To appear in: Proc. 1996 IEEE Virtual Reality Ann. Int. Symp. (VRAIS'96)

Affine Object Representations for Calibration-Free Augmented Reality

Kiriakos N. Kutulakos
kyros@cs.rochester.edu

James Vallino
vallino@cs.rochester.edu

Computer Science Department
University of Rochester
Rochester, NY 14627-0226

Abstract

We describe the design and implementation of a video-based augmented reality system capable of overlaying three-dimensional graphical objects on live video of dynamic environments. The key feature of the system is that it is completely uncalibrated: it does not use any metric information about the calibration parameters of the camera or the 3D locations and dimensions of the environment's objects. The only requirement is the ability to track across frames at least four feature points that are specified by the user at system initialization time and whose world coordinates are unknown. Our approach is based on the following observation: Given a set of four or more non-coplanar 3D points, the projection of all points in the set can be computed as a linear combination of the projections of just four of the points. We exploit this observation by (1) tracking lines and fiducial points at frame rate, and (2) representing virtual objects in a non-Euclidean, affine *frame of reference that allows their projection to be computed as a linear combination of the projection of the fiducial points.*

1. Introduction

Recent advances in display technology and graphics rendering techniques have opened up the possibility of mixing live video from a camera with computer-generated graphical objects registered in a user's three-dimensional environment. Applications of this powerful visualization tool include overlaying clinical 3D data with live video of patients during surgical planning [3, 17, 28] as well as developing three-dimensional user interfaces [14]. While several approaches have demonstrated the potential of augmented reality systems for human-computer interaction [11], the process of embedding three-dimensional "virtual" objects into a user's environment raises three issues unique to augmented reality:

$$\begin{tikzcd}
\text{Object-to-World} \arrow[d] \\
\text{World-to-Camera} \arrow[d] \\
\text{Camera-to-Image}
\end{tikzcd}$$

Figure 1. Calibration of a video-based augmented reality system requires specifying three transformations that relate the coordinate systems of the virtual objects, the environment, the camera, and the image it produces. This paper focuses on how specification of the top two transformations in the figure can be avoided.

- *Establishing 3D geometric relationships between physical and virtual objects:* The locations of virtual objects must be initialized in the user's environment before user interaction can take place.
- *Rendering virtual objects:* Realistic augmentation of a 3D environment can only be achieved if objects are continuously rendered in a manner consistent with their assigned location in 3D space and the camera's viewpoint.
- *Dynamically modifying 3D relationships between real and virtual objects:* Animation of virtual objects relative to the user's environment should be consistent with the objects' assigned physical properties (e.g.,rigidity).

At the heart of these issues is the ability to describe the camera's motion, the user's environment and the embedded virtual objects in the same frame of reference. Typical approaches rely on 3D position tracking devices [31] and precise calibration [27] to ensure that the entire sequence of transformations between the internal reference frames of the virtual and physical objects, the camera tracking device, and the user's display is known exactly (Figure 1). In practice, camera calibration and position tracking are prone to errors which accumulate in the augmented display. Furthermore, initialization of virtual objects requires additional calibration stages [17,22], and camera calibration must be performed whenever its intrinsic parameters (e.g., focal length) change.

Our approach is motivated by recent approaches to video-based augmented reality that reduce the effects of calibration errors through real-time processing of the live video images viewed by the user [31]. These approaches track a small number of fiducial points in the user's 3D environment to either weaken the system's calibration requirements [22,28] or to compensate for calibration errors in the system [4].

This paper describes the design and implementation of a novel video-based augmented reality system. The key feature of the system is that it allows operations such as virtual object placement, real-time rendering, and animation to be performed without relying on any information about the calibration parameters of the camera, the camera's motion, or the 3D locations and dimensions of the environment's objects. The only requirement is the ability to track across frames at least four features (points or lines) that are specified by the user during system initialization and whose world coordinates are unknown.

Our approach is based on the following observation, pointed out by Koenderink and van Doorn [18] and Ullman and Basri [29]: Given a set of four or more non-coplanar 3D points, the projection of all points in the set can be computed as a linear combination of the projections of just four of the points. We exploit this observation by (1) tracking lines and fiducial points at frame rate, and (2) representing virtual objects so that their projection can be computed as a linear combination of the projection of the fiducial points. The resulting *affine virtual object representation* is a non-Euclidean representation in which the coordinates of points on a virtual object are relative to an affine reference frame defined by the fiducial points.

Affine object representations have been a topic of active research in computer vision in the context of 3D reconstruction [18,30] and recognition [20]. While our results draw heavily from this research, the use of affine object models in the context of augmented reality has not been previously studied. Here we show that placement of affine virtual objects, visible-surface rendering, as well as animation can be performed efficiently using simple linear methods that operate at frame rate and exploit the ability of the augmented reality system to interact with its user.

Very little work has been published on augmented reality systems that reduce the effects of calibration errors through real-time processing of the live video stream [4,22,28,31]. To our knowledge, only two systems have been reported [22,28] that operate without specialized camera tracking devices and without relying on the assumption that the camera is always fixed [11] or perfectly calibrated. The system of Mellor [22] is capable of overlaying 3D medical data over live video of patients in a surgical environment. The system tracks circular features in an known 3D configuration to invert the object-to-image transformation using a linear method. Even though the camera does not need to be calibrated at all times, camera calibration is required at system initialization time and the exact 3D location of the tracked image features is recovered using a laser range finder. The most closely related work to our own is the work of Uenohara and Kanade [28]. Their system allows overlay of planar diagrams onto live video by tracking feature points in an unknown configuration that lie on the same plane as the diagram. Calibration is avoided by expressing diagram points as linear combinations of the feature points. Their study did not consider uncalibrated rendering, animation and interactive placement of 3D virtual objects.

Our approach both generalizes and extends previous approaches in three ways. First, we show that by representing virtual objects in an affine reference frame and by performing computer graphics operations such as projection and visible-surface determination directly on affine models, the entire video overlay process is described by a single 4×4 homogeneous *view transformation matrix* [15]. Furthermore, the elements of this matrix are simply the image x- and y-coordinates of feature points. This not only enables the efficient estimation of the view transformation matrix but also leads to the use of optimal estimators such as the Kalman filter [2,5,32] to both track the feature points and compute the matrix. Second, the use of affine models leads to a simple *through-the-lens method* [16] for interactively placing virtual objects within the user's 3D environment and for animating them relative to other physical or virtual objects. Third, efficient execution of computer graphics operations on affine virtual objects is achieved by implementing affine projection computations directly on dedicated graphics hardware.

