Calibration-Free Augmented Reality

Kiriakos N. Kutulakos, *Member, IEEE*, and James R. Vallino, *Student Member, IEEE*

Abstract—Camera calibration and the acquisition of Euclidean 3D measurements have so far been considered necessary requirements for overlaying three-dimensional graphical objects with live video. In this article, we describe a new approach to video-based augmented reality that avoids both requirements: 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 fiducial points that are specified by the user during system initialization and whose world coordinates are unknown.

Our approach is based on the following observation: Given a set of four or more noncoplanar 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 regions and color 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. Experimental results on two augmented reality systems, one monitor-based and one head-mounted, demonstrate that the approach is readily implementable, imposes minimal computational and hardware requirements, and generates real-time and accurate video overlays even when the camera parameters vary dynamically.

Index Terms—Augmented reality, real-time computer vision, calibration, registration, affine representations, feature tracking, 3D interaction techniques.

1 Introduction

There has been considerable interest recently in mixing live video from a camera with computer-generated graphical objects that are registered in a user's three-dimensional environment [1]. Applications of this powerful visualization technique include guiding trainees through complex 3D manipulation and maintenance tasks [2], [3], overlaying clinical 3D data with live video of patients during surgical planning [4], [5], [6], [7], [8], as well as developing three-dimensional user interfaces [9], [10]. The resulting augmented reality systems allow three-dimensional "virtual" objects to be embedded into a user's environment and raise two issues unique to augmented reality:

- *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.

At the heart of these issues lies the ability to register the camera's motion, the user's environment and the embedded virtual objects in the same frame of reference (Fig. 1). Typical approaches use a stationary camera [10] or rely on 3D position tracking devices [11] and precise camera calibration [12] 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. In practice, camera calibration and position tracking are prone to errors which propagate to the augmented display [13]. Furthermore, initialization of virtual objects requires additional calibration stages [4], [14], and the camera must be dynamically recalibrated whenever its position or its intrinsic parameters (e.g., focal length) change.

This article presents a novel approach to augmented reality whose goal is the development of simple and portable video-based augmented reality systems that are easy to initialize, impose minimal hardware requirements, and can be moved out of the highly-controllable confines of an augmented reality laboratory. To this end, we describe the design and implementation of an augmented reality system that generates fast and accurate augmented displays using live video from one or two uncalibrated camcorders as the only input. The key feature of the system is that it allows operations such as virtual object placement and real-time rendering to be performed without relying on any information about the calibration parameters of the camera, the camera's motion, or the 3D locations, dimensions, and identities of the environment's objects. The only requirement is the ability to track across frames at least four fiducial points that are specified by the user during system initialization and whose world coordinates are unknown.

Our work 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 [6], [14], [15], [16], [17]. These approaches rely on tracking the projection of a physical object or a small number of fiducial points in the user's 3D environment to obtain an independent estimate of the camera's position and orientation in space [18], [19], [20]. Even

• *The authors are with the Computer Science Department, University of Rochester, Rochester, NY 14627-0226.*
E-mail: {kyros, vallino}@cs.rochester.edu.

For information on obtaining reprints of this article, please send e-mail to: tvcg@computer.org, and reference IEEECS Log Number 106199.

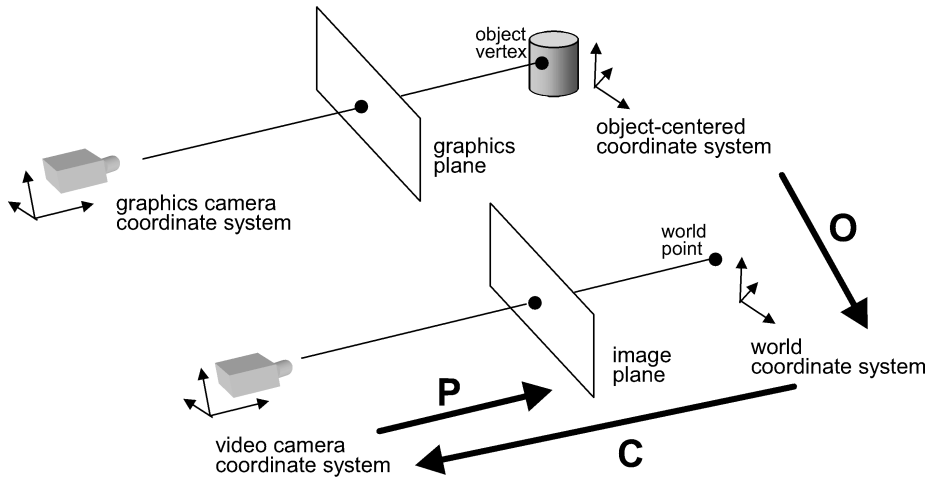


Fig. 1. Coordinate systems for augmented reality. Correct registration of graphics and video requires 1) aligning the internal coordinate systems of the graphics and the video cameras, and 2) specifying the three transformations **O**, **C**, and **P** that relate the coordinate systems of the virtual objects, the environment, the video camera, and the image it produces.

though highly-accurate video overlays have been achieved in this manner by either complementing measurements from a magnetic position tracker [15], [16] or by eliminating such measurements entirely [6], [14], current approaches require that

1) a precise Euclidean 3D model is available for the object or the fiducials being tracked,
2) the camera's calibration parameters are known at system initialization, and
3) the 3D world coordinates of all virtual objects are known in advance.

As a result, camera calibration and the acquisition of 3D measurements have so far been considered necessary requirements for achieving augmented reality displays [12], [13] and have created a need for additional equipment such as laser range finders [14], position tracking devices [11], and mechanical arms [16].

To eliminate these requirements, our approach uses the following observation, pointed out by Koenderink and van Doorn [21] and Ullman and Basri [22]: Given a set of four or more noncoplanar 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 regions and color 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 [21], [23], [24], [25] in which the coordinates of vertices on a virtual object are relative to an affine reference frame defined by the fiducial points (Fig. 2).

Affine object representations have been a topic of active research in computer vision in the context of 3D reconstruction [21], [24], [26] and recognition [27]. 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, as well as visible-surface rendering, can be performed efficiently using simple linear methods that operate at frame rate, do not require camera calibration or Euclidean 3D measurements, and exploit the ability of the augmented reality system to interact with its user [28], [29].

To our knowledge, only two systems have been reported [6], [14] that operate without specialized camera tracking devices and without relying on the assumption that the camera is always fixed or perfectly calibrated. The system of Mellor [14] is capable of overlaying 3D medical data over live video of patients in a surgical environment. The system tracks circular fiducials in a 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

Fig. 2. Example video overlays produced by our system. The virtual wireframe object is represented in an affine reference frame defined by the white planar region on the wall and the mousepad. These regions were tracked at frame rate by an uncalibrated camera. The shape and 3D configuration of the regions was unknown.

initialization time and the exact 3D location of the tracked fiducials is recovered using a laser range finder. The most closely related work to our own is the work of Uenohara and Kanade [6]. Their system allows overlay of planar diagrams onto live video by tracking fiducial 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 coplanar fiducial points. Their study did not consider uncalibrated rendering or interactive placement of 3D virtual objects.

Our approach both generalizes and extends previous approaches in four ways.

1) First, the embedding of affinely-represented virtual objects into live video of a 3D environment is achieved without using any metric information about the objects in the camera's field of view or the camera's calibration parameters.
2) Second, 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* [30]. Furthermore, the elements of this matrix are simply the image x- and y- coordinates of fiducial 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 [31], [32], [33], to track the fiducial points and to compute the matrix.
3) Third, the use of affine models leads to a simple *through-the-lens* method [34] for interactively placing virtual objects within the user's 3D environment.
4) Fourth, efficient execution of computer graphics operations on affine virtual objects and real-time (30Hz) generation of overlays are achieved by implementing affine projection computations directly on dedicated graphics hardware.

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, or about the environment's Euclidean 3D structure. 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 virtual objects. The resulting approach provides a simple and direct way to simultaneously handle lack of environmental 3D models and variability or errors in the camera calibration parameters. This is particularly useful when the live video signal is generated by a camera whose focal length can be changed interactively, when graphical objects are embedded in concurrent live video streams from cameras whose internal parameters are unknown and possibly distinct, or when explicit models for the space being augmented are not readily available (e.g., the desktop scene of Fig. 2).

The rest of the article is organized as follows. Section 2 introduces the geometry of the problem and reviews basic results from the study of affine object representations. Section 3 applies these results to the problem of rendering affinely-represented graphical objects and shows that the entire projection process can be described in terms of an affine view transformation matrix that is derived from image measurements. Section 4 then considers how affinely-represented objects can be "placed" in the camera's environment using a simple through-the-lens interactive technique, and Section 5 shows how to compute the affine view transformation matrix by tracking uniform-intensity regions and color fiducial points in the live video stream. Together, Sections 3, 4, and 5 form the core of our approach and provide a complete framework for merging graphical objects with live video from an uncalibrated camera. The implementation and experimental evaluation of two prototype augmented reality systems that use this framework, one monitor-based and one head-mounted, are presented in Section 6. Section 7 then briefly outlines an application that is particularly suited to our affine augmented reality approach and is aimed at interactively building affine object models from live video images. Limitations of our approach are summarized in Section 8.

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 [30]. In homogeneous coordinates, this projection is described by the equation

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

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 (Fig. 1).

Equation (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, (1) becomes

$$\begin{bmatrix} u \\ v \\ h \end{bmatrix} = \Pi_{3\times4}\begin{bmatrix} x' \\ y' \\ z' \\ w' \end{bmatrix}, \tag{2}$$

where $[x'\ y'\ z'\ w']^T$ are the transformed coordinates of point $[x\ y\ z\ w]^T$ and $\Pi_{3\times4}$ models the combined effects of the change in the object's representation as well as

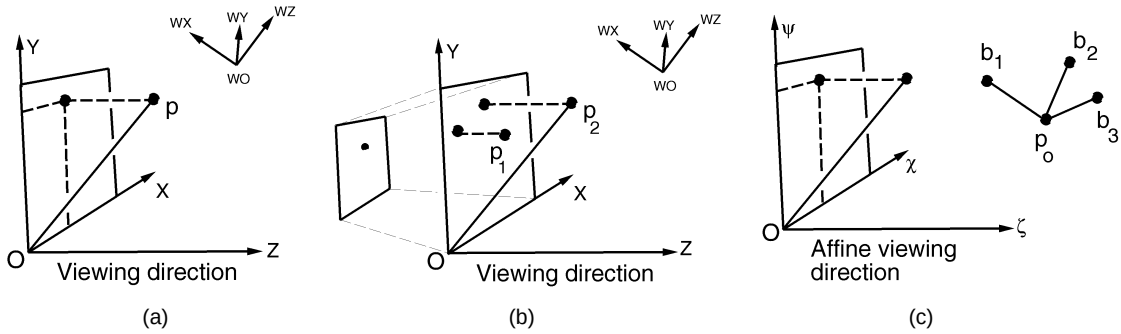


Fig. 3. Projection models and camera reference frames. (a) Orthographic projection. The image projection of a world point p is given by $[p^T X\ p^T Y]^T$, where X and Y are the unit directions of the rows and columns of the camera, respectively, in the world reference frame. The camera's internal reference frame is given by the vectors X and Y as well as the camera's viewing direction, Z, which is orthogonal to the image plane. (b) Weak perspective projection. Points p_1, p_2 are first projected orthographically onto the image plane and then the entire image is scaled by f/z_{avg}, where f is the camera's focal length and z_{avg} is the average distance of the object's points from the image plane. Image scaling is used to model the effect of object-to-camera distance on the object's projection; 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 [36]. (c) The image projection of an affinely-represented point p is given by $[p^T\chi\ p^T\psi]^T$, where χ and ψ are the directions of the rows and columns of the camera, respectively, in the reference frame of the affine basis points. The camera's internal reference frame is defined by the vectors χ and ψ as well as the camera's viewing direction, ζ. These vectors will, in general, *not* form an orthonormal reference frame in (Euclidean) 3D space.

the object-to-world, world-to-camera, and projection transformations.

- It allows the elements of the *projection matrix*, $\Pi_{3\times4}$, to be simply 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 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 be modeled using the *weak perspective projection* model [35] (Figs. 3a and 3b).

