Manuscript No MM 000198 Resubmitted: August 26, 2002

Live 3-Dimensional Content for Augmented Reality

Simon Prince, Adrian David Cheok, Todd Williamson, Nik Johnson, Farzam Farbiz, Mark Billinghurst and Hirokazu Kato

Dept. of Electrical and Computer Engineering,
National University of Singapore

Submitted to:

IEEE Transactions on Multimedia

Correspondence Address:

Dr Adrian David Cheok,
Dept. of Electrical and Computer Engineering,
National University of Singapore,
4 Engineering Drive 3,
Singapore 113576.

Tel.:	+65 91831911
Fax:	+65 67791103
E-mail:	eleadc@nus.edu.sg

Live Three-Dimensional Content for Augmented Reality

Abstract:

We describe an augmented reality system for superimposing three-dimensional live content onto two-dimensional fiducial markers in the scene. In each frame, the Euclidean transformation between the marker and the camera is estimated. The equivalent “virtual view” of the live model is then generated and rendered into the scene at interactive speeds. The three-dimensional structure of the model is calculated using a fast shape-from-silhouette algorithm based on the outputs of 15 cameras surrounding the subject. The novel view is generated by projecting rays through each pixel of the desired image and intersecting them with the 3-d structure. Pixel color is estimated by taking a weighted sum of the colors of the projections of this three-dimensional point in nearby real camera images. Using this system, we capture live human models and present them via the augmented reality interface at a remote location. We can generate 384x288 pixel images of the models at 25 fps, with a latency of <100ms. The result gives the strong impression that the model is a real three-dimensional part of the scene.

Introduction

Augmented reality refers to the real-time insertion of computer-generated three-dimensional content into a real scene (see [2], [3] for reviews). Typically, the observer views the world through an HMD with a camera attached to the front. The video is captured, modified and relayed to the observer in real time. The degree to which this is successful depends largely on the accuracy of the tracking of the head mounted camera. A common tracking formulation is to introduce fiducial markers into the environment and then track the position of these relative to the camera using computer vision techniques. Since the properties of the marker are known a priori, it is relatively simple to calculate its three-dimensional position and orientation relative to the camera.

Early studies such as [8] superimposed two-dimensional textual information onto real world objects. However, it has now become common to insert three-dimensional dynamic graphical objects into the world (eg [12]). Billinghurst et al [4][5] used the augmented reality interface to display small 2-D video streams of collaborators into the world in a video-conferencing application. In paper, we extend these techniques by introducing a full three-dimensional live captured image of a collaborator into the visual scene for the first time (see Figure 1). As the observer moves his head, the view of the collaborator changes appropriately. This results in the stable percept that the collaborator is three dimensional and present in the space with the observer.

The ultimate goal of this research is to introduce the image of a collaborator into the environment with such accuracy that one cannot distinguish him/her from reality (i.e. perfect telepresence). Perhaps closest to this goal in existing published literature is the Office of the Future work [21], the Virtual Video Avatar of Ogi et al.[20], and the work of Mulligan and Daniilidis [18][19]. All of these systems use multiple cameras to construct a geometric model of the participant. They then use this model to generate the appropriate view for remote collaborators. Although undoubtedly impressive, none of these systems generate the whole 3D model – one cannot move 360^o around the virtual avatar – these systems are designed for display via a projection screen and hence are less suited to a portable augmented reality setting.

In order to introduce live captured 3D content into the world, we must solve several sub-problems. In particular, we require that for each frame:

- The pose of the head-mounted camera relative to the scene is estimated.
- The appropriate view of the collaborator is generated.
- This view is rendered into the scene, possibly taking account of occlusions.

The structure of the paper is as follows. Firstly, we present an overview of the system.. In the following section we discuss the robust determination of the camera pose relative to an object in the scene. We describe in detail a method based on fiducial markers. Subsequently, we present a novel shape from silhouette algorithm that generates an arbitrary view of a subject at frame rates. Finally, we discuss details of system performance, and compare our system to other related technologies. We conclude by discussing potential applications.

System Structure

The overall system structure is depicted in Figure 2. Fourteen Sony DCX-390 video cameras were equally spaced around the subject, and one viewed him/her from above. All intrinsic and extrinsic parameters of the cameras were found prior to rendering. The exact procedure is described later. Five video-capture machines received data from three cameras each. Each video-capture machine had Dual 1GHz Pentium III processors and 2Gb of memory. The video-capture machines pre-process the video frames and pass them to the rendering server via gigabit Ethernet links. The rendering server had a 1.7 GHz Pentium IV Xeon processor and 2Gb of memory.

Each video-capture machine receives the three 640x480 video-streams in YcrCb format at 30Hz and pre-processes the video streams by calculating the sillhouettes, compensating for the radial distortion component of the camera model, and applying a simple lossless compression technique. The sillhouette computation is described in more detail in a later section.