2. Geometrical Foundations

Accurate projection of a virtual object requires knowing precisely the combined effect of the object-to-world, world-to-camera and camera-to-image transformations [15]. In homogeneous coordinates this projection is described by the equation

$$\begin{bmatrix} u \\ v \\ h \end{bmatrix} = \mathbf{P}_{3\times4}\mathbf{C}_{4\times4}\mathbf{O}_{4\times4}\begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix} \qquad (1)$$

where $[x\,y\,z\,w]^T$ is a point on the virtual object, $[u\,v\,h]^T$ is its projection, $\mathbf{O}_{4\times4}$ and $\mathbf{C}_{4\times4}$ are the matrices corresponding to the object-to-world and world-to-camera homogeneous transformations, respectively, and $\mathbf{P}_{3\times4}$ is the matrix modeling the object's projection onto the image plane.

Eq. (1) implicitly assumes that the 3D coordinate frames corresponding to the camera, the world, and the virtual object are not related to each other in any way. The main idea of our approach is to represent both the object and the camera in a single, *non-Euclidean* coordinate frame defined by fiducial points that can be tracked across frames in real time. This change of representations, which amounts to a 4×4 homogeneous transformation of the object and camera coordinate frames, has two effects:

- It simplifies the projection equation. In particular, Eq. (1) becomes

$$\begin{bmatrix} u \\ v \\ h \end{bmatrix} = \Pi_{3\times 4} \begin{bmatrix} x' \\ y' \\ z' \\ w' \end{bmatrix}, \qquad (2)$$

  where $[x' \; y' \; z' \; 1]^T$ are the transformed coordinates of point $[x \, y \, z \, 1]^T$ and $\Pi_{3\times 4}$ models the combined effects of the change in the object's representation as well as the object-to-world, world-to-camera and projection transformations.

- It allows the elements of the *projection matrix*, $\Pi_{3\times 4}$, to simply be the image coordinates of the fiducial points. Hence, the image location of the fiducial points contains all the information needed to project the virtual object; the 3D position and calibration parameters of the camera as well as the 3D location of the fiducial points can be unknown. Furthermore, the problem of determining the projection matrix corresponding to a given image becomes trivial.

To achieve these two effects, we use results from the theory of affine-invariant object representations which was recently introduced in computer vision research. These representations become important because they can be constructed for any virtual object without requiring any information about the object-to-world, world-to-camera, or camera-to-image transformations. The only requirement is the ability to track across frames a few fiducial points, at least four of which are not coplanar. The basic principles behind these representations are briefly reviewed next. We will assume in the following that the camera-to-image transformation can modeled using the *weak perspective projection* model [25] (Figure 2). [1]

[1]This assumption is not crucial, however. The analysis presented in this paper can be directly generalized to account for the perspective projection model. In particular, Eq. (2) still holds when the object is represented in a *projective* frame of reference defined by 5 fiducial points [12]. The use of projective-invariant representations for re-projecting virtual objects under perspective projection is currently under investigation.

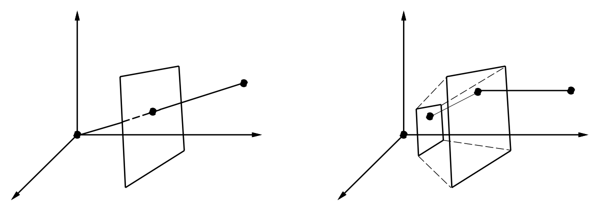


Figure 2. Projection models. (a) Perspective projection. Point $p = [x \; y \; z \; 1]^T$ is projected to $p = [fx/d \; fy/d \; 1]^T$, where d is the point's distance from the center of projection and f is the camera's focal length. (b) Weak perspective projection. Point p is projected to $[fx/d_{\text{avg}} \; fy/d_{\text{avg}} \; 1]^T$ where d_{avg} is the average distance of the object's points from the center of projection. Weak perspective projection amounts to first projecting the object orthographically and then scaling its image by f/d_{avg}; it is a good approximation to perspective projection when the camera's distance to the object is much larger than the size of the object itself.

2.1. Affine Point Representations

A basic operation in our method for computing the projection of a virtual object is that of *re-projection* [6,26]: given the projection of a collection of 3D points at two positions of the camera, compute the projection of these points at a third camera position. Affine point representations allow us to re-project points without knowing the camera's position and without having any metric information about the points (e.g., 3D distances between them).

In particular, let $p_1, \ldots, p_n \in \Re^3, n \geq 4$, be a collection of points, at least four of which are not coplanar. An *affine representation* of those points is a representation that does not change if the same non-singular linear transformation (e.g., translation, rotation, scaling) is applied to all the points. Affine representations consist of three components: The *origin*, which is one of the points $p_1, \ldots, p_n$; the *affine basis points*, which are three points from the collection that are not coplanar with the origin; and the *affine coordinates* of the points $p_1, \ldots, p_n$, expressing the points $p_i, i = 1, \ldots, n$ in terms of the origin and affine basis points. We use the following two properties of affine point representations [18,23,30] (Figure 3):

Property 1 (Re-Projection Property) *When the projection of the origin and basis points is known in an image I_m, we can compute the projection of a point p from its affine coordinates:*

$$\begin{bmatrix} u_p^m \\ v_p^m \\ 1 \end{bmatrix} = \begin{bmatrix} u_{\mathbf{b}_1}^m & u_{\mathbf{b}_2}^m & u_{\mathbf{b}_3}^m & u_{p_o}^m \\ v_{\mathbf{b}_1}^m & v_{\mathbf{b}_2}^m & v_{\mathbf{b}_3}^m & v_{p_o}^m \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \qquad (3)$$

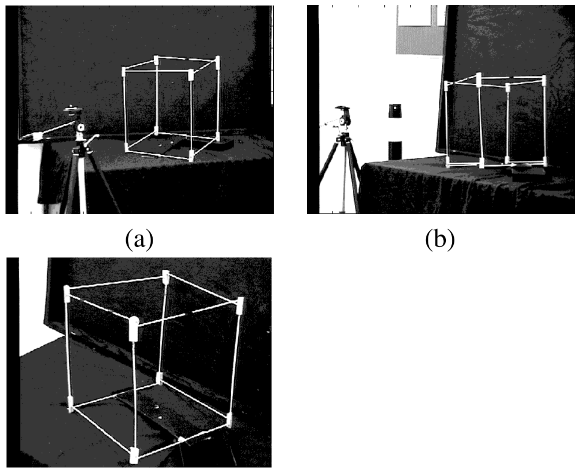


Figure 3. Properties of affine point representations. Three views of a real wireframe object are shown. Points $p_o, p_{b_1}, p_{b_2}, p_{b_3}$ define an affine coordinate frame within which all world points can be represented: Point p_o is the origin, and points $p_{b_1}, p_{b_2}, p_{b_3}$ are the basis points. The affine coordinates of a fifth point, p, are computed from its projection in images (a) and (b) using Property 2. p's projection in image (c) can then be computed from the projections of the four basis points using Property 1.