2.1 Affine Point Representations

A basic operation in our method for computing the projection of a virtual object is that of *reprojection* [37], [38]: 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 reproject 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 [21], [24], [26] (Fig. 4):

PROPERTY 1 (Reprojection 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*:

$$\begin{bmatrix} u_p^m \\ v_p^m \end{bmatrix} = \underbrace{\begin{bmatrix} u_{\mathbf{b}_1}^m - u_{p_o}^m & u_{\mathbf{b}_2}^m - u_{p_o}^m & u_{\mathbf{b}_3}^m - u_{p_o}^m \\ v_{\mathbf{b}_1}^m - v_{p_o}^m & v_{\mathbf{b}_2}^m - v_{p_o}^m & v_{\mathbf{b}_3}^m - v_{p_o}^m \end{bmatrix}}_{\Pi_{2\times3}} \begin{bmatrix} x \\ y \\ z \end{bmatrix} + \begin{bmatrix} u_{p_o}^m \\ v_{p_o}^m \end{bmatrix} \quad (3)$$

or, equivalently,

$$\begin{bmatrix} u_p^m \\ v_p^m \\ 1 \end{bmatrix} = \underbrace{\begin{bmatrix} u_{\mathbf{b}_1}^m - u_{p_o}^m & u_{\mathbf{b}_2}^m - u_{p_o}^m & u_{\mathbf{b}_3}^m - u_{p_o}^m & u_{p_o}^m \\ v_{\mathbf{b}_1}^m - v_{p_o}^m & v_{\mathbf{b}_2}^m - v_{p_o}^m & v_{\mathbf{b}_3}^m - v_{p_o}^m & v_{p_o}^m \\ 0 & 0 & 0 & 1 \end{bmatrix}}_{\Pi_{3\times4}} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \quad (4)$$

where $\begin{bmatrix} u_p^m & v_p^m & 1 \end{bmatrix}^T$ is the projection of p; $\mathbf{b_1}$, $\mathbf{b_2}$, $\mathbf{b_3}$ are the basis points; $\begin{bmatrix} u_{p_o}^m & v_{p_o}^m & 1 \end{bmatrix}^T$ 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 projection matrices collecting the image coordinates of the affine basis

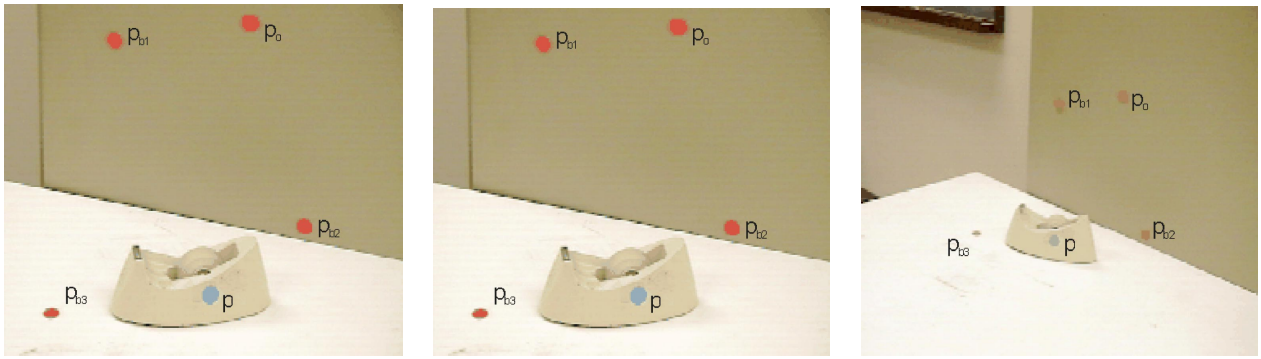


Fig. 4. Properties of affine point representations. The red fiducials $p_o, p_{\mathbf{b_1}}, p_{\mathbf{b_2}}, p_{\mathbf{b_3}}$ define an affine coordinate frame within which all world points can be represented: Point p_o is the origin, and points $p_{\mathbf{b_1}}, p_{\mathbf{b_2}}, p_{\mathbf{b_3}}$ 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.

points in (3) and (4). These equations, which make precise (2), imply 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* (4) *when their projection along two viewing directions is known.*

Intuitively, Property 2 shows that this process can be inverted if at least four noncoplanar 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 overdetermined system of equations

$$\begin{bmatrix} u_p^1 \\ v_p^1 \\ u_p^2 \\ v_p^2 \end{bmatrix} = \begin{bmatrix} u_{\mathbf{b_1}}^1 - u_{p_o}^1 & u_{\mathbf{b_2}}^1 - u_{p_o}^1 & u_{\mathbf{b_3}}^1 - u_{p_o}^1 & u_{p_o}^1 \\ v_{\mathbf{b_1}}^1 - v_{p_o}^1 & v_{\mathbf{b_2}}^1 - v_{p_o}^1 & v_{\mathbf{b_3}}^1 - v_{p_o}^1 & v_{p_o}^1 \\ u_{\mathbf{b_1}}^2 - u_{p_o}^2 & u_{\mathbf{b_2}}^2 - u_{p_o}^2 & u_{\mathbf{b_3}}^2 - u_{p_o}^2 & u_{p_o}^2 \\ v_{\mathbf{b_1}}^2 - v_{p_o}^2 & v_{\mathbf{b_2}}^2 - v_{p_o}^2 & v_{\mathbf{b_3}}^2 - v_{p_o}^2 & v_{p_o}^2 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}. \quad (5)$$

In Section 4, 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.

2.2 The Affine Camera Coordinate Frame

The projection of a set of affinely-represented points can be thought of as a more general transformation that maps these points to an affine 3D coordinate frame attached to the camera. The three vectors defining this frame are derived directly from the projection matrix $\Pi_{2\times3}$ in (3) and extend to the uncalibrated case the familiar notions of an "image plane" and a "viewing direction" (Fig. 3c):

PROPERTY 3 (Affine Image Plane). *Let χ and ψ be the vectors corresponding to the first and second row of $\Pi_{2\times3}$, respectively.*

1) *The vectors χ and ψ are the directions of the rows and columns of the camera, respectively, expressed in the coordinate frame of the affine basis points.*
2) *The affine image plane of the camera is the plane spanned by the vectors χ and ψ.*

The viewing direction of a camera under orthographic or weak perspective projection is defined to be the unique direction in space along which all points project to a single pixel in the image. In the affine case, this direction is expressed mathematically as the null-space of the matrix $\Pi_{2\times3}$:

PROPERTY 4 (Affine Viewing Direction). *When expressed in the coordinate frame of the affine basis points, the viewing direction, ζ, of the camera is given by the cross product*

$$\zeta = \chi \times \psi. \quad (6)$$

Property 4 guarantees that the set of points $\{p + t\zeta, t \in \Re\}$ that defines the line of sight of a point p will project to a single pixel under (4).

Together, the affine row, column, and viewing direction vectors define an affine 3D coordinate frame that describes the orientation of the camera and is completely determined by the projection of the basis points.

3 OBJECT RENDERING

The previous section suggests that once the affine coordinates of points on a virtual object are determined relative to four fiducials 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 noncoplanar 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 **A** such that

$$\left[p_1', p_2', p_3', p_4'\right] = \mathbf{A}\left[p_1\, p_2\, p_3\, p_4\right]. \quad (7)$$

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 [30], to be performed accurately. This is because 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 in decreasing distance from the image plane 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 viewing direction in the affine frame defined by the fiducial points being tracked and we order all object points back-to-front along this direction.

An expression for the camera's viewing direction is provided by (6). This equation, along with Property 3, tells us how to compute a camera's affine viewing direction, ζ, from the projection of the tracked fiducial points. To order points along this direction we assign to each point p on the model a z-value equal to the dot product $[\zeta^T\ 0]^T \cdot p$.

Correct back-to-front ordering of points requires that the affine viewing direction points toward the front of the image plane rather than behind it. Unfortunately, the projection matrix does not provide sufficient information to determine whether or not this condition is satisfied. We use a simple interactive technique to fix the sign of ζ and resolve this "depth reversal" ambiguity: When the first virtual object is overlaid with the live video signal during system initialization, the user is asked to select any two vertices p_1, p_2 on the object for which the vector $p_2 - p_1$ points away from the camera. The sign of ζ is then chosen to ensure that the dot product $\zeta^T \cdot (p_2 - p_1)$ is positive.[1]

The above considerations suggest that once the sign of the affine viewing direction is established, 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*:

$$\begin{bmatrix} u \\ v \\ w \\ 1 \end{bmatrix} = \begin{bmatrix} u_{\mathbf{b_1}} - u_{p_o} & u_{\mathbf{b_2}} - u_{p_o} & u_{\mathbf{b_3}} - u_{p_o} & u_{p_o} \\ v_{\mathbf{b_1}} - v_{p_o} & v_{\mathbf{b_2}} - v_{p_o} & v_{\mathbf{b3}} - v_{p_o} & v_{p_o} \\ & \zeta^{\mathbf{T}} & & z_o \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}, \quad (8)$$

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

The 4×4 matrix in (8) 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 (8) is input directly to a Silicon Graphics RealityEngine2 for implementing these operations efficiently in OpenGL and OpenInventor (Fig. 5).

1. Sign consistency across frames is maintained by requiring that successive ζ-vectors always have positive dot product.

4 INTERACTIVE OBJECT PLACEMENT

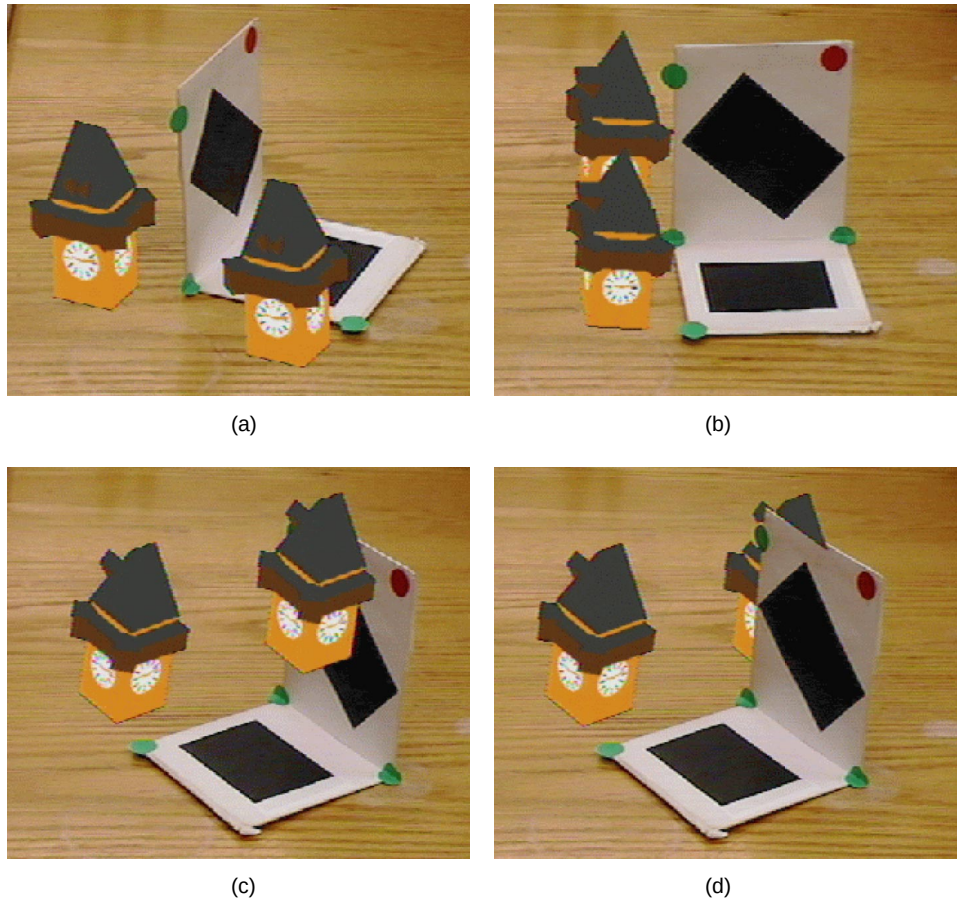
Before virtual objects can be overlaid with images of a three-dimensional environment, the geometrical relationship between these objects and the environment must be established. Our approach for placing virtual objects in the 3D environment borrows from a few simple results in stereo vision [25]: Given 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 (Fig. 6a). Rather than specifying the virtual objects' affine coordinates explicitly, the approach allows a user to interactively specify what the 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 affine transformation is uniquely determined by specifying the 3D location of four noncoplanar points on the virtual object that are selected interactively (7).

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. Once the projections of a point on a virtual object are specified in the two images, the point's affine coordinates can be determined by solving the linear system in (5). This solves the placement problem for virtual objects.