The user viewed the scene through a Daeyang Cy-Visor DH-4400VP head mounted display (HMD), which presented the same 640x480 pixel image to both eyes. A PremaCam SCM series color security camera was attached to the front of this HMD. This captures 25 images per second at a resolution of 640x480. For each frame, the camera image is processed, and the Euclidean transformation matrix relating the marker co-ordinates to the camera co-ordinates is extracted. This matrix is sent to the rendering server, which also receives input from the 5 video-capture servers. The rendering server then generates a novel view of the subject based on these video inputs. The novel image is generated such that the virtual camera views the subject from exactly the same angle and position as the head-mounted camera views the marker. A 384x288 pixel, 24bit color image, and a range estimate associated with each pixel is then returned to the augmented reality client. This simulated view of the remote collaborator is then superimposed on the original image and displayed to the user. The result is that the subject appears convincingly to be in the scene with the user.

CAMERA POSE ESTIMATION

In this section, we discuss how the pose of the head-mounted camera relative to the marker is established. We simplify the pose estimation problem by inserting 2-D square black and white fiducial markers into the scene. Virtual content is associated with each marker. The general problem of reconstructing camera pose given a known planar object in the scene is discussed in [26][30]. For this case, both the shape and pattern of these markers is known a priori, permitting very simple methods of locating the markers and calculating their position relative to the camera. The disadvantage of using such markers is that their insertion into the scene makes the world "less natural", and tracking breaks down if part or all of them move outside the camera's view. However, they also have several notable advantages: they permit tracking that is both extremely fast and robust. They also create a "tangible" interface to the virtual objects – by picking up and manipulating the marker, one can examine the three-dimensional virtual object and place it wherever one desires in the real world.

Figures 1 and 8 show examples of fiducial card markers of various scales. This marker, which is of known size, defines the basis of the marker coordinate system in which virtual objects are represented. The origin is in the center of the marker and the Z-axis projects at right angles from the card. We seek the Euclidean transformation matrix relating the marker coordinate system to the camera coordinates ($\mathbf{T}_{cm}$), represented in eq.1. This matrix consists of the rotation $\mathbf{R}_{3x3}$ and the translation $\mathbf{T}_{3x1}$.

$$\begin{bmatrix} X_c \\ Y_c \\ Z_c \\ 1 \end{bmatrix} = \begin{bmatrix} R_{11} & R_{12} & R_{13} & T_x \\ R_{21} & R_{22} & R_{23} & T_y \\ R_{31} & R_{32} & R_{33} & T_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_m \\ Y_m \\ Z_m \\ 1 \end{bmatrix}$$

$$= \begin{bmatrix} \mathbf{R}_{3\times3} & & & \mathbf{T}_{3\times1} \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_m \\ Y_m \\ Z_m \\ 1 \end{bmatrix} = \mathbf{T}_{\mathbf{cm}} \begin{bmatrix} X_m \\ Y_m \\ Z_m \\ 1 \end{bmatrix} \qquad \text{(eq.1)}$$

For a calibrated camera, it can be shown that the positions of the corners of the marker in the image uniquely identify its 3D position and orientation in the real world. Hence, we must first find the exact position of the four marker corners in the input image. The stages of this process are illustrated in Figure 3. The camera image is thresholded and contiguous dark areas are identified using a connected components algorithm. A contour seeking technique identifies the outline of these regions. Contours that do not contain exactly four corners are discarded, as are regions that are too large or too small to plausibly be markers. We estimate the corner positions by fitting straight lines to each edge using principle components analysis and determining the points of intersection of these lines. A projective transformation is used to map the enclosed region to a standard shape. This is then cross-correlated with stored patterns to establish the identity and orientation of the marker in the image.

Given the position of the four marker corners in the image, we now attempt to find the transformation matrix. The overall strategy is to find an initial estimate for the rotation matrix, $\mathbf{R}_{3x3}$, and then find the best translation vector, $\mathbf{T}_{3x1}$, given that this rotation matrix is correct. We then use this overall transformation as an initial estimate in a non-linear optimization procedure which aims to minimize the difference between the predicted and actual corner position. We consider each stage in turn:

Step 1: Estimation of $\mathbf{R}_{3x3}$

The rotation matrix is found by finding the vanishing points in the x and y marker directions in the image. We consider where the parallel sides of the square marker intersect in the image plane. The three-dimensional vectors projecting from the origin through these points are known to be mapped to (1,0,0,0) and (0,1,0,0) respectively by the correct rotation matrix. The camera coordinate frame is transformed to the ideal screen coordinates by the internal camera matrix **P**:

$$\mathbf{P} = \begin{bmatrix} P_{11} & P_{12} & P_{13} & 0 \\ 0 & P_{22} & P_{23} & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad \begin{bmatrix} hx_c \\ hy_c \\ h \\ 1 \end{bmatrix} = \mathbf{P} \begin{bmatrix} X_c \\ Y_c \\ Z_c \\ 1 \end{bmatrix} \qquad \text{(eq.2)}$$

Following the image processing stage, we know the equations of parallel line segments in the ideal screen coordinates:

$$a_1 x + b_1 y + c_1 = 0 \qquad \text{(eq. 3)}$$

$$a_2 x + b_2 y + c_2 = 0$$

The equations of the planes defined by these two lines in the image plane, and the camera center can hence be found by by substituting x_c and y_c in (eq.2) for x and y in (eq. 3)

$$a_1 P_{11} X_c + (a_1 P_{12} + b_1 P_{22}) Y_c + (a_1 P_{13} + b_1 P_{23} + c_1) Z_c = 0$$
$$a_2 P_{11} X_c + (a_2 P_{12} + b_2 P_{22}) Y_c + (a_2 P_{13} + b_2 P_{23} + c_2) Z_c = 0 \qquad \text{(eq.4)}$$