where $[u_p^m \; v_p^m \; 1]^T$ *is the projection of* p*;* b_1, b_2, b_3 *are the basis points;* $[u_{p_o}^m \; v_{p_o}^m \; 1]^T$ *is the projection of the origin; and* $[x \; y \; z \; 1]^T$ *is the homogeneous vector of* p*'s affine coordinates.*

Property 1 tells us that the projection process for any camera position is completely determined by the matrix collecting the image coordinates of the affine basis points in Eq. (3). This equation, which makes precise Eq. (2), implies that if the affine coordinates of a virtual object are known, the object's projection can be trivially computed by tracking the affine basis points. The following property suggests that it is possible, in principle, to extract the affine coordinates of an object without having any 3D information about the position of the camera or the affine basis points:

Property 2 (Affine Reconstruction Property) *The affine coordinates of* $p_1, \ldots, p_n$ *can be computed using Eq. (3) when their projection along two viewing directions is known.*

Intuitively, Property 2 shows that this process can be inverted if at least four non-coplanar 3D points can be tracked across frames as the camera moves. More precisely, given two images I_1, I_2, the affine coordinates of a point p can be recovered by solving an over-determined system of equations

$$\begin{bmatrix} u_p^1 \\ v_p^1 \\ u_p^2 \\ v_p^2 \end{bmatrix} = \begin{bmatrix} u_{\mathbf{b}_1}^1 & u_{\mathbf{b}_2}^1 & u_{\mathbf{b}_3}^1 & u_{p_o}^1 \\ v_{\mathbf{b}_1}^1 & v_{\mathbf{b}_2}^1 & v_{\mathbf{b}_3}^1 & v_{p_o}^1 \\ u_{\mathbf{b}_1}^2 & u_{\mathbf{b}_2}^2 & u_{\mathbf{b}_3}^2 & u_{p_o}^2 \\ v_{\mathbf{b}_1}^2 & v_{\mathbf{b}_2}^2 & v_{\mathbf{b}_3}^2 & v_{p_o}^2 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}. \quad (4)$$

In Section 5 we consider how this property can be exploited to interactively "position" a virtual object within an environment in which four fiducial points can be identified and tracked.

3. Affine Augmented Reality

The previous section suggests that once the affine coordinates of points on a virtual object are determined relative to four features in the environment, the points' projection becomes trivial to compute. The central idea in our approach is to ignore the original representation of the object altogether and perform all graphics operations with the new, affine representation of the object. This representation is related to the original object-centered representation by a homogeneous transformation: if p_1, p_2, p_3, p_4 are the coordinates of four non-coplanar points on the virtual object expressed in the object's coordinate frame and p_1', p_2', p_3', p_4' are their corresponding coordinates in the affine frame, the two frames are related by an invertible, homogeneous *object-to-affine* transformation $\mathbf{A}$ such that

$$[p_1' \; p_2' \; p_3' \; p_4'] = \mathbf{A}[p_1 \; p_2 \; p_3 \; p_4]. \quad (5)$$

The affine representation of virtual objects is both powerful and weak: it allows us to compute an object's projection without requiring information about the camera's position or calibration. On the other hand, this representation captures only properties of the virtual object that are maintained under affine transformations—metric information such as the distance between an object's vertices and the angle between object normals is not captured by the affine model. Nevertheless, our purpose is to show that the information that *is* maintained is sufficient for correctly rendering the virtual object. The augmented reality system we are developing based on this principle currently supports the following operations:

- **Object rendering:** Efficient and realistic rendering of virtual objects requires that operations such as point projection and z-buffering can be performed accurately and can exploit graphics rendering hardware when available. This is made possible by describing the entire projection process with a view transformation matrix expressed directly in terms of measurements in the image (Section 4).
- **Interactive placement of virtual objects:** This operation allows virtual objects to be "placed" in the environment with a simple through-the-lens interaction of the user with the system. The operation effectively determines the object-to-affine

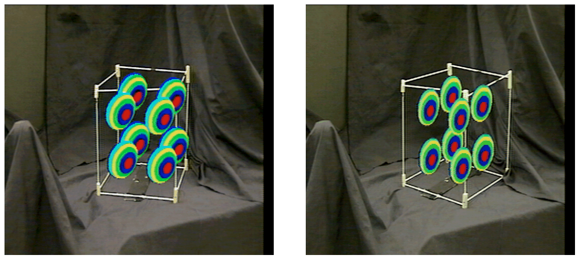


Figure 4. Real time visible-surface rendering of texture-mapped affine virtual objects. The objects were represented in OpenInventor$^{\text{TM}}$. Affine basis points were defined by the intersections of lines on the wireframe object which were tracked in real time. The virtual objects were defined with respect to those points (see Section 5). The wireframe was rotated clockwise between frames. Note that hidden-surface elimination occurs only between virtual objects; correct occlusion resolution between real and virtual objects requires information about the real objects' 3D structure [31].

transformation that maps points on the virtual object into the affine frame of the feature points being tracked (Section 5).

- **Real-time affine basis tracking:** Determination of the projection matrix for rendering virtual objects requires tracking the affine basis points reliably and efficiently across frames. Our system tracks lines in real time and uses line intersections to define the basis points (Section 6).
- **Object animation:** Because virtual objects are represented in terms of objects physically present in the environment, their motion relative to such objects becomes particularly easy to specify without any metric information about their 3D locations or about the camera (Section 7).

4. Visible Surface Rendering

The projection of points on an affinely-represented virtual object is completely determined by the location of the feature points defining the affine frame. One of the key aspects of affine object representations is that even though they are non-Euclidean, they nevertheless allow rendering operations such as z-buffering and clipping [15] to be performed accurately. This is because both depth order as well as the intersection of lines and planes is preserved under affine transformations.

More specifically, z-buffering relies on the ability to order in depth two object points that project to the same pixel in the image. Typically, this operation is performed by assigning to each object point a z-value which orders the points along the optical axis of the (graphics) camera. The observation we use to render affine objects is that the actual z-value assigned to each point is irrelevant as long as the correct ordering of points is maintained. To achieve such an ordering we represent the camera's optical axis in the affine frame defined by the feature points being tracked, and we order object points along this axis.

The optical axis of the camera can be defined as the 3D line whose points project to a single pixel in the image. This is expressed mathematically by representing the optical axis of the camera with the homogeneous vector $[\zeta^T \ 0]^T$ where ζ is given by the cross product

$$\zeta = \begin{bmatrix} u_{b_1} - u_{p_o} \\ u_{b_2} - u_{p_o} \\ u_{b_3} - u_{p_o} \end{bmatrix} \times \begin{bmatrix} v_{b_1} - v_{p_o} \\ v_{b_2} - v_{p_o} \\ v_{b_3} - v_{p_o} \end{bmatrix} \qquad (6)$$

and $[u_{b_i} \ v_{b_i} \ 1]^T, i = 1, \ldots, 3$ are the image locations of the affine basis points. To order points along the optical axis we assign to each point p on the model a z-value equal to the dot product $p \cdot [\zeta^T \ 0]^T$. Hence, the entire projection process is described by a single 4×4 homogeneous matrix.