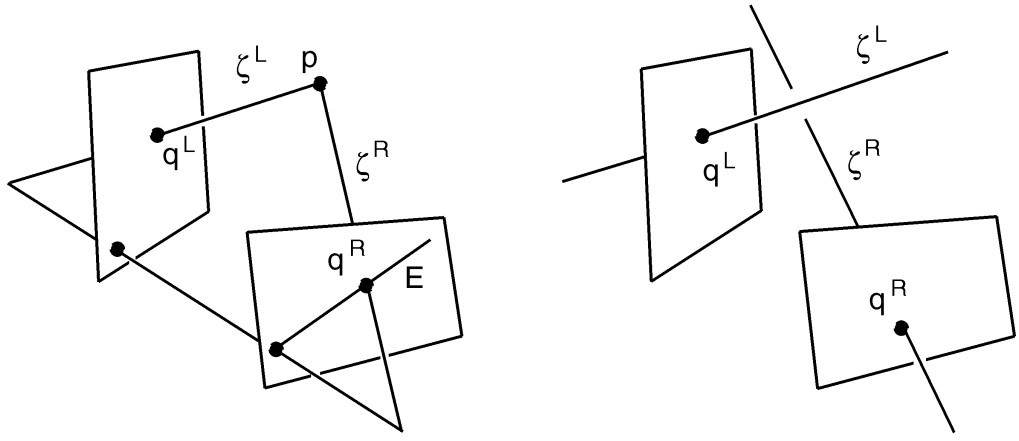
The two projections of point p cannot be selected in an arbitrary fashion. The constraints that govern this selection limit the user degrees of freedom during the interactive placement of virtual objects. Below we consider two constraints that, when combined, guarantee a physically-valid placement of virtual objects.



(a) (b)

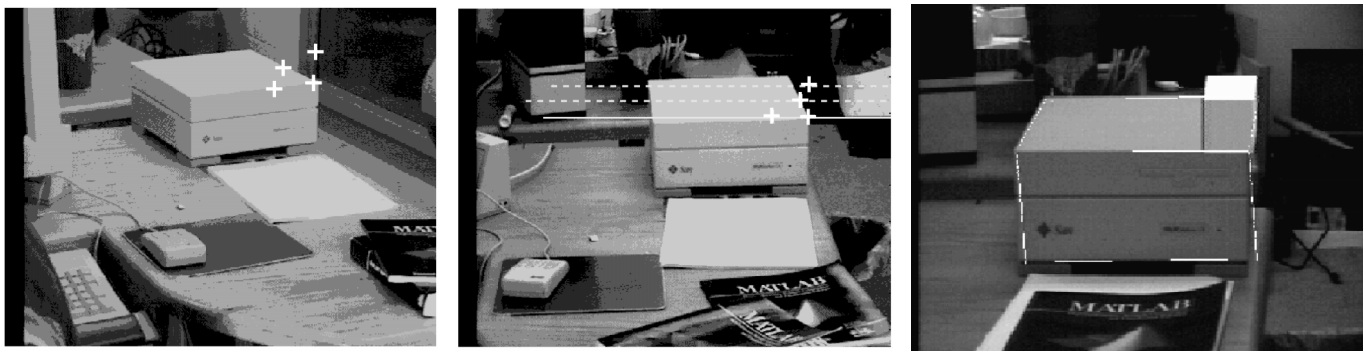
(c) (d)

Fig. 5. Visible-surface rendering of texture-mapped affine virtual objects. The virtual towers were represented in OpenInventor™. Affine basis points were defined by the centers of the four green dots. The virtual towers were defined with respect to those points (see Section 4). (a) Initial augmented view. (b) Augmented view after a clockwise rotation of the object containing the affine basis points. (c) Hidden-surface elimination occurs only between virtual objects; correct occlusion resolution between physical and virtual objects requires information about the geometric relations between them [11]. (d) Real-time visible surface rendering with occlusion resolution between virtual and real objects. Visibility interactions between the virtual towers and the L-shaped object were resolved by first constructing an affine graphical model for the object. By painting the entire model a fixed background color and treating it as an additional virtual object, occlusions between that object and all other virtual objects are resolved via chroma- or intensity-keying. Such affine models of real objects can be constructed using the "3D stenciling" technique of Section 7.



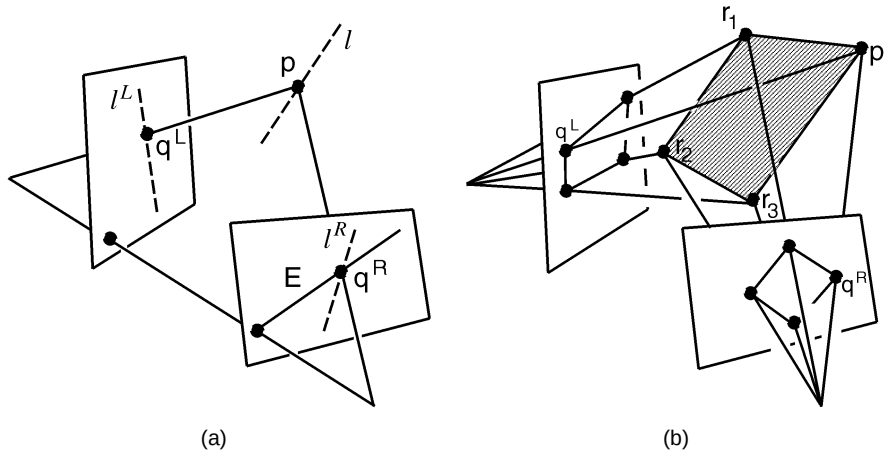
(a) (b)

Fig. 6. Positioning virtual objects in a 3D environment. (a) Any 3D point is uniquely specified by its projection in two images along distinct viewing directions. The point is the intersection of the two visual rays, ζ^L, ζ^R that are parallel to the camera's viewing direction 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 necessary and 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 affine view transformation matrix.



(a) (b) (c)

Fig. 7. Steps in placing a virtual parallelepiped on top of a workstation. (a) The mouse is used to select four points in the image at the position where four of the object's vertices should project. In this example, the goal is to align the object'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 object is drawn solid. Crosses represent the points selected by the user. (c) View of the object from a new position of the camera, overlaid with live video. The affine frame was defined by the workstation's vertices, which were tracked at frame rate. No information about the camera's position or the Euclidean shape of the workstation is used in the above steps.



(a) (b)

Fig. 8. Object placement constraints. (a) Enforcing the Point Collinearity Constraint. The projection q^R of p in a second image is the intersection of l's projection, l^R, and the epipolar line E corresponding to q^L. (b) Enforcing the Point Coplanarity Constraint. Since r_1, r_2, r_3, and p are coplanar, p can be written in the form $\alpha r_1 + \beta r_2 + \gamma r_3$ (i.e., the points r_1, r_2, r_3 constitute a 2D affine basis that can be used to describe point p). The coefficients α, β, γ are computed from q^L and the projections of r_1, r_2, r_3 in the first image. Once computed, these coefficients determine q^R uniquely.

4.1 Epipolar Constraints

In general, the correspondence induced by q^L and q^R may not define a physical point in space (Fig. 6b). Once p's projection is specified in one image, its projection in the second image must lie on a line satisfying the *epipolar constraint* [35]. 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 upper 2×3 blocks of the affine view transformation matrices associated with the first and second image, respectively, and ζ^L, ζ^R are the corresponding viewing directions defined by (6), the epipolar line can be parameterized by the set[2] [39]

$$\{\Pi^R[(\Pi^L)^{-1}q^L + t\zeta^L] \mid t \in \Re\}. \quad (9)$$

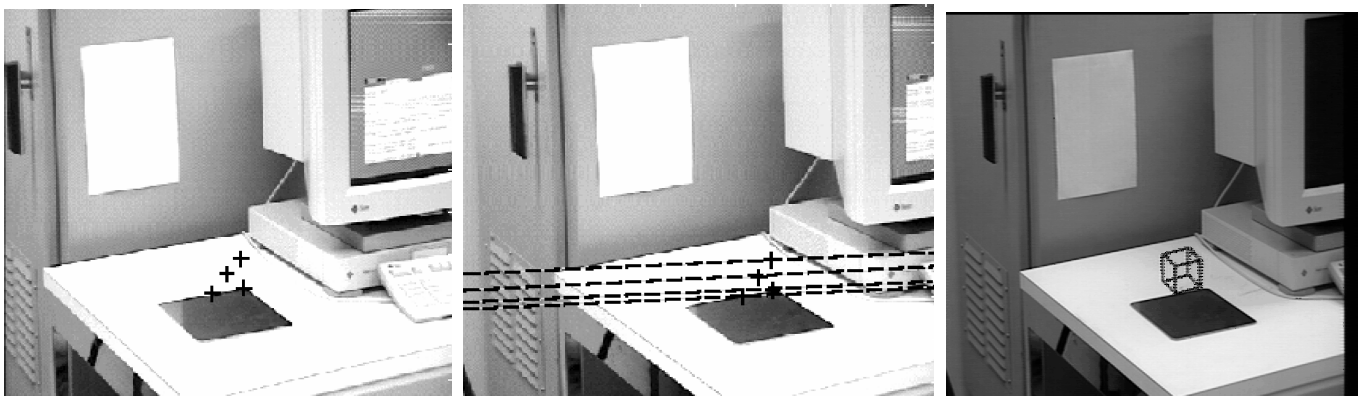
In practice, the position of q^R is specified by interactively dragging a pointer along the epipolar line of q^L. The entire process is shown in Fig. 7.

4.2 Object Snapping Constraints

Affine object representations lead naturally to a through-the-lens method [28], [34], [40] for further constraining the interactive placement of virtual objects. We call the resulting constraints *object snapping constraints* because they allow a user to interactively position virtual objects relative to physical objects in the camera's view volume. Two such constraints are used in our approach:

- **Point Collinearity Constraint:** Suppose p is a virtual object point projecting to q^L and l is a physical line whose projection can be identified in the image. The

2. We use the notation $\left(\Pi^L\right)^{-1}$ to denote the pseudo-inverse of the on-square matrix Π^L. The set $\left\{\left(\Pi^L\right)^{-1}q^L + t\zeta^L \middle| t \in \Re\right\}$ therefore corresponds to the ray of all affinely-represented points projecting to q^L.



(a) (b) (c)

Fig. 9. Aligning a virtual parallelepiped 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 object with the table, as well as the parallelism of the object's sides with the side of the workstation allows points on the virtual object to be specified interactively even though no fiducial points exist at any four of the object's vertices. (c) Real-time overlay of the virtual object.

constraint that p lies on l uniquely determines p's projection, q^R, in any other image in which l's projection can also be identified (Fig. 8a).[3]

- **Point Coplanarity Constraint:** Suppose p is a virtual object point projecting to q^L and r_1, r_2, r_3 are three points on a physical plane in the camera's view volume. The constraint that p lies on the plane of r_1, r_2, r_3 uniquely determines the projection q^R of p in any other image in which r_1, r_2, r_3 can be identified (Fig. 8b).[4]

The collinearity and coplanarity constraints allow virtual objects to be "snapped" to physical objects and interactively "dragged" over their surface by forcing one or more points on a virtual object to lie on lines or planes in the environment that are selected interactively. Similar constraints can be used to enforce parallelism between planes on a virtual object and lines or planes in the user's environment (Fig. 9).

The above considerations lead to the following interactive algorithm for placing virtual objects using a stereo pair of uncalibrated cameras:

4.2.1 Interactive Object Placement Algorithm

Step 1: (User action) Select four or more corresponding fiducial points in the stereo image pair to establish the affine basis.

Step 2: Use (8) and (6) to compute the matrices Π^L, Π^R and the viewing directions, ζ^L, ζ^R associated with the left and right camera, respectively.

Step 3: (User action) Select four noncoplanar vertices p_1, …, p_4 on the 3D model of the virtual object.

Step 4: (User action) Specify the projections of p_1, …, p_4 in the left image.

Step 5: Use (9) to compute the epipolar lines corresponding to the points p_1, …, p_4, given Π^L, Π^R, ζ^L, and ζ^R. Overlay the computed epipolar lines with the right image.

Step 6: (Optional user action) Specify Point Collinearity and Coplanarity Constraints for one or more of the vertices p_1, …, p_4.

Step 7: Specifying the projections of p_1, …, p_4 in the right image:

- **a.** For every p_1, …, p_4 satisfying a collinearity or coplanarity constraint, compute automatically the vertice's position in the right image by enforcing the specified constraint.
- **b.** For every p_1, …, p_4 that does not satisfy any collinearity or coplanarity constraints, allow the user to choose interactively the vertice's projection along its epipolar line.

Step 8: Compute the affine coordinates of p_1, …, p_4 using (5).

Step 9: Use (7) to compute the affine coordinates of all points on the virtual object from the affine coordinates of p_1, …, p_4.