The 3D vector along the intersection of these planes, **u**, is given by the outer product of their normals, $\mathbf{n_1}$ and $\mathbf{n_2}$. The two vectors, $\mathbf{u_1}$ and $\mathbf{u_2}$, obtained from the two sets of parallel sides should theoretically be perpendicular. However, image processing and numerical errors mean that the vectors are not exactly perpendicular in practice. We must hence adjust these vectors by forcing them to become perpendicular as shown in Figure 4. We denote the adjusted vectors**,** $\mathbf{v_1}$ and $\mathbf{v_2}$. This can also be done using the SVD [30]. Given that the unit direction vector which is perpendicular to both $\mathbf{v_1}$ and $\mathbf{v_2}$ is $\mathbf{v_3}$, the rotation component $\mathbf{R_{3x3}}$ in the transformation matrix $\mathbf{T_{cm}}$ from marker coordinates to camera coordinates specified in eq.1 is $[\mathbf{v_1}^\mathbf{T}\ \mathbf{v_2}^\mathbf{T}\ \mathbf{v_3}^\mathbf{T}]$.

Step 2: Estimation of $\mathbf{T_{3x1}}$

Since the rotation component $\mathbf{R_{3x3}}$ in the transformation matrix $\mathbf{T_{cm}}$ is now known, one can now use the camera projection equations to solve for the translation vector:

$$\begin{bmatrix} hx_c \\ hy_c \\ h \\ 1 \end{bmatrix} = \mathbf{P} \begin{bmatrix} R_{11} & R_{12} & R_{13} & T_x \\ R_{21} & R_{22} & R_{23} & T_y \\ R_{31} & R_{32} & R_{33} & T_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_m \\ Y_m \\ Z_m \\ 1 \end{bmatrix} \qquad \text{(eq. 5)}$$

We know the positions of the four corners relative to the marker center, and where they project to into the image, as well as the components of **R** and **P**. Each corner generates two independent equations that can be used to solve for the components of the translation vector. We employ an overdetermined least-squares solution based on all eight resulting equations.

Step 3: Optimisation of R_{3x3} and T_{3x1}

The transformation matrix will generally contain some error. We improve the estimate in the following way: Using Equation 5, we can predict where each vertex will appear in the image given a certain estimate of the transformation matrix. We now perform a non-linear minimization of the transformation matrix to attempt to minimize the difference between the predicted positions of the corners and the actual positions. There are six independent variables in the transformation matrix. However, in order to reduce the computational cost, we only optimize the rotation components and then re-estimate the translation components by using the method in step 2. By iteration of this process a number of times the transformation matrix is more accurately found.

In order to evaluate accuracy of this marker detection technique, the detected position and pose were recorded while a square marker 80 mm wide was moved perpendicular to the camera and tilted at different angle. Figure 5 shows errors of position. Accuracy decreases the further the cards are from the camera and the further they are tilted from the camera. The complete software for this AR tracking procedure can be downloaded from [31].

NOVEL VIEWPOINT GENERATION

Background

Given a transformation matrix relating the camera to the marker, we now wish to generate a view of collaborator from the same position for each video frame. Most approaches to this problem is to develop a 3D depth reconstruction of the collaborator, from which an arbitrary view can be generated (see [27] for a review of representational possibilities). For example, Depth information could be garnered using stereo-vision. Stereo reconstruction can been achieved at frame rate [11][18][19] although the processing overhead is high. Moreover, the resulting dense depth map requires smoothing constraints to increase the robustness and prevent artifacts in the novel views.

A related approach is image-based rendering, which sidesteps depth-reconstruction by warping between several captured images of an object to generate the new view. Seitz and Dyer [22] presented the first image-morphing scheme that was guaranteed to generate physically correct views, although this was limited to novel views along the camera baseline. Avidan and Shashua [1] presented a more general scheme that allowed arbitrary novel views to be generated from a stereoscopic image pair, based on the calculation of the tri-focal tensor. Although depth is not explicitly computed in these methods, they still require the computation of dense matches between multiple views and are hence afflicted with the same problems as depth from stereo.

A more attractive approach to fast 3D model construction is shape-from-silhouette. A number of cameras are placed around the subject. We classify each pixel in each camera as either belonging to the subject (i.e. the sillhouette) or the background. Each pixel in each camera collects light over a (very narrow) rectangular-based pyramid in 3D space, where the vertex of the pyramid is at the focal point of the camera and the pyramid extends infinitely away from this. For background pixels, this space can be assumed to be unoccupied. Shape-from-silhouette algorithms work by initially assuming that space is completely occupied, and using each background pixel from each camera to carve away pieces of the space to leave a representation of the foreground object.

Intuitively, the reconstructed model will improve with the addition of more cameras. However, it can be proven that the resulting depth reconstruction may not capture all aspects of the true shape of the object, even given an infinite number of cameras. The actual reconstructed shape was termed the "visual hull" by Laurentini [14], who did the initial work in this area. This may be an important limitation for rendering man-made objects which have sharp corners and concavities, but is less important for capturing human beings, whose visual hull is generally quite close to their true shape. Despite this bias in the final estimated structure, shape-from-silhouette has three significant advantages over competing technologies. First, it is more *robust* than stereo vision – it only requires the reliable segregation of the silhouette to produce consistent results. Second, it is significantly *faster* than either stereo, which requires vast computation to calculate cross-correlation, or laser range scanners, which generally have a slow update rate. Third since it relies only on mesaurements from cameras, it is *inexpensive* relative to dedicated scanners which use specialized hardware.