Observation 1 (Projection Equation) *Visible surface rendering of a point p on an affine object can be achieved by applying the following transformation to p:*

$$\begin{bmatrix} u \\ v \\ w \\ 1 \end{bmatrix} = \begin{bmatrix} u_{b_1} & u_{b_2} & u_{b_3} & u_{p_o} \\ v_{b_1} & v_{b_2} & v_{b_3} & v_{p_o} \\ & \zeta^T & & z_o \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \qquad (7)$$

where u and v are the image coordinates of p's projection and w is p's assigned z-value.

The matrix in Eq. (7) is an affine generalization of the *view transformation matrix*, which is commonly used in computer graphics for describing arbitrary orthographic and perspective projections of Euclidean objects and for specifying clipping planes. A key practical consequence of the similarity between the Euclidean and affine view transformation matrices is that graphics operations on affine objects can be performed using existing hardware engines for real-time projection, clipping and z-buffering. In our experimental system the matrix of Eq. (7) is input directly to a Silicon Graphics RealityEngine2 for implementing these operations efficiently in OpenGL (Figure 4).

5. Interactive Object Placement

Before virtual objects can be overlaid with images of a three-dimensional environment, the geometrical relationship between these objects and the environment must be established. Our approach for placing virtual objects in the 3D environment borrows from a few simple results in stereo vision [13]: given a point in space, its 3D location is uniquely determined by the point's projection in two images taken at different positions of the camera (Figure 5(a)). Rather than specifying the virtual objects' affine coordinates explicitly, our system allows the user to interactively specify what the

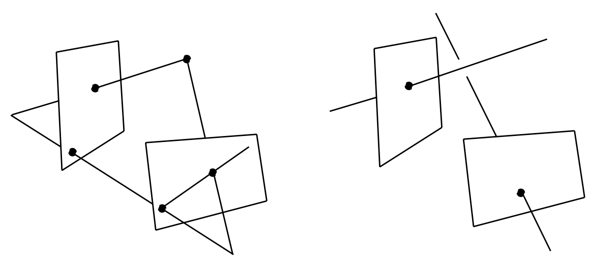


Figure 5. Positioning virtual objects in a 3D environment. (a) Any 3D point is uniquely specified by its projection in two images along distinct lines of sight. The point is the intersection of the two visual rays, ζ^L, ζ^R that emanate from the camera's center of projection and pass through the point's projections. (b) In general, a pair of arbitrary points in two images does not specify two intersecting visual rays. A sufficient condition is to require the point in the second image to lie on the *epipolar line*, i.e., on the projection of the first visual ray in the second image. This line is determined by the projection matrix.

objects should "look like" in two images of the environment. In practice, this involves specifying the projection of points on the virtual object in two images in which the affine basis points are also visible. The main questions here are (1) how many point projections need to be specified in the two images, (2) how does the user specify the projection of these points, and (3) how do these projections determine the objects' affine representation?

The number of point correspondences required to determine the position and shape of a virtual object is equal to the number of points that uniquely determine the object-to-affine transformation. This transformation is uniquely determined by specifying the 3D location of four non-coplanar points on the virtual object that are selected interactively (Eq. (5)).

To fix the location of a selected point p on the virtual object, the point's projection in two images taken at distinct camera positions is specified interactively using a mouse. The process is akin to stereo triangulation: By selecting interactively the projections, q^L, q^R, of p in two images in which the projection of the affine basis points is known, p's affine coordinates can be recovered using the Affine Reconstruction Property.

The two projections of point p cannot be selected in an arbitrary fashion. In general, the correspondence induced by q^L and q^R may not define a physical point in space. Once p's projection is specified in one image, its projection in the second image must lie on a line satisfying the *epipolar constraint* [25]. This line is computed automatically and is used to constrain the user's selection of q^R in the second image. In particular, if Π^L, Π^R are the projection matrices associated with the first and second image, respectively, and ζ^L, ζ^R are the corresponding optical axes defined by Eq. (6), the epipolar line can be parameterized by the set [24]

$$\left\{\Pi^R[(\Pi^L)^{-1}q^L + t\zeta^L] \mid t \in \Re\right\}. \quad (8)$$

By taking the epipolar constraint into account, we can ensure that the points specified by the user provide a physically valid placement of the virtual object.

Once the projections of a point on a virtual object are specified in the two images, the points' affine coordinates can be determined by solving the linear system of equations in Eq. (4). This solves the placement problem for virtual objects. The entire process is shown in Figure 6.

Affine object representations lead naturally to a through-the-lens method for constraining further the user degrees of freedom during the interactive placement of virtual objects. Such constraints have been used in vision-based pointing interfaces [10]. An example of such a constraint is planarity: if r_1, r_2, r_3 are three points on a physical plane in the environment and p is a point on the virtual object, the constraint that p lies on the plane of r_1, r_2, r_3 implies that p's projection is a linear combination of the projections of r_1, r_2, r_3. This constraint completely determines the projection of p in a second image in which r_1, r_2, r_3 are visible. The planarity constraint allows virtual objects to be "snapped" to physical objects and "dragged" over their surface by forcing one or more points on the virtual object to lie on planes in the environment that are selected interactively (Figure 7).

6. Affine Basis Tracking

The ability to track the projection of 3D points undergoing rigid transformations with respect to the camera becomes crucial in any method that relies on image information to represent the position and orientation of the camera [1,4,22,28]. Real-time tracking of image features has been the subject of extensive research in computer vision (e.g., see [7–9]). Below we describe a simple approach that exploits the existence of more than the minimum number of feature points to increase robustness, tracks lines rather than points, and automatically provides an updated projection matrix used for rendering virtual objects.

The approach is based on the following observation: Suppose that the affine coordinates of a collection of N non-coplanar feature points is known. Then, the changes in the projection matrix caused by a change in the camera's position, orientation, or calibration parameters can be modeled by the equation

