and V. Sundareswaran, "Model-based visual tracking for outdoor augmented reality", *Proc. ISMAR '02*
- [4] P. Bouthemy, "A Maximum Likelihood Framework for Determining Moving Edges", *IEEE PAMI*, Vol. 11, No. 5, May 1989.
- [5] C. Chen, C. Yu and Y. Hung, "New Calibration-free Approach for Augmented Reality Based on Parameterized Cuboid Structure", *ICCV 1999*.
- [6] K. Chia, A. Cheok, and S. Prince, "Online 6DOF Augmented Reality Registration from Natural Features", In *Proc. ISMAR 2002*.
- [7] T. Drummond, R. Cipolla, "Real-time Visual Tracking of Complex Structures", *IEEE PAMI* Vol. 24, No. 7, July 2002.
- [8] R. Hartley, A. Zisserman, "Multiple View Geometry in Computer Vision", Cambridge University Press, 2004
- [9] T. Höllerer, S. Feiner, and J. Pavlik, "Situated Documentaries: Embedding Multimedia Presentations in the Real World", *Proc. of ISCW' 99*, pp79-86.
- [10] B. Jiang, U. Neumann and S. You, "A Robust Hybrid Tracking System for Outdoor Augmented Reality", *IEEE VR2004*.
- [11] K.N. Kutulakos and J.R. Vallino, "Calibration-free Augmented Reality", *IEEE Transaction on Visualization and Computer Graphics*, vol. 4, pp.1-20, 1998.
- [12] S. Keren, I. Shimshoni and A. Tal, "Placing three-dimensional models in an uncalibrated single image of an architectural scene", *ACM VRST 2002*.
- [13] E. Marchand, P. Bouthemy, F. Chaumette and V. Moreau, "Robust real-time visual tracking using a 2D-3D model-based approach", *Proc. ICCV' 99*, pages 262-268.
- [14] J. Park, B. Jiang and U. Neumann, "Vision-based Pose Computation: Robust and Accurate Augmented Reality Tracking", *Proc. IWAR'99*, pp. 3-12.
- [15] W. Piekarski, B. Gunther, and B. Thomas, "Integrating Virtual and Augmented Realities in an Outdoor Application", *Proc. of IWAR' 99*, pp. 45-54.
- [16] M. Pollefeys, R. Koch, and L.V. Gool, "Self-Calibration and Metric Reconstruction in spite of Varying and Unknown Internal Camera Parameters", *Proc. ICCV'98*, pp. 90-95, 1998.

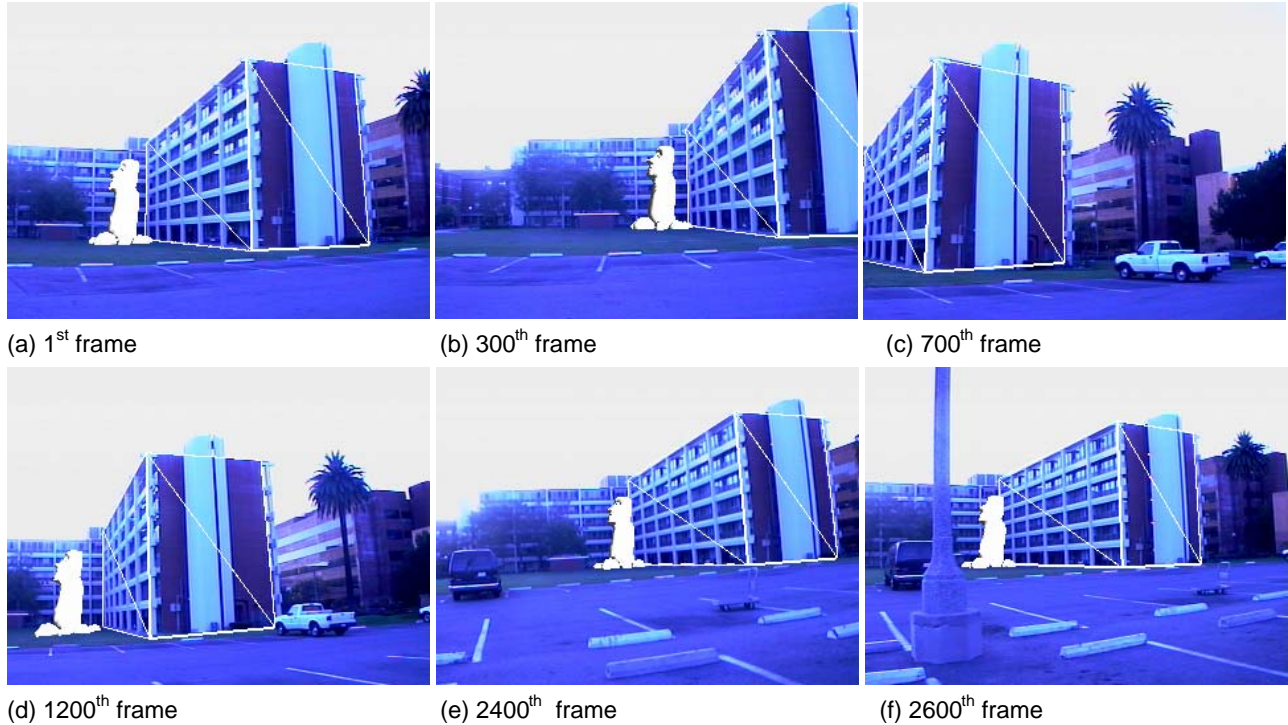


Figure 7 Snapshots for video augmentation. A white sculpture and the white wire-frame model of the cube were inserted into the video sequence.

- [17] W. Press, S. Teukolsky, S. Woo, and B. Flannery, "Numerical Recipes in C", Cambridge University Press, 1992.
- [18] B.S. Reddy, B.N. Chatterji, "An FFT-Based Technique for Translation, Rotation, and Scale-Invariant Image Registration", IEEE Transactions on Image Processing, Vol. 5. No. 8, August 1996.
- [19] M. Ribo, P. Lang, H. Ganster, M. Bradner, G. Stock, and A. Pinz, "Hybrid Tracking for Outdoor Augmented Reality Applications", IEEE Computer Graphics and Applications vol. 22. Nov/Dec. 2002, pp54-63.
- [20] Y. Seo, M.H. Ahn, and K.S. Hong, "Video Augmentation by Image-based Rendering under the Perspective Camera Model", Proc. ICPR'98, pp.1694-1697, 1998.
- [21] D. Stricker, T. Kettenbach, "Real-time and Marker-less Vision-based Tracking for Outdoor Augmented Reality Applications", IEEE and ACM ISMAR 2001.
- [22] G. Welch, G. Bishop, "SCAAT: Incremental Tracking with Incomplete Information", SIGGRAPH 97, pp. 333-344.
- [23] Zhengyou Zhang, "Parameter Estimation Techniques: A Tutorial with Application to Conic Fitting", INRIA Research Report No. 2676, 1995.