For these reasons, the system described in this paper is based on shape-from-sillhouette information. There are two main approaches to shape-from-sillhouette: one possibility is to generate a full volumetric model in which voxels are explicitly [13] or implicitly [24] denoted as being part of the object and then project this model down to create a novel view. This is the approach taken by a class of methods known as "space carving" techniques, which use both silhouette and other photo-consistency information to eliminate voxels in an initially full volume [6][13][23][25][28][29]. These methods are mostly very slow (varying from seconds to hours depending on the specific method) although recently [6] [15] have produced methods based on silhouette alone which run at frame rate. One inherent aspect of these systems which makes them undesirable for real-time applications is that the resolution of the voxel model must remain relatively low, as the computational demands obviously scale with the cube of the voxel diameter. Typical current systems use volumes that are only 100x100x100 pixels,

Our algorithm takes a different approach and constructs the novel image on a pixel by pixel basis by raycasting. This follows from the pioneering work of Matusik et al [16] who presented the first fast image- based novel view generation algorithm based on shape from sillhouette. We present a detailed comparison of our algorithm with competing systems following the algorithm description.

Novel View Generation Algorithm

The "virtual camera" can be completely described by taking the product of the (head mounted) camera calibration matrix and the estimated transformation matrix. Given this 4x4 camera matrix, the center of each pixel of the virtual image is associated with a ray in space that starts at the camera center and extends outward. Any given distance along this ray corresponds to a point in 3D space. We calculate an image based depth representation by seeking the closest point along this ray that is inside the visual hull. This 3D point is then projected back into each of the real cameras to obtain samples of the color at that location. These samples are then combined to produce the final virtual pixel color. In summary, the algorithm must perform three operations for each virtual pixel:

- Determine the *depth* of the virtual pixel as seen by the virtual camera.
- Find corresponding pixels in nearby real images
- Determine pixel color based on all these measurements.

We consider each of these operations in turn.

Determining Pixel Depth

The depth of each virtual pixel is determined by an explicit search starting at the virtual camera projection center and proceeding outward along the ray corresponding to the pixel center (see Figure 6). Each candidate 3D point along this ray is evaluated for potential occupancy. A candidate point is unoccupied if its projection into *any* of the silhouettes is marked as background. When a point is found for which all of the silhouettes are marked as foreground, the point is considered occupied, and the search stops. The search is made efficient using a number of heuristics. Notice that for the search to terminate, the point in space must project to the silhouette in every image. Hence, for the majority of points we need to test only one image to reject it. We consider first the camera which is most aligned with the virtual camera. We project the start and end points of the ray into this image and then test each pixel along this line. The camera is closely aligned with the virtual image, so a small change in position in the image corresponds to a large change in position in space, and we can eliminate most pixels very quickly. When a silhouette pixel is found along the line, we establish which 3D point this corresponds to, and project this into the next most aligned image. If we do this for the most closely aligned few images, we can establish approximately where the visual hull lies very quickly. Starting with this approximate estimate, we then refine the depth value using a linear search regime along the ray in three-dimensional space.

To constrain the search the corresponding ray is intersected with the boundaries of each image. We project the ray into each real image to form the corresponding epipolar line. The points where these epipolar lines meet the image boundaries are found and these boundary points are projected back onto the ray. The intersections of these regions on the ray define a reduced search space. This is equivalent to making the assumption that the subject is completely visible in all of the cameras. If the search reaches the furthest limit of the search space without finding an occupied point, the virtual pixel is marked as background.

Finding Corresponding Pixels in Real Images

The resulting depth is an estimate of the closest point along the ray that is on the surface of the visual hull. However, since the visual hull may not accurately represent the shape of the object, this 3D point may actually lie outside of the object surface. Hence, care needs to be taken in choosing the cameras from which the pixel colors will be combined (see Figure 7). Depth errors will cause incorrect pixels to be chosen from each of the real camera views. We aim to minimize the visual effect of these errors. In general it is better to choose incorrect pixels that are physically closest to the simulated pixel. So the optimal camera should be the one minimizing the angle between the rays corresponding to the real and virtual pixels. For a fixed depth error, this minimizes the distance between the chosen pixel and the correct pixel. We rank the camera's proximity once per image, based on the angle between the real and virtual camera axes.

We can now compute where the virtual pixel lies in each candidate camera's image. Unfortunately, the real camera does not necessarily see this point in space - another object may lie between the real camera and the point. If the real pixel is occluded in this way, it cannot contribute its color to the virtual pixel. In order to test whether this is the case, our basic approach is to run the depth search algorithm on a pixel from the real camera. If the recovered depth lies close enough in space to the 3D point computed for the virtual camera pixel, we assume the real camera pixel is not occluded - the color of this real pixel is allowed to contribute to the color of the virtual pixel. In practice, we increase system speed by immediately accepting points that are geometrically certain not to be occluded.