$$\Delta \mathbf{I} = \Delta\Pi_{3\times4}\,\mathbf{M} \quad (9)$$

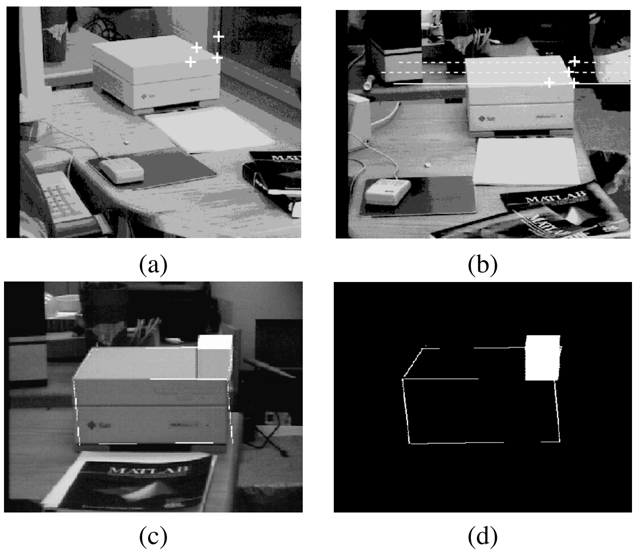


Figure 6. Steps in placing a virtual cube on top of a workstation. (a) The mouse is used to select four points in the image at the position where four of the cube's vertices should project. In this example, the goal is to align the cube's corner with the right corner of the workstation. (b) The camera is moved to a new position and the epipolar line corresponding to each of the points selected in the first image is computed automatically. The epipolar line corresponding to the lower right corner of the cube is drawn solid. Crosses represent the points selected by the user. (c) View of the virtual cube from a new position of the camera, overlaid with live video. (d) Lines being tracked in (c) for defining the affine frame. The affine basis points used for representing the cube are the intersections of the tracked lines. No information about the camera's position or the 3D location of the affine basis points is used in the above steps.

where $\Delta\mathbf{I}$ is the change in the image position of the feature points, $\Delta\Pi$ is the change in the projection matrix, and $\mathbf{M}$ is the matrix holding the affine coordinates of the feature points.[2] Eq. (9) leads directly to a Kalman filter based method for both tracking the feature points and for continuously updating the projection matrix. We use two independent constant acceleration Kalman filters [5] whose states consist of the first and second rows of the projection matrix $\Pi_{3\times4}$, respectively, as well as their time derivatives. The filter's measurement equation is given by Eq. (9).

Feature points and their affine coordinates are determined at system initialization time. Tracking is bootstrapped by interactively specifying groups of coplanar linear features in the initial image of the environment. Feature points are defined to be the intersections of these lines. Once feature points have been tracked over several frames their affine coordinates are computed using the Affine Reconstruction Property. Eq. (9) is used to update the affine basis and the projection matrix. During the tracking phase, the first two rows of the affine view transformation matrix are contained in the state of the Kalman filters. The third row of the matrix is computed from Eq. (6).

7. Rigid Animation By Example

The problem of animating a virtual object so that it appears to undergo a rigid transformation is particularly challenging when the camera's position and calibration parameters are unknown because metric information is lost in the projection process. In general, one cannot distinguish, based solely on an object's affine coordinates, between arbitrary homogeneous 4×4 transformations that cause shear and scaling of a virtual object from those that transform the object rigidly.

Our approach for rigidly animating affine virtual objects is twofold. First we note that pure translations of a virtual

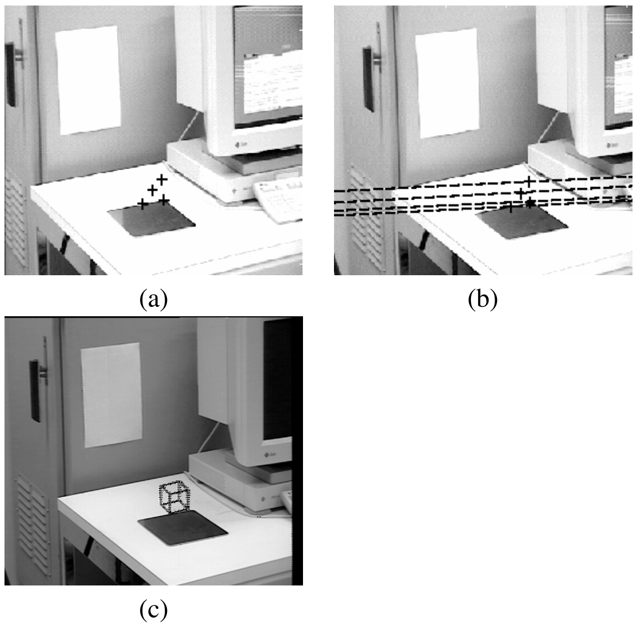


Figure 7. Aligning a virtual quadrangle with a mousepad. Crosses show the points selected in each image. Dotted lines in (b) show the epipolars associated with the points selected in (a). The constraints provided by the epipolars, the planar contact of the cube with the table, as well as the parallelism of the cubes' sides with the side of the workstation allows points on the virtual object to be specified interactively even though no image feature points exist at any four of the object's vertices. (c) Real time overlay of virtual cube.

[2]When only four feature points are available the matrix $\mathbf{M}$ degenerates to a unit 4×4 matrix.

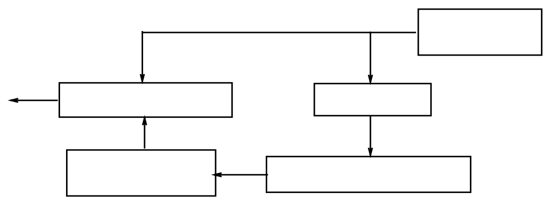


Figure 8. Configuration of our augmented reality system.

object do correspond to rigid translations in the world coordinate frame. Second, to realize arbitrary rigid rotations we exploit the user's ability to interact with the augmented reality system: the user effectively "demonstrates" to the system which transformations to use to generate rigid rotations.

Suppose that all feature points defining the affine basis lie on a single rigid object and that this object is viewed simultaneously by two cameras. To specify a valid rigid rotation, the user simply rotates the object containing the affine basis in front of the two cameras. The image positions of the feature points at the end of the rotation define four points in 3D space whose affine coordinates with respect to the original affine basis can be determined using the Affine Re-Projection Property. Eq. (5) tells us that these coordinates define a 4×4 homogeneous transformation $\mathbf{T}$ that corresponds to a rigid rotation since the object itself was rotated rigidly. By repeating this process three times, the user provides enough information to span the entire space of rotations. See [19] for more details.

8 Experimental System

We have implemented a prototype augmented reality system consisting of a Silicon Graphics RealityEngine2 that handles all graphics operations using the OpenGL and Open-Inventor graphics libraries, and a tracking subsystem implemented in C that runs on a SUN SPARCserver2000. Video input is provided by a Sony camcorder, a TRC BiSight stereo head and a Datacube MaxVideo 10 board used only for frame grabbing (Figure 8). The intrinsic and extrinsic parameters of the cameras were not computed.

Operation of the system involves three steps: (1) initialization of the affine basis, (2) virtual object placement, and (3) affine basis tracking and projection update. Initialization of the affine basis establishes the frame in which all virtual objects will be represented during a run of the system. Basis points are initialized as intersections of line segments that are selected interactively in the initial image. Virtual object initialization follows the sequence of steps shown in Figure 6. Once the affine coordinates of all points on a virtual object are computed, the affine object models are transmitted to the graphics subsystem where they are treated as if they were defined in a Euclidean frame of reference.