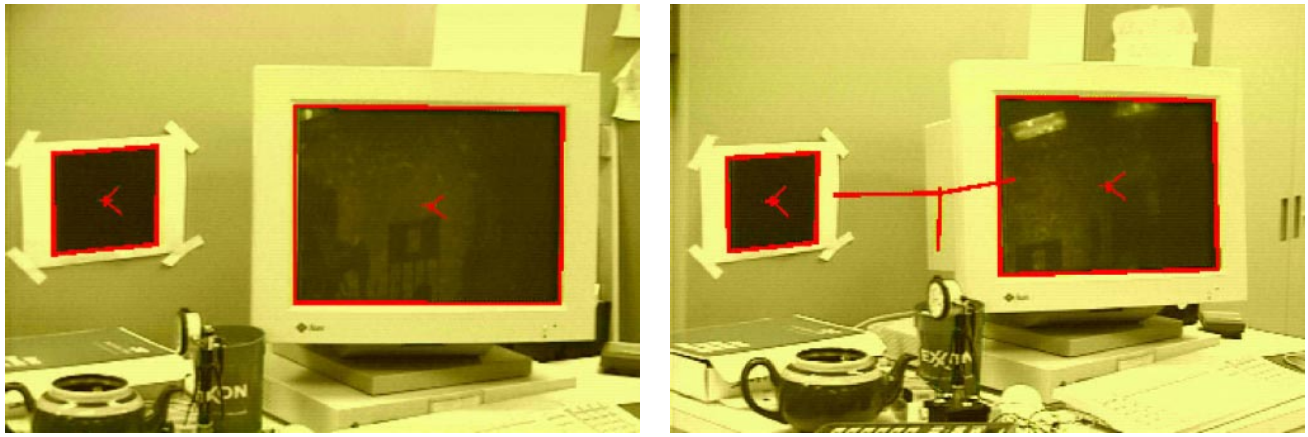
5 Tracking and Projection Update

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 [6], [14], [15], [29]. Real-time tracking of image features has been the subject of extensive research in computer vision (e.g., see [17], [18], [41], [42], [43], [44]). Here, we describe a simple approach that exploits the existence of more than the minimum number of fiducial points to increase robustness and automatically provides an updated affine view transformation matrix for rendering virtual objects.

The approach is based on the following observation: Suppose the affine coordinates of a collection of n noncoplanar fiducial points is known. Then, changes in the view transformation matrix caused by a change in the camera's position, orientation, or calibration parameters can be modeled by the equation

3. The Point Collinearity Constraint is degenerate when the projection of l is parallel to the epipolar lines. This degeneracy can be avoided by simply moving the camera manually to a new position where the degeneracy does not occur. Since no information about the camera's 3D position is required to achieve correct video overlays, this manual repositioning of the camera does not impose additional computational or calibration steps.

4. The simultaneous enforcement of the coplanarity and epipolar constraints leads to an overdetermined system of equations that can be solved using a least squares technique [35].



(a) (b)

Fig. 10. Real-time affine basis tracking. (a) Tracking uniform-intensity regions. In this example, two regions are being simultaneously tracked, the dark region on the wall and the workstation's screen. The centers of each detected region are marked by a red cross. Also shown are the directions of lines connecting these centers to two of the detected region vertices. (b) Updating the view transformation matrix. The affine frame is defined by the eight vertices of the dark regions being tracked. Once the affine coordinates of these vertices are computed using (11), (12), and (13), the view transformation matrix is continuously updated using (10). The projection of the affine basis points corresponding to the current estimate of matrix $\Pi_{2\times4}$ is overlaid with the image and shown in red: The projection of the frame origin, p_o, is given by the last column of $\Pi_{2\times4}$; the projection of the three basis points, $\mathbf{b_1}$, $\mathbf{b_2}$, $\mathbf{b_3}$ is given by the first, second, and third column of $\Pi_{2\times4}$, respectively. The affine coordinates of all region vertices are computed relative to this affine frame. Note that, even though this frame is defined by four physical points in space that remain fixed when the camera moves, the points themselves are only *indirectly* defined—they coincide with the center of mass and principal components of the 3D point set containing the vertices of the regions being tracked.

$$\Delta\mathbf{I} = \Delta\Pi_{2\times4}\,\mathbf{M}, \tag{10}$$

where $\Delta\mathbf{I}$ is the change in the image position of the fiducial points, $\Delta\Pi_{2\times4}$ is the change in the upper two rows of the view transformation matrix, and M is the matrix holding the affine coordinates of the fiducial points.

Equation (10) leads directly to a Kalman filter-based method both for tracking fiducial points (i.e., predicting their image position across frames) and for continuously updating the view transformation matrix. We use two independent constant velocity Kalman filters [31] whose states consist of the first and second row of the matrix $\Pi_{2\times4}$, respectively, as well as their time derivatives. The filters' measurement equations are given by (10). Interpreted as physical systems, these filters estimate the motion of the coordinate frame of the affine camera (Section 2.2). Furthermore, since the matrix $\Pi_{2\times4}$ holds the projections of the four points defining the affine basis, these filters can also be thought of as estimating the image position and velocity of these projections. 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 (6).

Equation (10) reduces the problem of updating the view transformation matrix to the problems of tracking the image trajectories of fiducial points in real time and of assigning affine coordinates to these points. Affine basis tracking relies on real-time algorithms for tracking uniform-intensity planar polygonal regions within the camera's view volume and for tracking color "blobs." Affine coordinate computations are performed during system initialization. We consider each of these steps below.

5.1 Tracking Polygonal Regions

The vertices of uniform-intensity planar polygonal regions constitute a set of fiducial points that can be easily and efficiently tracked in a live video stream. Our region tracking algorithm proceeds in three steps:

1) searching for points on the boundary of the region's projection in the current image,
2) grouping the localized boundary points into linear segments using the polyline curve approximation algorithm [45], and
3) fitting lines to the grouped points using least squares in order to construct a polygonal representation of the region's projection in the current image.

Efficient search for region boundary points is achieved through a radially-expanding, coarse-to-fine search that exploits the region's uniform intensity and starts from a "seed" point that can be positioned anywhere within the region. Once the boundary points are detected and grouped, the region's vertices are localized with subpixel accuracy by intersecting the lines fitted to adjacent point groups.

Region tracking is bootstrapped during system initialization by interactively selecting a seed point within two or more uniform-intensity image regions. These seed points are subsequently repositioned at the predicted center of the regions being tracked (Fig. 10a). Since the only requirement for consistent region tracking is that the seed points lie within the regions' projection in the next frame, region-based tracking leads to accurate and efficient location of fiducial points while also being able to withstand large interframe camera motions.

5.2 Tracking Color Blobs

Polygonal region tracking is limited by the requirement that large polygonal regions must lie in the camera's field of

view. To overcome this restriction, we also employ an alternative algorithm that exploits the availability of a dedicated Datacube MV200 color video processor to track small colored markers. Such markers can be easily placed on objects in the environment, are not restricted to a single color, and can occupy small portions of a camera's visual field.

Tracking is achieved by detecting connected groups of pixels of one or more a priori-specified colors. Each connected group of pixels constitutes a "blob" feature whose location is defined by the pixels' centroid. Blob detection proceeds by

1) digitizing the video stream in a hue-saturation-value (HSV) color space,
2) using a lookup table to map every pixel in the image to a binary value that indicates whether or not the pixel's hue and saturation are close to those of one of the a priori-specified marker colors, and
3) computing the connected components in the resulting binary image [45].

Because these steps are performed by the video processor at a rate of 30Hz, accurate localization of multiple color blobs as small as 8×8 pixels is accomplished in each frame independently. This capability ensures that small color markers are localized and tracked even if their projected position changes significantly between images (e.g., due to a rapid rotation of the camera). It also allows marker tracking to resume after temporary occlusions caused by a hand moving in front of the camera or a marker that temporarily exits the field of view.

5.3 Affine Coordinate Computation

The entries of the affine view transformation matrix can, in principle, be updated by tracking just four non-coplanar fiducials in the video stream. In order to increase resistance to noise due to image localization errors, we use all detected fiducial points to define the affine basis and to update the matrix. We employ a variant of Tomasi and Kanade's factorization method [26], [46] that only allows the matrix $\mathbf{M}$ of affine coordinates of $n \geq 4$ fiducial points in (10) to be recovered from $m \geq 2$ views of the points. The only input to the computation is a *centered measurement matrix* that collects the image coordinates of the n fiducial points at m unknown camera positions, centered by the points' center of mass in each frame. This matrix, which is of rank three under noise-free conditions, is constructed by tracking the selected fiducials while the camera is repositioned manually.

As shown in [26], [46], the rank-three property of the measurement matrix allows us to assign a set of affine coordinates to each fiducial point and a view transformation matrix to each of the m views through a singular-value decomposition of the matrix:

$$\begin{bmatrix} u^1_{p_1} - u^1_c & \dots & u^1_{p_n} - u^1_c \\ v^1_{p_1} - v^1_c & \dots & v^1_{p_n} - v^1_c \\ & \vdots & \\ u^m_{p_1} - u^m_c & \dots & u^m_{p_n} - u^m_c \\ v^m_{p_1} - v^m_c & \dots & v^m_{p_n} - v^m_c \end{bmatrix} = \mathbf{U}\Sigma\mathbf{V}^{\mathbf{T}}, \tag{11}$$

where p_i, $i = 1, \dots, n$ are the detected fiducials and $[u_c\ v_c]^T$ is the center of mass of their projection. Specifically, if $\mathbf{U_{m\times 3}}$, $\mathbf{\Sigma_{3\times 3}}$, and $\mathbf{V_{n\times 3}}$ are the upper $m \times 3$, 3×3, and $n \times 3$ blocks of $\mathbf{U}$, Σ, and $\mathbf{V}$, respectively, the first two rows of the view transformation matrix in the jth image are defined by (12) and (13) (Fig. 10b):

$$\begin{bmatrix} u^1_{\mathbf{b_1}} - u^1_c & u^1_{\mathbf{b_2}} - u^1_c & u^1_{\mathbf{b_3}} - u^1_c \\ v^1_{\mathbf{b_1}} - v^1_c & v^1_{\mathbf{b_2}} - v^1_c & v^1_{\mathbf{b_3}} - v^1_c \\ & \vdots & \\ u^m_{\mathbf{b_1}} - u^m_c & u^m_{\mathbf{b_2}} - u^m_c & u^m_{\mathbf{b_3}} - u^m_c \\ v^m_{\mathbf{b_1}} - v^m_c & v^m_{\mathbf{b_2}} - v^m_c & v^m_{\mathbf{b_3}} - v^m_c \end{bmatrix} = \mathbf{U_{m\times 3}}\left(\Sigma_{\mathbf{3\times 3}}\right)^{\frac{1}{2}} \tag{12}$$

$$\Pi^{\mathbf{j}}_{\mathbf{2\times 4}} = \begin{bmatrix} u^j_{\mathbf{b_1}} - u^j_c & u^j_{\mathbf{b_2}} - u^j_c & u^j_{\mathbf{b_3}} - u^j_c & u^j_c \\ v^j_{\mathbf{b_1}} - v^j_c & v^j_{\mathbf{b_2}} - v^j_c & v^j_{\mathbf{b_3}} - v^j_c & v^j_c \end{bmatrix}. \tag{13}$$

Furthermore, the affine coordinates of the fiducials are given by

$$\mathbf{M} = \begin{bmatrix} \left(\Sigma_{3\times 3}\right)^{\frac{1}{2}}\left(\mathbf{V}_{3\times n}\right)^T \\ 1 \end{bmatrix}. \tag{14}$$

Intuitively, this decomposition of the measurement matrix simply corresponds to a multiple-point, multiple-view generalization of the Affine Reconstruction Property and of (5).

6 Implementation and Results

To demonstrate the effectiveness of our approach we implemented two prototype augmented reality systems: a monitor-based system that relies on polygonal region tracking to maintain correct registration of graphics and live video, and a system that is based on a head-mounted display (HMD) and relies on a dedicated video processor and color blob tracking to track the affine basis points. Both systems are briefly described below.

6.1 Monitor-Based System

The configuration of our monitor-based augmented reality system is shown in Fig. 11. The system consists of two subsystems. A graphics subsystem, consisting of a Silicon Graphics RealityEngine2 that handles all graphics operations using the OpenGL and OpenInventor graphics libraries, and a tracking subsystem that runs on a Sun SPARCserver2000. Video input is provided by two consumer-grade Sony TR CCD-3000 camcorders and is digitized by a Datacube MaxVideo 10 board that is used only for frame grabbing. The position and intrinsic camera parameters were not computed. Video output is generated by merging the analog video signal from one of the cameras with the output of the graphics subsystem. This merging operation is performed in hardware using a Celect Translator luminance keyer [47]. Operation of the system involves four steps:

1) alignment of the graphics frame buffer with the digitizer frame buffer,
2) initialization of the affine basis,
3) virtual object placement, and
4) affine basis tracking and projection update.