Determining Virtual Pixel Color

After determining the depth of a virtual pixel and which cameras have an unoccluded view, all that remains is to combine the colors of real pixels to produce a color for the virtual pixel. The simplest method would be to choose the pixel from the closest camera. However, this produces sharp images that often contain visible borders where adjacent pixels were taken from different cameras. Pixel colors might vary between cameras for several reasons. First, different cameras may have slightly different spectral responses. Second, the 3D model is only an approximate, and therefore the pixels in different cameras may not receive light from the same 3D point. Third, unless the bi-directional reflectance distribution function is uniform and the original lighting was completely ambient, the actual reflected light will vary at different camera vantage points.

In order to compensate for these effects, we average together the colors of several candidate pixels. The simplest and fastest method is to take a straight average of the pixel color from the N closest cameras. This method produces results that contain no visible borders within the image. However, it has the disadvantage that it produces a blurred image even if the virtual camera is exactly positioned at one of the real cameras. Hence, we take a weighted average of the pixels from the closest N cameras, such that the closest camera is given the most weight. This method produces better results than either of the others, but requires more substantial computation.

Sillhouette Computation

A crucial element of this system is the ability to cleanly and successfully separate the foreground objects from the background to create the silhouettes. There are a number of sophisticated methods in the literature for this type of separation such as [7][10] . However, it is imperative that our method be extremely fast as the video capture machines, must process images from 3 cameras at 90 Hz. Hence, we base our method on a simple statistical decision. Before recording the subject, we pre-capture several thousand frames of the empty studio, and use this raw data to construct a statistical model for the luminance variation at each pixel. In particular, we model this distribution as a multi-variate Gaussian in RGB space. In order to compensate for the possibility of shadows, we artificially increase the variance of this distribution along the intensity axis by a constant factor. We then classify each pixel in each input image based on the squared Mahalanobis distance:

$$d = (\mathbf{x} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\mathbf{x} - \boldsymbol{\mu})$$

where $\mathbf{x}$ is the RGB vector corresponding to the current pixel, $\boldsymbol{\mu}$ was the mean of the pre-captured distribution and $\boldsymbol{\Sigma}$ is the pre-captured pixel covariance. While this system is extremely simple and fast to implement it naturally misclassifies pixels where the background and foreground are similar pixels. In practice, most pixels are are classified correctly as we use a bright green background in our studio, and use multiple diffuse light sources to reduce the effect of shadows. We apply morphological operators to perform opening and closing operations. This eliminates isolated erroneous pixels. We set a deliberately conservative threshold for rejecting pixels as being part of the silhouette. This ensures that the silhouette is captured. The algorithm requires that a point must project to within the silhouette in all of the images to be considered part of the object. Hence, the effect of the mis-classified pixels is not great, as the same point in space must be mis-classified in a number of images to effect the final result.

SYSTEM CONFIGURATION AND CALIBRATION

Configuration

Since we assume that each foreground object must be completely visible from all of the cameras, the zoom level/ position of each camera must be adjusted so that it can see the subject, even as he/she moves around. This means that the limited resolution of each camera must be spread over the desired imaging area. Hence, there is an inevitable trade-off between image quality and the volume that is captured. Similarly, the physical space needed for the system is determined by the size of the desired capture area and the field of view of the lenses used. We have experimented with a 2.8 mm lens that provides approximately a 90 degree field of view. With this lens, it is possible to capture a space that is 2.5m high and 3.3m in diameter with cameras that are 1.25 meters away. The figures in this paper were created with this configuration. It would however, be perfectly possible to generate very high quality images of just the upper torso, similar to the stereo-vision based virtual video avatars of [18][19]. The accuracy of the approximation to the visual hull also varies with the camera distance – if one assumes that the sillhouette is calculated to within a single pixel accuracy, then the depth error will be of the order of 0.5 cms in our current configuration, depending on exactly where in the volume the pixel is, and in which camera, the point projects to the bounding edge.

System Calibration

In order to create the illusion that the subject is in the space with the user, it is imperative to calibrate both the AR camera attached to the head-mounted display, and the multi-camera recording set-up which captures the subject. We consider each in turn.

There are two reasons why it is crucial to estimate accurately the intrinsic parameters of the head-mounted camera for augmented reality. First, we must simulate the projective camera parameters in order to realistically render any kind of three-dimensional object into the scene. Second, we wish to compensate for any radial distortion when we display captured video to the user. In the absence of radial distortion, straight lines in the world generate straight lines in the image. Hence, we fit straight lines to the image of a regular 2D grid of points. We search the distortion parameter space exhaustively to maximize goodness of fit. We estimate the center point of the distortion and the second order distortion co-efficient. The camera perspective projection parameters (focal length and principal point) are estimated using a regular 2-D grid of dots. Given the exact position of each point relative to the grid origin, and the corresponding image position, one can trivially solve for these parameters using the projection equations.