Upon initialization of the affine basis, the linear features defining the basis are tracked automatically. Line tracking runs on a single processor at rates between 30Hz and 60Hz for approximately 12 lines and provides updated Kalman filter estimates for the elements of the projection matrix [5]. Conceptually, the tracking subsystem can be thought of as an "affine camera position tracker" that returns the current affine projection matrix asynchronously upon request. For each new video frame, the rows of the projection matrix are used to build the view transformation matrix. This matrix is sent to the graphics subsystem. Figure 9 shows snapshots from example runs of our system. The image overlay was initialized using the placement method of Section 5 at two viewpoints close to the view in Figure 9(a). The objects were then rotated together through the sequence of views in Figure 9(b)-(e) while tracking was maintained on the two black squares.

The accuracy of the image overlays is limited by radial distortions of the camera [4] and the affine approximation to perspective projection. Radial distortions are not currently taken into account. In order to assess the limitations resulting from the affine approximation to perspective we computed mis-registration errors as follows. We used the image projection of vertices on a physical object in the environment (a box) to serve as ground truth and compared these projections at multiple camera positions to those predicted by their affine representation and computed by our system. The image points corresponding to the projection of the affine basis in each image were not tracked automatically but were hand-selected on four of the box's corners to establish a best-case tracking scenario for affine-based image overlay.[3] These points were used to define the affine view transformation matrix. The affine coordinates of the remaining vertices on the box were then computed using the Affine Reconstruction Property, and their projection was computed for roughly 50 positions of the camera. As the camera's distance to the object increased, the camera zoom was also increased in order to keep the object's size constant and the mis-registration errors comparable. Results are shown in Figures 10 and 11. While errors remain within 15 pixels for the range of motions we considered (in a 512×480 image), the results show that, as expected, the affine approximation to perspective leads to errors as the distance to the object decreases. These effects strongly suggest the utility of projectively-invariant representations for representing virtual objects in calibration-free video overlay.

The accuracy of the image overlays generated by our system was measured as follows. A collection of line trackers was used to track the outline of the two black squares on the object of Figure 12. The affine coordinates of an easily-identifiable 3D point that was rigidly attached to the object

[3]As a result, mis-registration errors reported in Figure 10 include the effects of small inaccuracies due to manual corner localization.

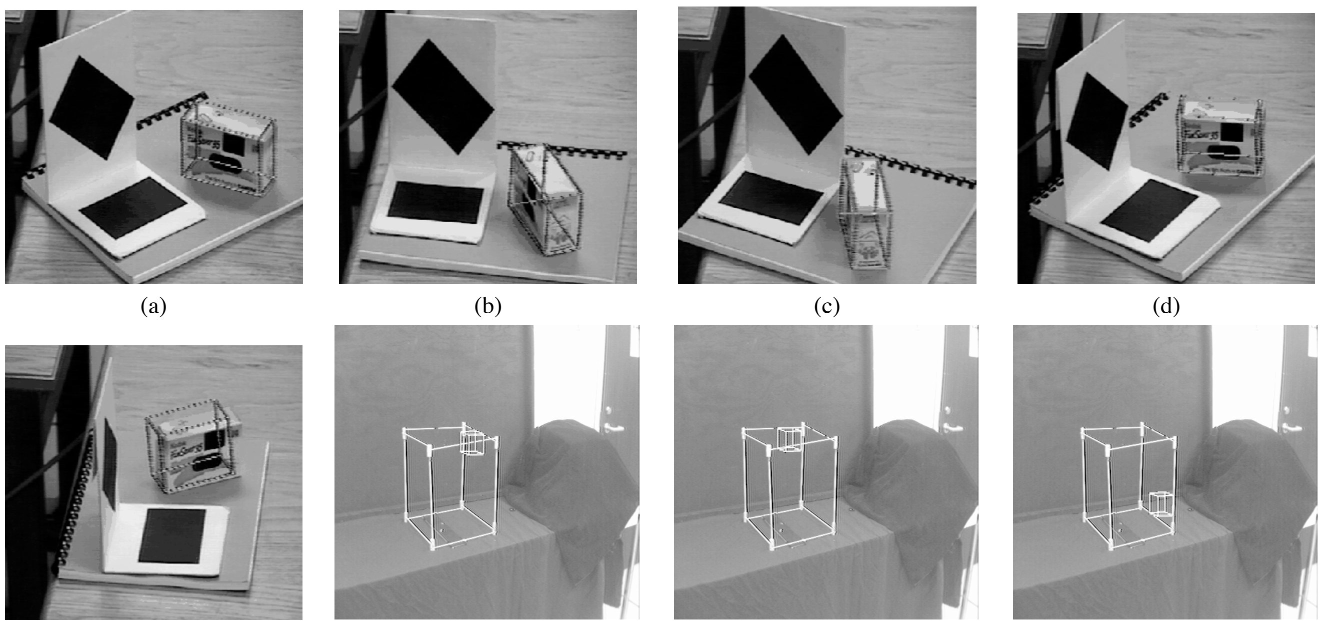


Figure 9. Experimental runs of the system. (a) View from the position where the virtual object was interactively placed over the image of the box. The affine basis points were defined by tracking the outline of the two black squares. The 3D world coordinates of the squares are unknown. (b)-(d) Image overlays after a combined rotation of the box and the object defining the affine basis. (e) Limitations of the approach due to tracking errors. Since the only information used to determine the projection matrix comes from tracking the basis points, tracking errors inevitably lead to wrong overlays. In this example, the extreme foreshortening of the top square led to inaccurate tracking of the affine basis points. (f)-(h) Three snapshots of an animated virtual wireframe cube.

were then computed with respect to the basis defined by the two squares. These coordinates were sent to the graphics subsystem and used to display in real time a small dot at the predicted position of the 3D point as the affine basis points and the 3D point were rotated freely. Two correlation-based trackers were used to track in real-time the image projections of the 3D point and the synthetic dot thus providing an on-line estimate of the ground truth for image overlay. Plots of the $x-$ and $y-$ coordinates of the tracked and re-projected points are shown in Figure 13.

9 Concluding Remarks

The complete reliance on point tracking for generating live video overlays is both the strength and the major limitation of our calibration-free augmented reality approach. On one hand, the approach suggests that real-time tracking of fiducial points contains all the information needed for placement, animation and correct overlay of graphical 3D objects onto live video. Hence, the need for camera position measurements and for information about the sizes and identities of objects in the camera's environment is avoided. On the other hand, the approach is limited by the accuracy, speed, and robustness of point tracking. Significant changes in the camera's position inevitably lead to tracking errors or occlusions of one or more of the tracked fiducial points. In addition, fast rotational motions of the camera make tracking particularly difficult due to large point displacements across frames. Both difficulties can be overcome by using recursive estimation techniques that explicitly take into account feature occlusions and re-appearances [21] and by using fiducials that can be efficiently identified and accurately localized in each frame [22,28].