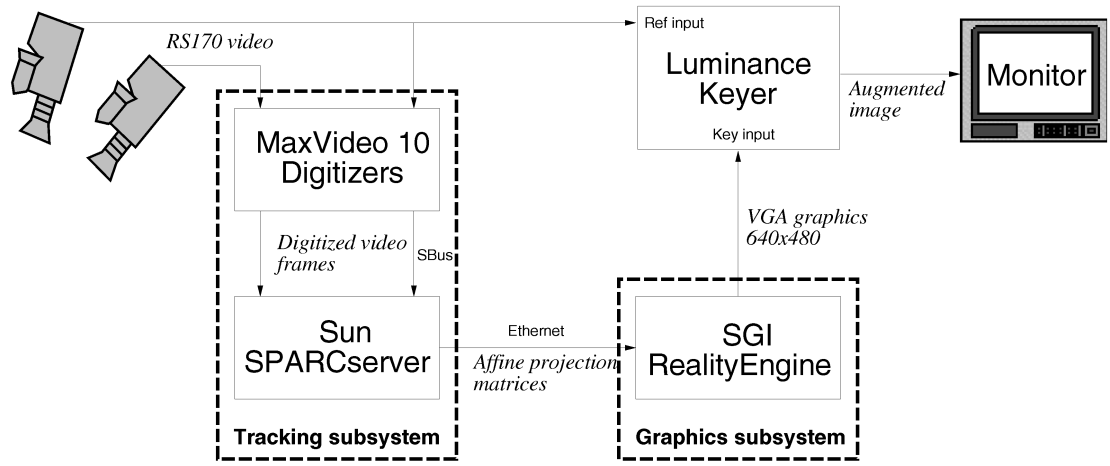
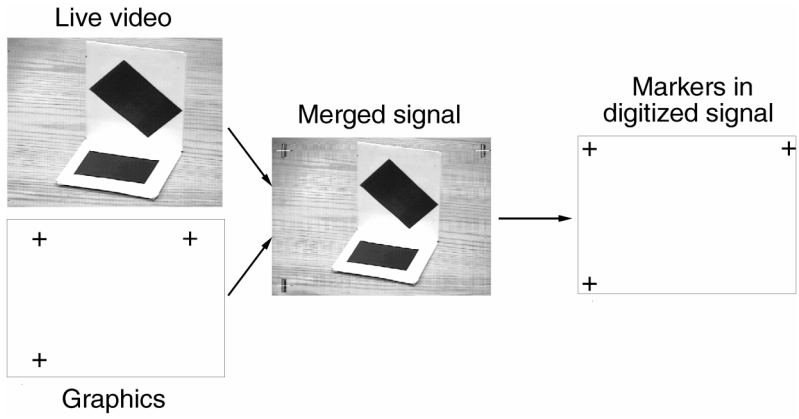
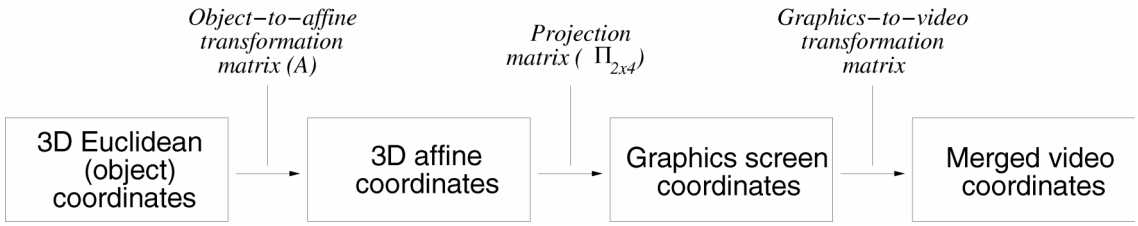


Fig. 11. Configuration of our monitor-based augmented reality system.



(a)



(b)

Fig. 12. Augmented reality display generation. (a) Aligning the graphics and live video signals. During system initialization, a known test pattern consisting of three crosses is generated by the graphics subsystem and merged with the live video signal. The merged video signal is then digitized and the three crosses in the digitized image are located manually. The image coordinates of the localized crosses together with their graphics frame buffer coordinates allow us to compute the 2D transformation mapping graphics frame buffer coordinates to pixel coordinates in the merged and digitized video image. (b) Coordinate systems involved in rendering affine virtual objects.

Alignment of the graphics and digitizer frame buffers ensures that pixels with the same coordinates in the two buffers map to the same position in the video signal. This step is necessary for ensuring that the graphics output signal and the live video signal are correctly aligned before video merging takes place. The step amounts to computing the 2D affine transformation that maps pixels in one frame buffer to pixels in the other. This *graphics-to-video transformation* is described by a 2×3 matrix and can be computed if correspondences between three points in the two buffers are available. The procedure is a generalization of the Image Calibration Procedure detailed in [12] and is outlined in Fig. 12a. The recovered transformation is subsequently applied to all images generated by the graphics subsystem (Fig. 12b).

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 vertices of uniform-intensity regions that are selected interactively in the initial view of the environment. Virtual object initialization follows the steps of the Interactive Object Placement Algorithm, as illustrated in Fig. 7. Once the affine coordinates of

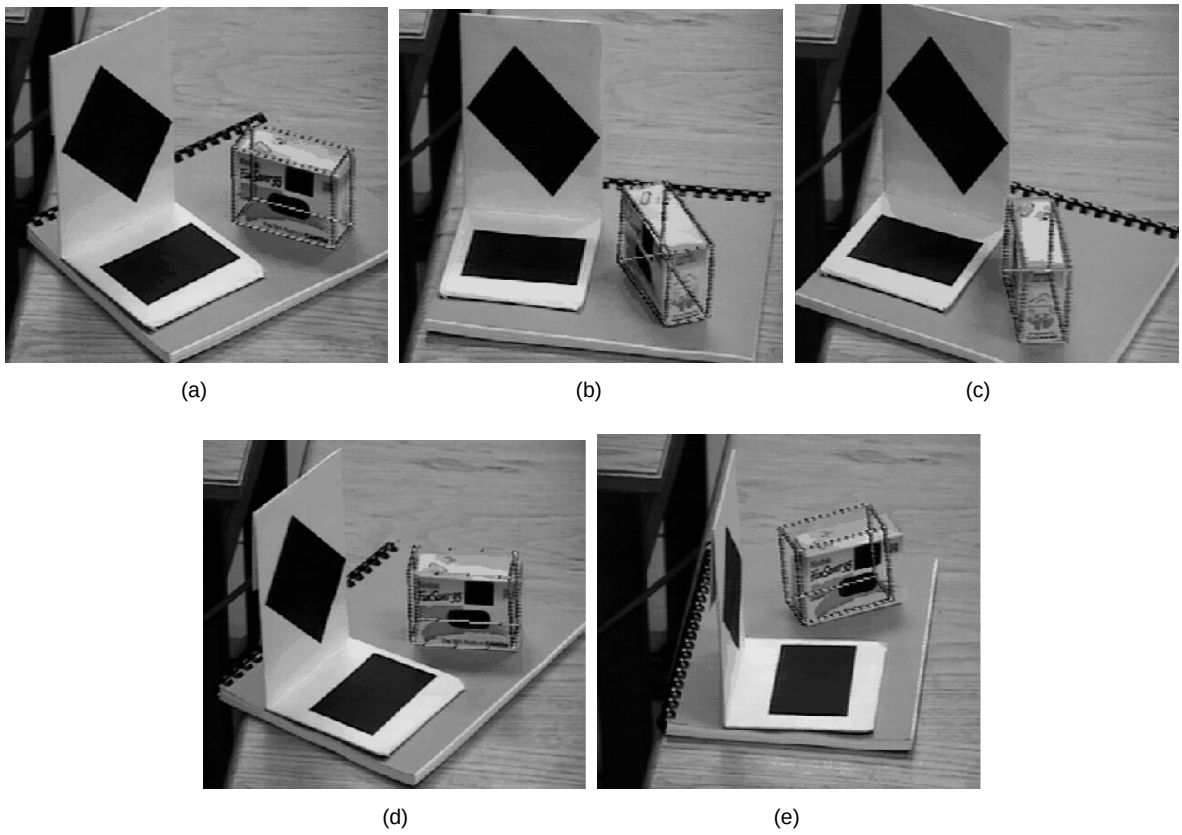


Fig. 13. Experimental runs of the system. (a) View from the position where the virtual object was interactively placed over the image of the box. The affine basis points were defined by tracking the two black polygonal regions. The shape, dimensions, and 3D configuration of the regions was unknown. (b)-(d) 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 affine view transformation matrix comes from tracking the basis points, tracking errors inevitably lead to wrong overlays. In this example, the extreme foreshortening of the top region led to inaccurate tracking of the affine basis points.

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 fiducial points defining the basis are tracked automatically. Region tracking uses the (monochrome) intensity signal of the video stream, runs on a single processor at rates between 30Hz and 60Hz for simultaneous tracking of two regions, and provides updated Kalman filter estimates for the elements of the affine view transformation matrix [31]. Conceptually, the tracking subsystem can be thought of as an "affine camera position tracker" that returns the current affine view transformation matrix asynchronously upon request. This matrix is sent to the graphics subsystem. System delays consist of a 30msec delay due to region tracking and an average 90msec delay due to Ethernet-based communication between the two subsystems. Fig. 13 shows snapshots from example runs of our system. The image overlay was initialized by applying the Interactive Object Placement Algorithm to two viewpoints close to the view in Fig. 13a. The objects were then rotated together through the sequence of views in Figs. 13b, 13c, 13d, and 13e while tracking was maintained on the two black regions. More examples are shown in Fig. 14.

The accuracy of the image overlays is limited by radial distortions of the camera [15], [48] 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 misregistration errors as follows: We used the image projection of vertices on a physical object in the environment to serve as ground truth (the box of Fig. 2) and compared these projections at multiple camera positions to those computed by our system and predicted by the affine representation. 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 corners to establish a best-case tracking scenario for affine-based image overlay.[5] 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 misregistration errors comparable. Results are shown in Figs. 15 and 16. While errors remain within 15 pixels for the range of motions we considered (in a 640×480 image),

5. As a result, misregistration errors reported in Fig. 16 include the effects of small inaccuracies due to manual corner localization.

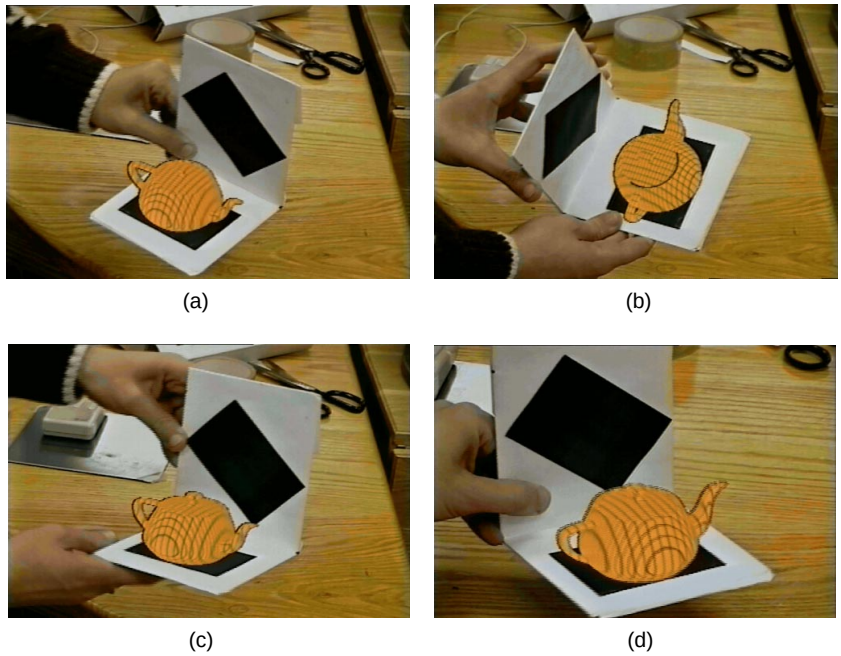


Fig. 14. Overlaying a virtual teapot with live video. The virtual teapot is represented in an affine reference frame defined by the corners of the two black polygonal regions. (a)-(c) Snapshots of the merged live video signal while the object defining the affine frame is being rotated manually. The update rate of the augmented display is approximately 30 Hz. (d) Since no information about camera calibration is used by the system, the camera's position and zoom setting can be changed interactively during a live session.

the results show that, as expected, the affine approximation to perspective leads to errors as the distance to the object decreases [36], [49]. These effects suggest the utility of projectively-invariant representations for representing virtual objects when the object-camera distance is small.