For the virtual view generation capture cameras, we must not only estimate the internal parameters, but also the spatial transformation between each of the cameras. As described above, it is crucial that we know exactly where a given point in the three-dimensional space will project to in each of the cameras. Calibration data is gathered by presenting a large checkerboard to all of the cameras. For our calibration strategy to be successful, it is necessary to capture many views of the target in a sufficiently large number of different positions. Standard routines from Intel's OpenCV library [32] are used to detect all the corners on the checkerboard, in order to calculate both a set of intrinsic parameters for each camera and a set of extrinsic parameters relative to the checkerboard's coordinate system. Where two cameras detect the checkerboard in the same frame, the relative transformation between the two cameras can be calculated. By chaining these estimated transforms together across frames, the transform from any camera to any other camera can be derived.

Each time a pair of cameras both see the calibration pattern in a frame, we calculate the transformation matrix between these camera positions. We consider this to be one estimate of the true transform. Given a large number of frames, we generate a large number of these estimates that may differ considerably. We wish to combine these measurements to attain an improved estimate. One approach would be to simply take the mean of the estimates, but better results can be obtained by removing outliers before averaging using a RANSAC procedure [9]. For each camera pair, a relative transform is chosen at random and a cluster of similar transforms is selected, based on proximity to the randomly selected one. This smaller set is averaged, to provide an improved estimate of the relative transform for that pair of cameras. These stochastically chosen transforms are then used to calculate the relative positions of the complete set of cameras relative to a reference camera.

Since the results of this process are heavily dependent on the initial randomly chosen transform, we repeat it several times to generate a family of calibration sets. We then pick the "best" of all these calibration sets. For each camera, the point at which the corners of the checkerboard are detected corresponds to a ray through space. With perfect calibration, all the rays describing the same checkerboard corner will intersect at a single point in space. In practice, calibration errors mean that the rays never quite intersect. The "best" calibration set is defined to be the set for which these rays most nearly intersect. We solve for the point in space where the Euclidean distance from all the rays is minimal, and use the sum of squared distances at this point as the error metric.

SYSTEM PERFORMANCE

The virtual content in the figures in this paper as generated at 384x288 resolution at 25 fps with a latency of <100ms, based on a 15 camera recording system. To maintain such high speeds, we introduce a single frame delay into the presentation of the augmented reality video. Hence, the augmented reality system starts processing the next frame while the virtual view server generates the view for the previous one. A swap then occurs. The graphics are returned to the augmented reality system for display, and the new transformation matrix is sent to the virtual view renderer. The delay ensures that neither machine wastes significant processing time waiting for the other and a high throughput is maintained. Rendering speed scales linearly with the number of pixels in the image, so it is quite possible to render more detailed images at the expense of this frame rate. However, for augmented reality applications, a high frame rate (>15 fps) is crucial if the illusion of interactivity is to be maintained. Rendering speed scales sub-linearly with the number of cameras, and image quality could be improved by adding more. Although the latency of the whole system is 100ms, the delay in rendering the geometry from the current pose is only of the order of a single frame (~40ms). Hence the pose of the model changes rapidly and interactively as the user manipulates the marker, but the actual content shown has an additional delay, due to the time taken to pre-process the images.

RELATED WORK

Our system is similar in spirit the work of Matusik et al. [15] who have also presented an image-based novel view generation algorithm using sillhouette information. In common with our system, they generate a view-dependent representation of the visual hull. The principal differences are that (i) Matusik et al. generate the entire visual hull from the current camera angle, whereas we generate only the visible part. (ii) Matusik et al. bin the sillhouette data and hence generate only an approximation to the visual hull whereas we construct it exactly. Matusik et al [16] have also since developed a polyhedral rendering which does reconstruct the exact visual hull. Lok [15] has proposed a volume based approach to shape from sillhouette. Essentially each voxel in the imaged volume is assessed for occupancy in turn. Although this procedure is extremely computationally expensive, the system performs the bulk of the processing on the graphics card, allowing reasonable speed. The advantage of this approach is that the resulting volumetric data allows simple interaction with virtual environments and can more easily be viewed from multiple angles simultaneously. This is an interesting approach, but unfortunately, the computation scales scales with the cube of the rendering resolution and hence methods based on this scheme are unlikely to produce very high quality results in the near future even with the help of graphics hardware.

Moreover, all of the above systems scale poorly with the number of cameras – this is unfortunately especially true of the polyhedral representation which needs to intersect every silhouette edge in a given image with every edge in every other image and hence scales with the square of the number of cameras. – Our system scales very slowly in practice, since the pixel color estimation (which takes the bulk of the rendering time) only uses of a fixed number of camera images, rather than all of them and most of the depth search is limited to a few strategically chosen cameras. The characteristics of our algorithm allow us to generate very high quality models based on 15 cameras very quickly. To the best of our knowledge, the closest comparable system employs only 5 cameras and model quality suffers accordingly. Moreover, most existing systems have not produced frame rates that are high enough for interactive augmented reality display.

APPLICATIONS

The system has a number of obvious real-world applications (see Figures 8,9 and 10). First, since the model capture and the augmented reality display both operate in real time, this system could form the basis for a three-dimensional video-conferencing and collaboration system. This is the natural development of the work of [4][5] who presented 2d images of collaborators in augmented reality space. AR conferencing has a number of advantages over traditional video conferncing: users can arrange markers representing collaborators about them to create a virtual spatial conferencing space. The user is no longer tied to the desktop and can potentially conference from any location, so the remote collaborators become part of any real world surroundings, potentially increasing the sense of social presence. A particular advantage of our system is that it can capture a large space and hence can represent non-verbal communication such as gestures. It can also capture more than one subject at once without any modification. Since only one image is sent by the server per frame, the bandwidth requirements of the system are no greater than for traditional video conferencing. In fact, because of the inherent foreground/background separation in the virtual view generation, the potential for compression is even greater than for conventional conferencing. However, in order to maintain the interactive nature of augmented reality, the quality of service must be excellent.