Limitations of our specific implementation are (1) reliance on the affine approximation to perspective, which inevitably introduces errors in the re-projection process and restricts operation of the system to large object-to-camera distances, (2) lack of a mechanism for correcting the camera's radial distortion, and (3) the existence of a 5-10 frame lag in the re-projection of virtual objects due to communication delays between the tracking and graphics subsystems. On a theoretical level, we are extending the basic approach by representing virtual objects in a projective reference frame, investigating the use of image synthesis techniques for shading non-Euclidean virtual objects, and considering

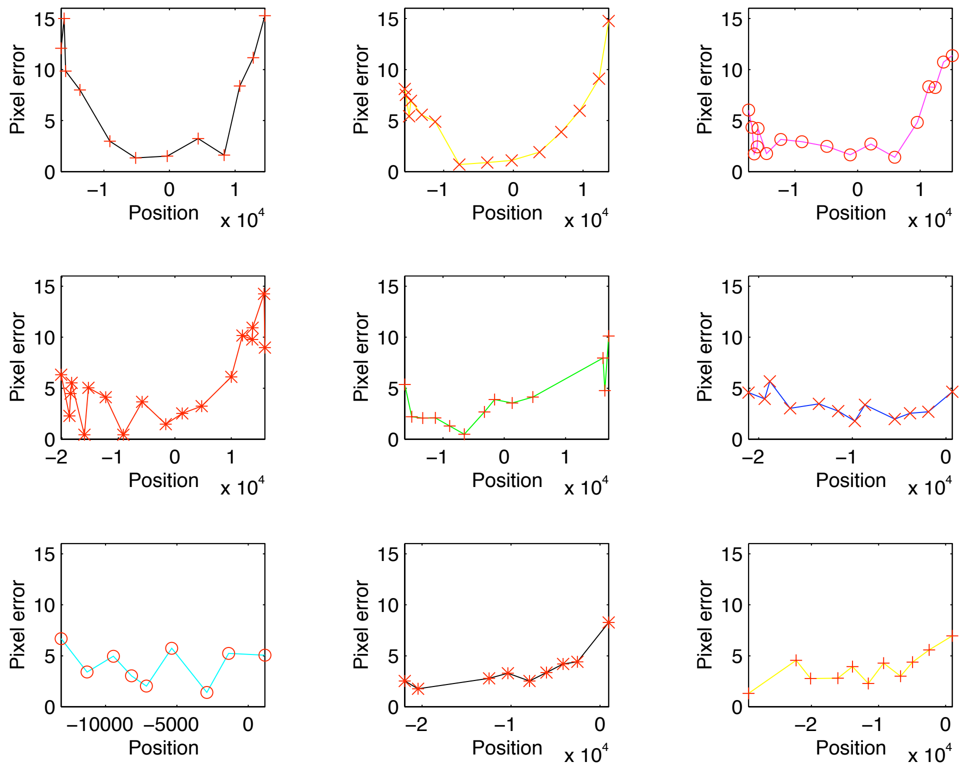


Figure 10. Mis-registration errors. The errors are averaged over three vertices on a $15cm \times 15cm \times 15cm$ box that are not participating in the affine basis. The line style of the plots corresponds to the camera paths shown in Figure 11.

how information from camera position tracking devices can be incorporated in the system when they are available. On a practical level, we are considering techniques for correcting radial distortion and are planning to use specialized fiducial trackers [22] to improve tracking accuracy and versatility.

Acknowledgements

The authors would like to thank Chris Brown for many helpful discussions and for his constant encouragement and support throughout the course of this work. The financial support of the National Science Foundation under Grant No. CDA-9503996, of the University of Maryland under Subcontract No. Z840902 and of Honeywell under Research Contract No. 304931455 is also gratefully acknowledged.

References

[1] A. Azarbayejani, T. Starner, B. Horowitz, and A. Pentland. Visually controlled graphics. *IEEE Trans. Pattern Anal. Machine Intell.*, 15(6):602–605, 1993.

[2] R. Azuma and G. Bishop. Improving static and dynamic registration in an optical see-through HMD. In *Proc. SIGGRAPH'94*, pages 197–204, 1994.

[3] M. Bajura, H. Fuchs, and R. Ohbuchi. Merging virtual objects with the real world: Seeing ultrasound imagery within the patient. In *Proc. SIGGRAPH'92*, pages 203–210, 1992.

[4] M. Bajura and U. Neumann. Dynamic registration correction in video-based augmented reality systems. *IEEE Computer Graphics and Applications*, 15(5):52–60, 1995.

[5] Y. Bar-Shalom and T. E. Fortmann. *Tracking and Data Association*. Academic Press, 1988.

[6] E. B. Barrett, M. H. Brill, N. N. Haag, and P. M. Payton. Invariant linear methods in photogrammetry and model-matching. In *Geometric Invariance in Computer Vision*, pages 277–292. MIT Press, 1992.

[7] A. Blake and M. Isard. 3D position, attitude and shape input using video tracking of hands and lips. In *ACM SIGGRAPH'94*, pages 185–192, 1994.

[8] A. Blake and A. Yuille, editors. *Active Vision*. MIT Press, 1992.

[9] C. M. Brown and D. Terzopoulos, editors. *Real-Time Computer Vision*. Cambridge University Press, 1994.

[10] R. Cipolla, P. A. Hadfield, and N. J. Hollinghurst. Uncalibrated stereo vision with pointing for a man-machine interface. In *Proc. IAPR Workshop on Machine Vision Applications*, 1994.

[11] T. Darrell, P. Maes, B. Blumberg, and A. P. Pentland. A novel environment for situated vision and action. In *IEEE Workshop on Visual Behaviors*, pages 68–72, 1994.

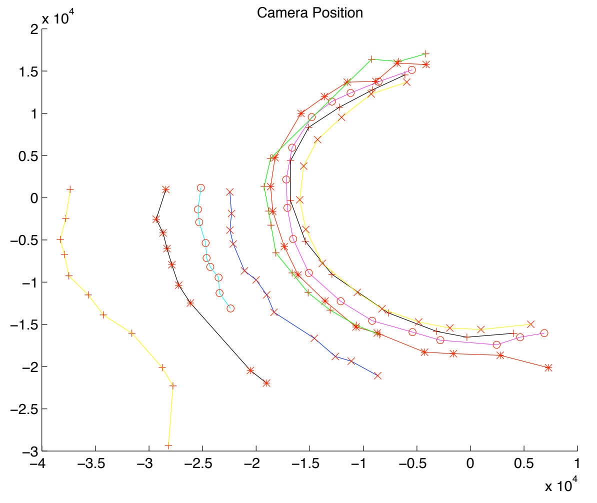


Figure 11. Camera positions used for computing mis-registration errors. The plot shows the first two components of the (unnormalized) vector ζ, corresponding to the computed optical axis of the affine camera. Because the camera was moved on a plane, the plot provides a qualitative indication of the path along which the camera was moved to quantify the mis-registration errors. The path followed a roughly circular course around the box at distances ranging from $0.5m$ to $5m$. The same four non-coplanar vertices of the box defined the affine frame throughout the measurements. The affine coordinates of all visible vertices of the box were computed from two views near position (-1.5,0.5).