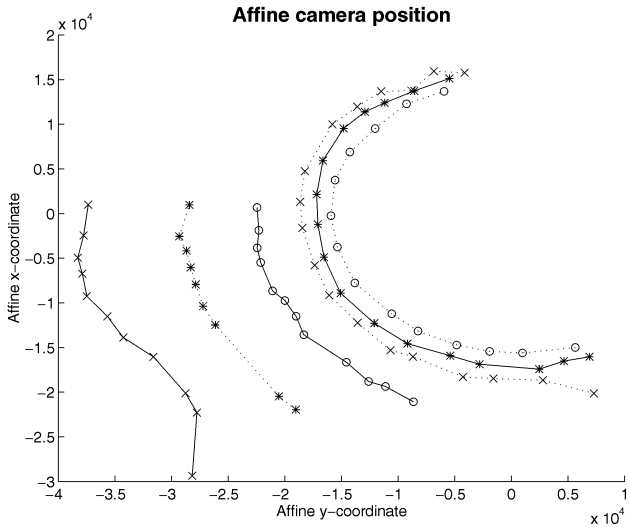


Fig. 15. Camera positions used for computing misregistration errors. The camera was moved manually on a horizontal plane. Because the camera's position was computed in an affine reference frame, the plot and its units of measurement correspond to an affine distortion of the Euclidean plane on which the camera was moved. The camera's actual path followed a roughly circular course around the box at distances ranging up to approximately 5*m*. The same four noncoplanar 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 $(-1.5 \times 10^4, 0.5 \times 10^4)$.

The accuracy of the real-time video overlays generated by our system was measured as follows. A pair of region trackers was used to track the outline of the two black regions on the frame shown in Fig. 17. The affine coordinates of a white dot on the tip of a nail attached to this frame were then computed. These coordinates were sent to the graphics subsystem and used to display in real time a small "virtual dot" at the predicted position of the real dot. Two correlation-based trackers were used to track the position of the real dot in the live video signal as well as the position of the virtual dot in the video signal generated by the graphics subsystem. These trackers operated independently and provided an on-line estimate of the ground truth (i.e., the position of the real dot) as well as the position of the overlay image. Fig. 18 shows results from one run of the error measurement process in which the frame was manually lifted and rotated in an arbitrary fashion for approximately 90 seconds. The mean absolute overlay error in the vertical and horizontal image directions was 1.74 and 3.47 pixels, respectively.

6.2 HMD-Based System

The configuration of our HMD-based system is shown in Fig. 19. Stereo views of the environment are provided by two miniature Panasonic color CCD cameras. The cameras are mounted on a Virtual Research VR4 head-mounted display and are equipped with 7.5mm lenses. Since neither the 3D position nor the intrinsic camera parameters are required in our approach, the HMD-based system is just as easy to set up and initialize as the monitor-based system; the only additional initialization step is a manual adjustment of the cameras' position for each user to aid fusion of the left and right video streams.

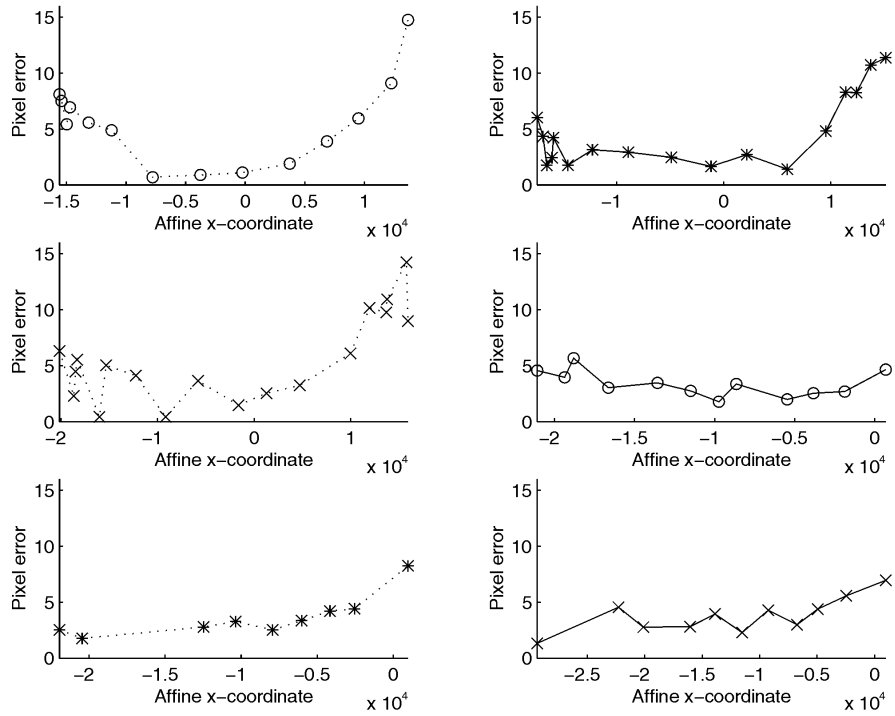


Fig. 16. Misregistration errors. The errors are averaged over three vertices on the box shown in Fig. 2 that are not participating in the affine basis. The line style of the plots corresponds to the camera paths shown in Fig. 15.

The ability to correctly update the affine view transformation matrix even under rapid and frequent head rotations becomes critical for any augmented reality system that employs a head-mounted display [16]. Camera rotations induced by such head motions can cause a significant shift in the projection of individual affine basis points and can cause one or more of these points to leave the field of view. In order to overcome these difficulties, our system employs the color blob tracking algorithm of Section 5.2, which runs at a rate of 30Hz. Because the algorithm does not impose restrictions on the magnitude of interframe motion of image features, the projection of virtual objects can be updated even in the presence of large interframe motions as long as the tracked fiducials remain visible. In the event of a temporary fiducial occlusion, tracking and projection update resume when the occluded fiducials become visible again. With the exception of affine basis tracking, all other computational components of the HMD-based system are identical to our monitor-based system. Overall performance, both in terms of overlay accuracy and in terms of lag, are also comparable to the monitor-based system.

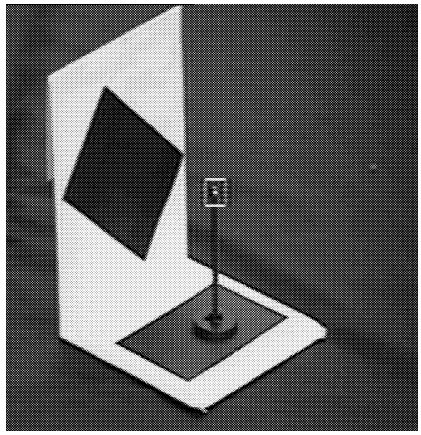


Fig. 17. Experimental setup for measuring the accuracy of image overlays. Affine basis points were defined by the corners of the two black regions. The affine coordinates for the tip of a nail rigidly attached to the object were computed and were subsequently used to generate the overlay. Overlay accuracy was measured by independently tracking the nail tip and the generated overlay using correlation-based trackers.

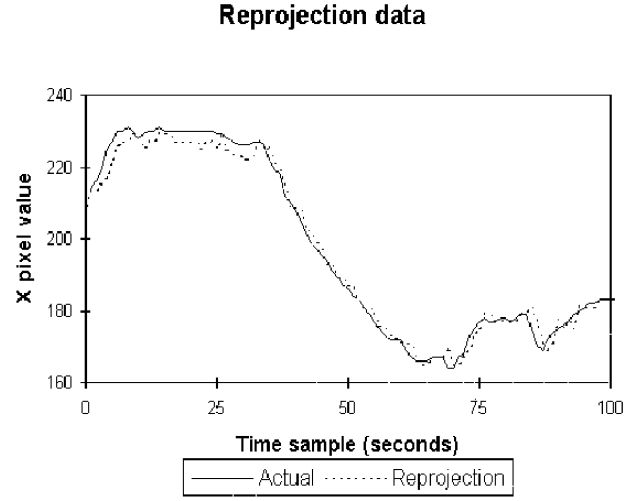


Fig. 18. Real-time measurement of overlay errors. Solid lines correspond to the white dot tracked in the live video signal and dotted lines to the generated overlay. The plot shows that a significant component of the overlay error is due to a lag between the actual position of the dot and the generated overlay [13]. This lag is due to Ethernet-related communication delays between the tracking and graphics subsystems and the fact that no effort was put into synchronizing the graphics and live video streams.

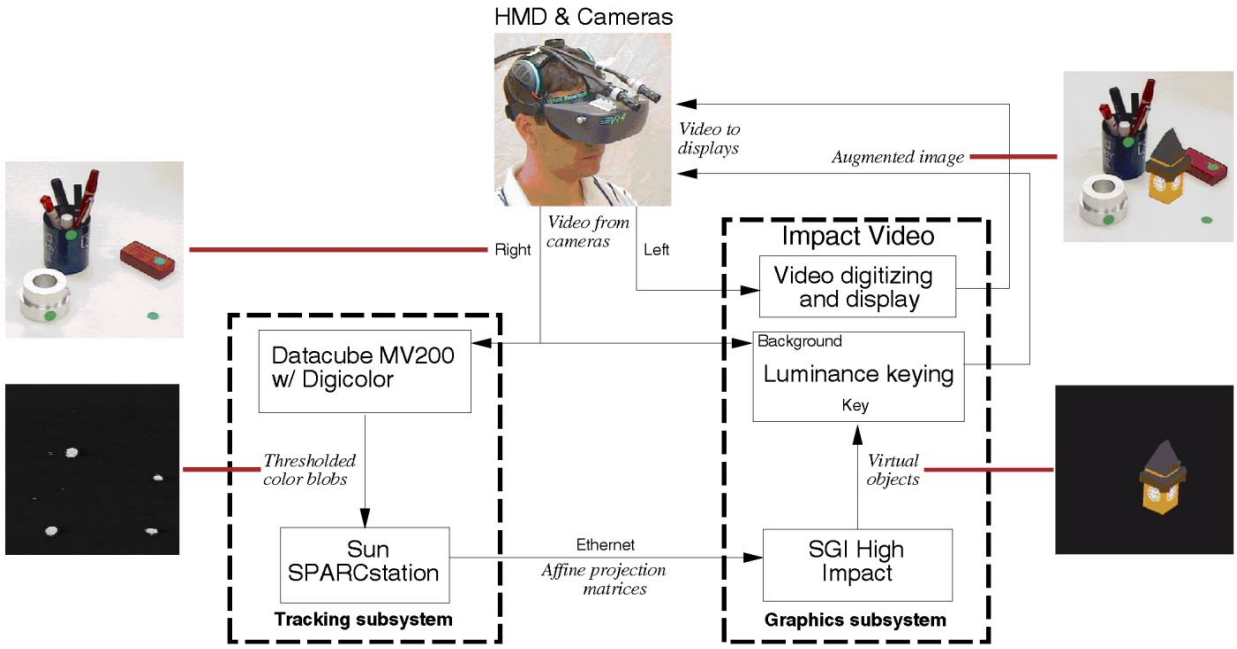


Fig. 19. Configuration of our HMD-based augmented reality system. The affine basis in the example images was defined by the four green circular markers, which were tracked in real time. The markers were manually attached to objects in the environment and their 3D configuration was unknown.

The HMD-based system is presently capable of merging graphics with only one live video stream due to hardware limitations. As a result, video overlays can be viewed by either the left or the right eye, but not by both. Interestingly, we have observed during informal tests in our laboratory that when the video overlays were projected to the users' dominant eye, this limitation was not noticed: Users became aware that they were viewing the augmented environment with just their dominant eye only after being instructed to close that eye. The perceptual implications of these tests are currently under investigation.

7 An Example Application: 3D Stenciling

The ability to merge affinely-represented virtual objects with live video raises the question of how such objects can be constructed in the first place. While considerable progress has been achieved in the fields of image-based shape recovery [25], [46] and active range sensing [50], currently, no general purpose systems exist that can rely on one or more cameras to autonomously construct accurate 3D models of objects that are physically present in a user's environment. This is because the unknown geometry of an object's surface, its unknown texture and reflectance properties, self-occlusions, as well as the existence of shadows and lighting variations, raise a host of research challenges in computer vision that are not yet overcome. Interactive video-based modeling offers a complementary and much simpler approach: By exploiting a user's ability to interact with the modeling system [51], [52], [53], [54] and with the physical object being modeled, we can avoid many of the hard problems in autonomous video-based modeling. Below, we outline a simple interactive approach for building affinely-represented virtual objects that employs the augmented reality system described in the previous sections to interface with the user. We call the approach *3D stenciling* because the physical object being modeled is treated as a three-dimensional analog of a stencil.

In 3D stenciling, an ordinary pen or a hand-held pointer plays the role of a 3D digitizer [30], [55]: The user moves the pointer over the object being modeled while constantly maintaining contact between the pointer's tip and the surface of the object (Fig. 20a). Rather than directly processing the object's live image to recover shape, the stenciling system simply tracks the tip of the pointer with a pair of uncalibrated cameras to recover the tip's 3D affine coordinates at every frame (Fig. 20b). User feedback is provided by overlaying the triangulated tip positions with live video of the object (Figs. 20c, 20d, and 20e).