In fact, the large space that is imaged is a design choice that was chosen to demonstrate large scale movements and gestures. The method of display on a small card-marker on a table top is especially suited for viewing these movements as the large area is scaled to a smaller space and can be manipulated easily to view a collaborator from the desired angle. We make this the primary demonstration of our technology as the ability to capture such movements is a novel aspect of our system that cannot be achieved by competing methods. However, one disadvantage of this arrangement is that for bi-directional systems is that there is an asymmetry in the viewing positions. Consider two users. It is certainly possible to indicate to each the viewing direction of the other so that they can orient themselves correctly. However, the user cannot simultaneously orient himself to the collaborators viewing direction (presumably above his head) and to his view of the collaborator on his card marker (presumably below his head).

The obvious solution to these problems is to render the collaborator at life-size. Relative body posture and orientation of users then remains exactly the same as in real life. In such a situation, one might wish to to move the cameras much closer and image a smaller volume at much higher resolution, similar to conferencing presented in [18]. It is still possible to track the user's head movements in this configuration using fiducial markers on the floor or walls. However, in practice, subjects report difficult keeping the marker in the scene and viewing the collaborator simultaneously and feature tracking or a commercial virtual reality tracking solution are superior solutions. We have experimented using an Intersense IS 900 tracker and the result is extremely stable. In this case, communication is as close to real-life as we can simulate. The collaborator appears in full 3-D at real scale in the user's space. This allows spatial relationships between participants to be maintained in addition to the very natural visualisation that this technology permits.

Our system is currently unidirectional – the user can see the collaborator but not vice versa. A symmetrical relationship introduces a further complication: completely natural communication is disrupted because mutual eye contact cannot be maintained when head-mounted displays are worn. One obvious solution is to mediate augmented reality conferencing with optical "see through" head-mounted displays in which the eyes can still be seen.

A second application of the system is for visualization of collaborators with virtual environments. A symbolic graphical representation (avatar) has previously been used to represent a collaborator in virtual reality. Indeed, considerable research effort has been invested in identifying those non-verbal behaviors that are crucial for collaboration [6] and elaborate interfaces have been developed to control expression in avatars. Our system allows us to replace the symbolic avatar with a simulated view of the actual person as they explore the virtual space in real time. Since a large space is captured, the system allows the collaborator to move around and gesture naturally to objects within the virtual space. We present the example of a guide to a virtual art gallery.

A third application of the 3-D live system is to develop an augmented book in which a different fiducial marker is presented on each page. Associated with each is virtual content consisting both of 3-D graphics and a narrator who was captured in our system. The system can also be applied for training. Pre-recorded sequences like the ballet-dancers and karate demonstrations illustrated in Figure 9, can be played back in slow motion or even paused and viewed from any angle. This can be used to facilitate learning of these complex physical skills.

Finally, we present a novel entertainment application that utilizes the "tangible" user interface created by associating virtual content with real world card markers (see Figure 10). The 3-D live system presented here allows us to visualize a player inside a virtual reality game-space. We view a miniaturized version of the virtual environment containing the player who is being tracked using standard VR technology. We can use a card marker in the shape of a paddle to manipulate the position of the player in the environment. We can also use the paddle to interact with him in a natural way. We have developed a game in which we use the paddle to try and drop virtual objects onto the player's head. Objects are selected by moving the paddle close to them. We drop objects by tilting the paddle relative to the table. The player attempts to jump out of the way of the falling objects in real time. We implement a crude collision detection routing based on the depth information retrieved from the rendering algorithm.

CONCLUSION

In this paper, we have described a complete system for live capture of a collaborator and realistic augmented reality presentation. We believe that this is a significant step towards the goal of perfect "tele-presence" for remote collaborations. This research developed an augmented reality system for superimposing three-dimensional live content onto two-dimensional fiducial markers in the scene. The three-dimensional structure of the model was calculated using a fast shape-from-silhouette algorithm based on the outputs of 15 cameras surrounding the subject, and the novel view is generated by projecting rays through each pixel of the desired image and intersecting them with the 3-d structure. Using this system, we capture live human models and present them via the augmented reality interface at a remote location.

Future work will focus on applying space-carving methods to improve the reconstruction quality, and relaxing the assumption that the subject must be seen in all of the images to increase the imaged volume.

REFERENCES

[1] S. Avidan and A. Shashua. Novel View Synthesis by Cascading Trilinear Tensors. IEEE Transactions on Visualization and Computer Graphics, 4(4): 293-305, October-December 1998.

[2] R.T. Azuma. A survey of augmented reality. Presence, 6(4): 355-385, August 1997.

[3] R. Azuma, Y. Baillot, R. Behringer, S. Feiner, S. Julier and B. MacIntyre. Recent Advances in Augmented Reality. IEEE Computer Graphics and Applications, 21(6): 34-37, November/December 2001.