[12] O. D. Faugeras. What can be seen in three dimensions with an uncalibrated stereo rig? In *Proc. 2nd European Conf. on Computer Vision*, pages 563–578, 1992.

[13] O. D. Faugeras. *Three-Dimensional Computer Vision: A Geometric Viewpoint*. MIT Press, 1993.

[14] S. Feiner, B. MacIntyre, and D. Soligmann. Knowledge-based augmented reality. *Comm. of the ACM*, 36(7):53–62, 1993.

[15] J. D. Foley, A. van Dam, S. K. Feiner, and J. F. Hughes. *Computer Graphics Principles and Practice*. Addison-Wesley Publishing Co., 1990.

[16] M. Gleicher and A. Witkin. Through-the-lens camera control. In *Proc. SIGGRAPH'92*, pages 331–340, 1992.

[17] W. Grimson et al. An automatic registration method for frameless stereotaxy, image guided surgery, and enhanced reality visualization. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pages 430–436, 1994.

[18] J. J. Koenderink and A. J. van Doorn. Affine structure from motion. *J. Opt. Soc. Am.*, A(2):377–385, 1991.

[19] K. N. Kutulakos and J. Vallino. Affine object representations for calibration-free augmented reality. In *Proc. Image Understanding Workshop*, 1996. To appear.

[20] Y. Lamdan, J. T. Schwartz, and H. J. Wolfson. Object recognition by affine invariant matching. In *Proc. Computer Vision and Pattern Recognition*, pages 335–344, 1988.

[21] P. F. McLauchlan, I. D. Reid, and D. W. Murray. Recursive affine structure and motion from image sequences. In *Proc. 3rd European Conf. on Computer Vision*, pages 217–224, 1994.

[22] J. Mellor. Enhanced reality visualization in a surgical environment. Master's thesis, Massachusetts Institute of Technology, 1995.

[23] J. L. Mundy and A. Zisserman, editors. *Geometric Invariance in Computer Vision*. MIT Press, 1992.

[24] S. M. Seitz and C. R. Dyer. Complete scene structure from four point correspondences. In *Proc. 5th Int. Conf. on Computer Vision*, pages 330–337, 1995.

[25] L. S. Shapiro, A. Zisserman, and M. Brady. 3D motion recovery via affine epipolar geometry. *Int. J. Computer Vision*, 16(2):147–182, 1995.

[26] A. Shashua. A geometric invariant for visual recognition and 3D reconstruction from two perspective/orthographic views. In *Proc. IEEE Workshop on Qualitative Vision*, pages 107–117, 1993.

[27] M. Tuceyran et al. Calibration requirements and procedures for a monitor-based augmented reality system. *IEEE Trans. Visualization and Computer Graphics*, 1(3):255–273, 1995.

[28] M. Uenohara and T. Kanade. Vision-based object registration for real-time image overlay. In *Proc. CVRMED'95*, pages 14–22, 1995.

[29] S. Ullman and R. Basri. Recognition by linear combinations of models. *IEEE Trans. on Pattern Anal. and Mach. Intell.*, 13(10):992–1006, 1991.

[30] D. Weinshall and C. Tomasi. Linear and incremental acquisition of invariant shape models from image sequences. In *Proc. 4th Int. Conf. on Computer Vision*, pages 675–682, 1993.

[31] M. M. Wloka and B. G. Anderson. Resolving occlusion in augmented reality. In *Proc. Symposium on Interactive 3D Graphics*, pages 5–12, 1995.

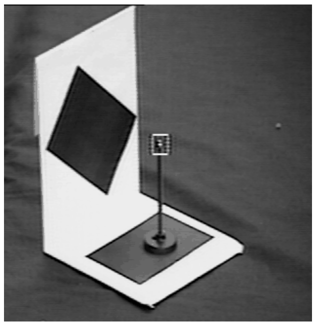
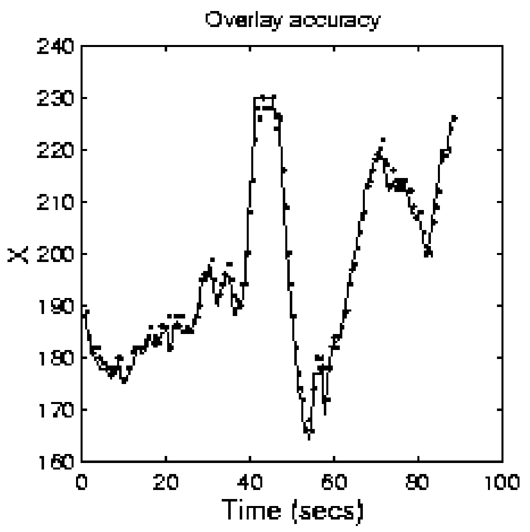


Figure 12. Experimental setup for measuring the accuracy of image overlays. Affine basis points were defined by the corners of the two black squares. The affine coordinates for the tip of a nail rigidly attached to the object were computed and were subsequently used to generate the overlay. Overlay accuracy was measured by independently tracking the nail tip and the generated overlay using correlation-based trackers.



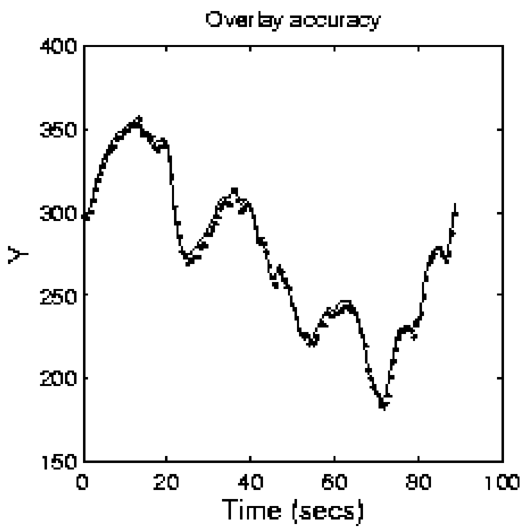


Figure 13. Real-time measurement of overlay errors. Solid lines correspond to the tracked image point and dotted lines to the tracked overlay. The object in Figure 12 was manually lifted from the table and freely rotated for approximately 90 seconds. The mean absolute overlay error in the $x-$ and $y-$ directions was 1.74 and 3.47 pixels, respectively.

[32] J.-R. Wu and M. Ouhyoung. A 3D tracking experiment on latency and its compensation methods in virtual environments. In *Proc. 8th ACM Symp. on User Interface Software and Technology*, pages 41–49, 1995.