The key requirement in 3D stenciling is that registration of the partially-reconstructed model with the physical object is maintained at all times during the stenciling operation. The user can, therefore, freely rotate the object in front of the cameras, e.g., to force the visibility of parts of the object that are obstructed from the cameras' initial viewpoint (Fig. 20f, 20g, and 20h). As a result, the augmented reality system guides the user by providing direct visual feedback about the accuracy of the partially-reconstructed model, about the parts of the object that are not currently reconstructed, and about those reconstructed parts that require further refinement. In effect, 3D stenciling allows the user to cover the object being modeled with a form of "virtual 3D paint" [56]; the affine model of a physical object is complete when the object's entire surface is painted.

From a technical point of view, the theoretical underpinnings of the 3D affine reconstruction and overlay generation operations for 3D stenciling are completely described in Sections 2 and 3. In particular, once an affine frame is established and affine view transformation matrices are

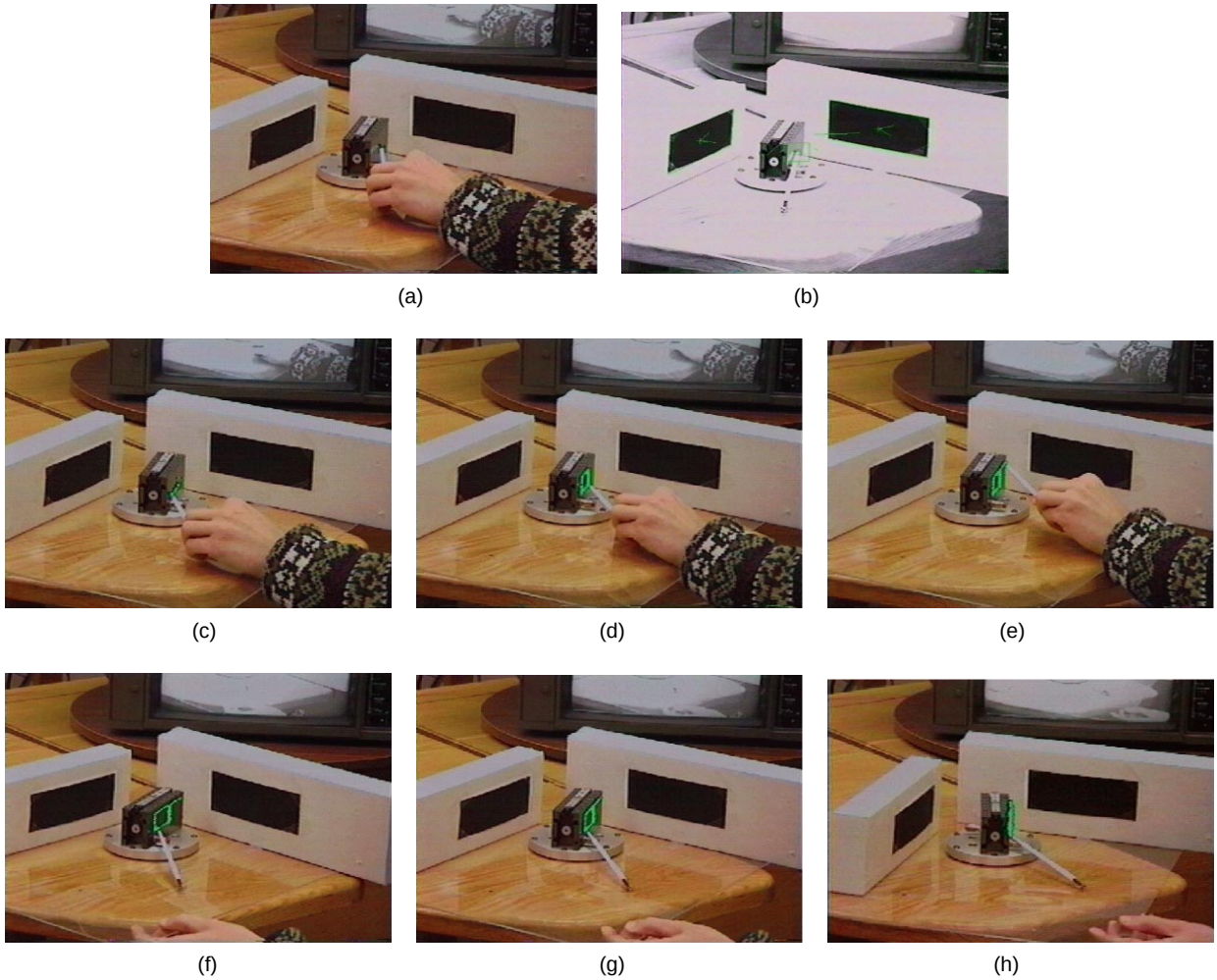


Fig. 20. A 3D stenciling example. Live video is provided by two camcorders whose position and intrinsic parameters were neither known in advance nor estimated. (a) An easily-distinguishable hand-held pointer is moved over the surface of an industrial part. (b) Tracking operations during 3D stenciling. The dark polygonal regions are tracked to establish the affine basis frame. The regions were only employed to simplify tracking and their Euclidean world coordinates were unknown. The green square is centered on the tip of the pointer, also tracked in real time. Tracking takes place simultaneously in two live video streams. (c)-(e) Visualizing the progress of 3D stenciling. The augmented display shows the user drawing a virtual curve on the object's surface in real time. For illustration purposes, the reconstructed tip positions are rendered as small green spheres in the merged live video signal (real-time triangulation is not currently supported). (f)-(h) When the object is manually rotated in front of the two cameras, the reconstructed points appear "locked" on the object's surface, as though the curve traced by the pointer was actually drawn on the object.

assigned to each camera in a stereo pair, the affine coordinates of the pointer's tip can be computed using the Affine Reconstruction Property. Since the reconstructed coordinates are relative to the same affine reference frame that is used to define the view transformation matrices for rendering, overlay generation simply requires transforming the affinely-reconstructed tip positions according to (8).

In order to understand the issues involved in 3D stenciling, we have developed a preliminary real-time 3D stenciling system. System initialization follows the steps outlined in Section 6. The only exception is the Interactive Object Placement step, which is not required to achieve 3D stenciling. In its present form, the system uses normalized correlation [45] to track the tip of a hand-held pointer in two live video streams at frame rate. Once the affine coordinates of the pointer's tip are computed by the tracking subsystem, they are transmitted to the graphics subsystem in order to generate the video overlay. MPEG video sequences demonstrating the system in operation can be found in [57].

We believe that the application of our affine augmented reality approach to tasks such as 3D stenciling offers a great deal of versatility: The user can simply point two uncalibrated camcorders toward a physical 3D object, select a few easily-distinguishable landmarks in the workspace around the object or on the object itself, pick up a pen, and literally start "painting" the object's surface. Key questions that we are currently considering and that are beyond the scope of this article are:

1) What real-time incremental triangulation algorithms are most useful for incrementally modeling the object,
2) How can we use the sequential information available in the pointer's trace to increase modeling accuracy,
3) What surface representations are appropriate for supporting interactive growth, display and refinement of the reconstructed model, and
4) How can we accurately texture map in real time the incrementally-constructed model from the object's live image [16]?

8 Limitations

The use of an affine framework for formulating the video overlay problem is both a strength and a limitation of our calibration-free augmented reality approach. On one hand, the approach suggests that real-time tracking of fiducial points in an unknown 3D configuration contains all the information needed for interactive placement 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 relies on an affine approximation to perspective projection, ignores radial camera distortions, uses purely nonmetric quantities to render virtual objects, and relies on point tracking to generate the live video overlays.

The affine approximation to perspective projection can introduce errors in the reprojection process and restricts system operation to relatively large object-to-camera distances (greater than 10 times the object's size[6] [36]). This restriction can be overcome by formulating the video overlay process within a more general *projective* framework; the analysis presented in this article can be directly generalized to account for the perspective projection model by representing virtual objects in a projective frame of reference defined by five fiducial points [24], [59]. Similarly, radial camera distortions can be corrected by automatically computing an image warp that maps lines in space to lines in the image [60].

The use of non-Euclidean models for representing virtual objects implies that only rendering operations that rely on nonmetric information can be implemented directly. As a result, while projection computations, texture-mapping, and visible surface determination can be performed by relying on affine or projective object representations, rendering techniques that require metric information (e.g., angle measurements for lighting calculations) are not directly supported. In principle, image-based methods for shading affine virtual objects can provide a solution to this problem by linearly combining multiple a priori-stored shaded images of these objects [61], [62], [63].

Complete reliance on the live video stream to extract the information required for merging graphics and video implies that the approach is inherently limited by the accuracy, speed, and robustness of point and region tracking [16]. 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, unless real-time video processing hardware is available, fast rotational motions of the camera will make tracking particularly difficult due to large fiducial point displacements across frames. Both difficulties can be overcome by using recursive estimation techniques that explicitly take into account fiducial occlusions and reappearances [64], by processing images in a coarse-to-fine fashion, and by using fiducials that can be efficiently identified and accurately localized in each frame [6], [14], [65].

6. The approximation is not only valid at such large object-to-camera distances, but has been shown to yield more accurate results in structure-from-motion computations [49], [58].

Limitations of our specific implementation are 1) the existence of a four to five frame lag in the re-projection of virtual objects due to communication delays between the tracking and graphics subsystems, 2) the ability to overlay graphics with only one live video stream in the HMD-based system, and 3) the need for easily-identifiable markers or regions in the scene to aid tracking. We are currently planning to enhance our computational and network resources to reduce communication delays and allow simultaneous merging of two live video streams. We are also investigating the use of efficient and general purpose correlation-based trackers [44], [46] to improve tracking accuracy and versatility.

9 Concluding Remarks

We have demonstrated that fast and accurate merging of graphics and live video can be achieved using a simple approach that requires no metric information about the camera's calibration parameters or about the 3D locations and dimensions of the environment's objects. The augmented reality systems we developed show that the approach leads to algorithms that are readily implementable, are suitable for a real-time implementation, and impose minimal hardware requirements.

Our current efforts are centered on the problem of merging graphics with live video from an "omni-directional" camera [67]. These cameras provide a 360 degree field of view, enable the use of simple and efficient algorithms for handling rotational camera motions, and promise the development of new, image-based techniques that establish a camera's position in a Euclidean, affine, or projective frame without explicitly identifying or tracking features in the live video stream.

Acknowledgments

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 U.S. 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. A preliminary version of this article appeared in the *Proceedings of the 1996 IEEE Virtual Reality Annual International Symposium* (*VRAIS'96*).

References

[1] R.T. Azuma, "A Survey of Augmented Reality," *Presence: Teleoperators and Virtual Environments*, vol. 6, no. 4, pp. 355-385, 1997.

[2] T.P. Caudell and D. Mizell, "Augmented Reality: An Application of Heads-Up Display Technology to Manual Manufacturing Processes," *Proc. Int'l Conf. System Sciences*, vol. 2, pp. 659-669, Hawaii, 1992.

[3] S. Feiner, B. MacIntyre, and D. Soligmann, "Knowledge-Based Augmented Reality," *Comm. ACM*, vol. 36, no. 7, pp. 53-62, 1993.

[4] W. Grimson et al., "An Automatic Registration Method for Frameless Stereotaxy, Image Guided Surgery, and Enhanced Reality Visualization," *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pp. 430-436, 1994.

[5] W.E.L. Grimson et al., "Evaluating and Validating an Automated Registration System for Enhanced Reality Visualization in Surgery," *Proc. CVRMED '95*, pp. 3-12, 1995.
[6] M. Uenohara and T. Kanade, "Vision-Based Object Registration for Real-Time Image Overlay," *Proc. CVRMED '95*, pp. 14-22, 1995.
[7] M. Bajura, H. Fuchs, and R. Ohbuchi, "Merging Virtual Objects With the Real World: Seeing Ultrasound Imagery Within the Patient," *Proc. SIGGRAPH '92*, pp. 203-210, 1992.
[8] A. State, M.A. Livingston, W.F. Garrett, G. Hirota, M.C. Whitton, E.D. Pisano, and H. Fuchs, "Technologies for Augmented Reality Systems: Realizing Ultrasound-Guided Needle Biopsies," *Proc. SIGGRAPH '96*, pp. 439-446, 1996.
[9] P. Wellner, "Interacting With Paper on the DigitalDesk," *Comm. ACM*, vol. 36, no. 7, pp. 86-95, 1993.
[10] T. Darrell, P. Maes, B. Blumberg, and A.P. Pentland, "A Novel Environment for Situated Vision and Action," *IEEE Workshop Visual Behaviors*, pp. 68-72, 1994.
[11] M.M. Wloka and B.G. Anderson, "Resolving Occlusion in Augmented Reality," *Proc. Symp. Interactive 3D Graphics*, pp. 5-12, 1995.
[12] M. Tuceyran, D.S. Greer, R.T. Whitaker, D.E. Breen, C. Crampton, E. Rose, and K.H. Ahlers, "Calibration Requirements and Procedures for a Monitor-Based Augmented Reality System," *IEEE Trans. Visualization and Computer Graphics*, vol. 1, no. 3, pp. 255-273, 1995.
[13] R.L. Holloway, "Registration Errors in Augmented Reality Systems," PhD thesis, Univ. of North Carolina, Chapel Hill, 1995.
[14] J. Mellor, "Enhanced Reality Visualization in a Surgical Environment," Master's thesis, Massachusetts Inst. of Technology, 1995.
[15] M. Bajura and U. Neumann, "Dynamic Registration Correction in Video-Based Augmented Reality Systems," *IEEE Computer Graphics and Applications*, vol. 15, no. 5, pp. 52-60, 1995.
[16] A. State, G. Hirota, D.T. Chen, W.F. Garrett, and M.A. Livingston, "Superior Augmented Reality Registration by Integrating Landmark Tracking and Magnetic Tracking," *Proc. SIGGRAPH '96*, pp. 429-438, 1996.
[17] S. Ravela, B. Draper, J. Lim, and R. Weiss, "Adaptive Tracking and Model Registration Across Distinct Aspects," *Proc. 1995 IEEE/RSJ Int'l Conf. Intelligent Robotics and Systems*, pp. 174-180, 1995.
[18] D.G. Lowe, "Robust Model-Based Tracking Through the Integration of Search and Estimation," *Int'l J. Computer Vision*, vol. 8, no. 2, pp. 113-122, 1992.
[19] D.G. Lowe, "Fitting Parameterized Three-Dimensional Models to Images," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 13, no. 5, pp. 441-449, May 1991.
[20] G. Verghese, K.L. Gale, and C.R. Dyer, "Real-Time Motion Tracking of Three-Dimensional Objects," *Proc. IEEE Conf. Robotics and Automation*, pp. 1,998-2,003, 1990.
[21] J.J. Koenderink and A.J. van Doorn, "Affine Structure From Motion," *J. Optical Soc. Am.*, vol. A, no. 2, pp. 377-385, 1991.
[22] S. Ullman and R. Basri, "Recognition by Linear Combinations of Models," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 13, no. 10, pp. 992-1,006, Oct. 1991.
[23] O.D. Faugeras, "Stratification of Three-Dimensional Vision: Projective, Affine, and Metric Representations," *J. Optical Soc. Am.*, vol. A, vol. 12, no. 3, pp. 465-484, 1995.
[24] *Geometric Invariance in Computer Vision*, J.L. Mundy and A. Zisserman, eds. MIT Press, 1992.
[25] O.D. Faugeras, *Three-Dimensional Computer Vision: A Geometric Viewpoint*. MIT Press, 1993.
[26] D. Weinshall and C. Tomasi, "Linear and Incremental Acquisition of Invariant Shape Models From Image Sequences," *Proc. Fourth Int'l Conf. Computer Vision*, pp. 675-682, 1993.
[27] Y. Lamdan, J.T. Schwartz, and H.J. Wolfson, "Object Recognition by Affine Invariant Matching," *Proc. Computer Vision and Pattern Recognition*, pp. 335-344, 1988.
[28] R. Cipolla, P.A. Hadfield, and N.J. Hollinghurst, "Uncalibrated Stereo Vision With Pointing for a Man-Machine Interface," *Proc. IAPR Workshop on Machine Vision Applications*, 1994.
[29] A. Azarbayejani, T. Starner, B. Horowitz, and A. Pentland, "Visually Controlled Graphics," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 15, no. 6, pp. 602-605, June 1993.
[30] J.D. Foley, A. van Dam, S.K. Feiner, and J.F. Hughes, *Computer Graphics Principles and Practice*. Addison-Wesley, 1990.
[31] Y. Bar-Shalom and T.E. Fortmann, *Tracking and Data Association*. Academic Press, 1988.
[32] J.-R. Wu and M. Ouhyoung, "A 3D Tracking Experiment on Latency and Its Compensation Methods in Virtual Environments," *Proc. Eighth ACM Symp. User Interface Software and Technology*, pp. 41-49, 1995.
[33] R. Azuma and G. Bishop, "Improving Static and Dynamic Registration in an Optical See-Through HMD," *Proc. SIGGRAPH '94*, pp. 197-204, 1994.
[34] M. Gleicher and A. Witkin, "Through-the-Lens Camera Control," *Proc. SIGGRAPH '92*, pp. 331-340, 1992.
[35] L.S. Shapiro, A. Zisserman, and M. Brady, "3D Motion Recovery Via Affine Epipolar Geometry," *Int'l J. Computer Vision*, vol. 16, no. 2, pp. 147-182, 1995.
[36] W.B. Thompson and J.L. Mundy, "Three-Dimensional Model Matching From An Unconstrained Viewpoint," *Proc. IEEE Conf. Robotics and Automation*, pp. 208-220, 1987.
[37] A. Shashua, "A Geometric Invariant for Visual Recognition and 3D Reconstruction From Two Perspective/Orthographic Views," *Proc. IEEE Workshop Qualitative Vision*, pp. 107-117, 1993.
[38] E.B. Barrett, M.H. Brill, N.N. Haag, and P.M. Payton, "Invariant Linear Methods in Photogrammetry and Model-Matching," *Geometric Invariance in Computer Vision*, pp. 277-292. MIT Press, 1992.
[39] S.M. Seitz and C.R. Dyer, "Complete Scene Structure From Four Point Correspondences," *Proc. Fifth Int'l Conf. Computer Vision*, pp. 330-337, 1995.
[40] G.D. Hager, "Calibration-Free Visual Control Using Projective Invariance," *Proc. Fifth Int'l Conf. Computer Vision*, pp. 1,009-1,015, 1995.
[41] *Active Vision*, A. Blake and A. Yuille, eds. MIT Press, 1992.
[42] *Real-Time Computer Vision*, C.M. Brown and D. Terzopoulos, eds. Cambridge Univ. Press, 1994.
[43] A. Blake and M. Isard, "3D Position, Attitude and Shape Input Using Video Tracking of Hands and Lips," *Proc. ACM SIGGRAPH '94*, pp. 185-192, 1994.
[44] G.D. Hager and P.N. Belhumeur, "Real-Time Tracking of Image Regions With Changes in Geometry and Illumination," *Proc. Computer Vision and Pattern Recognition*, pp. 403-410, 1996.
[45] D.H. Ballard and C.M. Brown, *Computer Vision*. Prentice Hall, 1982.
[46] C. Tomasi and T. Kanade, "Shape and Motion From Image Streams Under Orthography: A Factorization Method," *Int'l J. Computer Vision*, vol. 9, no. 2, pp. 137-154, 1992.
[47] K. Jack, *Video Demystified: A Handbook for the Digital Engineer*. HighText Publications Inc., 1993.
[48] R.Y. Tsai, "A Versatile Camera Calibration Technique for High Accuracy 3D Machine Vision Metrology Using Off-the Shelf TV Cameras and Lenses," *IEEE Trans. Robotics and Automation*, vol. 3, no. 4, pp. 323-344, 1987.
[49] B. Boufama, D. Weinshall, and M. Werman, "Shape From Motion Algorithms: A Comparative Analysis of Scaled Orthography and Perspective," *Proc. European Conf. Computer Vision*, J.-O. Eklundh, ed., pp. 199-204, 1994.
[50] G. Turk and M. Levoy, "Zippered Polygon Meshes From Range Images," *Proc. SIGGRAPH '94*, pp. 311-318, 1994.
[51] E.K.-Y. Jeng and Z. Xiang, "Moving Cursor Plane for Interactive Sculpting," *ACM Trans. Graphics*, vol. 15, no. 3, pp. 211-222, 1996.
[52] S.W. Wang and A.E. Kaufman, "Volume Sculpting," *Proc. Symp. Interactive 3D Graphics*, pp. 151-156, 1995.
[53] H. Qin and D. Terzopoulos, "D-Nurbs: A Physics-Based Framework for Geometric Design," *IEEE Trans. Visualization and Computer Graphics*, vol. 2, no. 1, pp. 85-96, Mar. 1996.
[54] P.E. Debevec, C.J. Taylor, and J. Malik, "Modeling and Rendering Architecture From Photographs: A Hybrid Geometry- and Image-Based Approach," *Proc. SIGGRAPH '96*, pp. 11-20, 1996.
[55] S.A. Tebo, D.A. Leopold, D.M. Long, S.J. Zinreich, and D.W. Kennedy, "An Optical 3D Digitizer for Frameless Stereotactic Surgery," *IEEE Computer Graphics and Applications*, vol. 16, pp. 55-64, Jan. 1996.
[56] M. Agrawala, A.C. Beers, and M. Levoy, "3D Painting on Scanned Surfaces," *Proc. Symp. Interactive 3D Graphics*, pp. 145-150, 1995.
[57] K.N. Kutulakos and J. Vallino, *Affine Object Representations for Calibration-Free Augmented Reality: Example MPEG Sequences*. http://www.cs.rochester.edu:/u/kyros/mpegs/TVCG.html, 1996.
[58] C. Wiles and M. Brady, "On the Appropriateness of Camera Models," *Proc. Fourth European Conf. Computer Vision*, pp. 228-237, 1996.

[59] O.D. Faugeras, "What Can Be Seen in Three Dimensions With an Uncalibrated Stereo Rig?," *Proc. Second European Conf. Computer Vision*, pp. 563-578, 1992.
[60] R. Mohr, B. Boufama, and P. Brand, "Accurate Projective Reconstruction," *Applications of Invariance in Computer Vision*, J. Mundy, A. Zisserman, and D. Forsyth, eds., pp. 257-276. Springer-Verlag, 1993.
[61] J. Dorsey, J. Arvo, and D. Greenberg, "Interactive Design of Complex Time Dependent Lighting," *IEEE Computer Graphics and Applications*, vol. 15, pp. 26-36, Mar. 1996.
[62] A. Shashua, "Geometry and Photometry in 3D Visual Recognition," PhD thesis, Massachusetts Inst. of Technology, 1992.
[63] P.N.B.D.J. Kriegman, "What is the Set of Images of an Object Under All Possible Lighting Conditions," *Proc. Computer Vision and Pattern Recognition*, pp. 270-277, 1996.
[64] P.F. McLauchlan, I.D. Reid, and D.W. Murray, "Recursive Affine Structure and Motion From Image Sequences," *Proc. Third European Conf. Computer Vision*, pp. 217-224, 1994.
[65] L.O. Gorman, "Subpixel Precision of Straight-Edged Shapes for Registration and Measurement," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 18, no. 7, pp. 746-751, July 1996.
[66] K. Toyama and G.D. Hager, "Incremental Focus of Attention for Robust Visual Tracking," *Proc. Computer Vision and Pattern Recognition*, pp. 189-195, 1996.
[67] S.K. Nayar, "Catadioptric Omnidirectional Camera," *Proc. Conf. Computer Vision and Pattern Recognition*, pp. 482-488, 1997.

Kiriakos N. Kutulakos received a BS degree in computer science from the University of Crete, Greece, in 1988, and the MS and PhD degrees in computer science from the University of Wisconsin-Madison in 1989 and 1994, respectively. He joined the Computer Science Department at the University of Rochester in 1994, and was a U.S. National Science Foundation CISE postdoctoral research associate there until the spring of 1997. In the fall of 1997, he took the position of assistant professor of dermatology and computer science at the University of Rochester. His research interests include augmented reality, active and real-time computer vision, 3D shape recovery, and geometric methods for visual exploration and robot motion planning. Dr. Kutulakos was a recipient of the Siemens Best Paper Award at the 1994 IEEE Computer Vision and Pattern Recognition Conference for his work on visually exploring geometrically-complex, curved 3D objects. He is a member of the IEEE and the ACM.

James R. Vallino received a BE in mechanical engineering in 1975 from The Cooper Union and, in 1976, he received an MS degree from the University of Wisconsin in electrical engineering. From 1976 until 1993, he held industry positions involved with business terminal development, data acquisition, and communication protocols for process control and medical imaging. In 1993, he started PhD studies in computer science at the University of Rochester and anticipates completion in 1998. In December, 1997, he began as assistant professor of computer science at Rochester Institute of Technology. His professional interests include augmented and virtual reality, computer graphics, computer vision, and software engineering.