[4] M. Billinghurst and H. Kato. Real World Teleconferencing. In Proceedings of CHI'99 Conference Companion ACM, New York, 1999.

[5] M. Billinghurst, I. Poupyrev, H. Kato and R. May. Mixing realities in shared space: An augmented reality interface for collaborative computing. IEEE International Conference on Multimedia and Expo, New York, July 2000.

[6] G.K.M. Cheung, T. Kanade, J.Y. Bouguet and M. Holler. A real time system for robust 3d voxel reconstruction of human motions. In Proc. CVPR, 714-720, 2000.

[7] A. Elgammal, D. Harwood and L.S. Davis. Non-parametric model for background subtraction. In Proc. ICCV Frame-Rate Workshop, Corfu, Greece, September 1999.

[8] S. Feiner, B. MacIntyre, M. Haupt and E. Solomon. Windows on the World: 2D Windows for 3D Augmented Reality. In Proceedings of UIST 93, pages 145-155, Atlanta, Ga, 3-5 November, 1993.

[9] M. A. Fischler and R. C. Bolles. Random sample concensus: a paradigm for model fitting with applications to image analysis andautomated cartography. Comm. Assoc. Comp. Mach., 24(6):381-395, 1981.

[10] T. Horprasert, D. Harwood and L.S. Davis. A statistical approach for robust background subtraction and shadow detection. Proc. IEEE ICCV'99 Frame Rate Workshop, Kerkyra, Greece, Sept. 1999.

[11] T. Kanade, H. Kano, S. Kimura, A. Yoshida and O. Kazuo "Development of a Video-Rate Stereo Machine." Proceedings of International Robotics and Systems Conference, pages 95-100, Pittsburgh, PA, August 1995.

[12] H. Kato, M. Billinghurst, I. Poupyrev, K. Inamoto and K. Tachibana. Virtual Object Manipulation on a table-top AR environment. Proceedings of International Symposium on Augmented Reality, 2000.

[13] K. Kutulakos and S. Seitz. A theory of shape by space carving. International Journal of Computer Vision, 38,(3), July 2000 pp. 199-213.

[14] A. Laurentini. The Visual Hull Concept for Sillhouette Based Image Understanding. IEEE PAMI, 16(2): 150-162, February 1994.

[15] B. Lok. Online model reconstruction for interactive virtual environments. Proceedings 2001 Symposium on Interactive 3D Graphics, Chapel Hill, NC 18-21 March 2001, 69-72

[16] W. Matusik, C. Buehler, R. Raskar, S.J. Gortler and L. McMillan. Image-Based Visual Hulls. SIGGRAPH 00

Conference Proceedings, Annual Conference Series, pages 369-374, 2000.

[17] W. Matusik, C. Buehler and L. McMillan. Polyhedral visual hulls for real-time rendering. In Proc. Eurogrpahics Workshop on Rendering 2001.

[18] J. Mulligan, V. Isler and K. Daniilidis. Performance evaluation of stereo for telepresence. 8'th IEEE International Conference on Computer Vision (ICCV'01), Vancouver.

[19] J. Mulligan and K. Daniilidis. Real time trinocular stereo for tele-immersion. Proceedings of the 2001 international conference on image processing (ICIP '01). Thessaloniiki, Greece.

[20] T. Ogi,, T. Yamada, K. Tamagawa, M. Kano and M. Hirose, Immersive Telecommunication Using Stereo Video Avatar. IEEE VR 2001, pages 45-51, IEEE Press, March 2001.

[21] R. Raskar, G. Welch, M. Cutts, A. Lake, L. Stesin and H. Fuchs. The Office of the Future: A unified approach to image based modeling and spatially immersive displays. pages 179-188, ACM SIGGRAPH, 1998

[22] S.M. Seitz and C.R. Dyer. View morphing. SIGGRAPH 96 Conference Proceedings, Annual Conference Series, pages 21-30. ACM SIGGRAPH 96, August 1996

[23] S.M. Seitz and C.R. Dyer. Photorealistic scene reconstruction by voxel colouring. IJCV 35(2): 1-23, 1999.

[24] G. G. Slabaugh, R. W. Schafer and M. C. Hans. Multi-resolution space carving using level set methods. In Proc. ICIP, Sept. 2002.

[25] D. Snow and P. Viola and R. Zabih. Exact Voxel Occupancy with Graph Cuts. In Proc. CVPR, 2000.

[26] P. Sturm. Algorithms for plane based pose estimation. In *Proc. CVPR*, pp. 1010-1017, 2000.

[27] R. Szeliski. Stereo algorithms and representations for image-based rendering. In Proc. BMVC, 314-328. Nottingham, England, 1999.

[28] S. Vedula, S. Baker and T. Kanade. Spatio-temporal view interpolation. In Proc. Eurographics, 2002.

[29] S. Vedula, S. Baker, S. Seitz and T. Kanade. Shape and motion carving in 6D. In Proc. CVPR, 2592-, June 2000.

[30] Z. Zhang. Flexible Camera Calibration By Viewing a Plane From Unknown Orientations. In *Proc. ICCV* , pp. 666-673,1999.

[31] ARToolkit library Version 2.51, Mar. 2002. http://www.hitl.washington.edu/artoolkit/

[32] Intel. Open source computer vision library version 0.0.7, Jun 2001. http://www.intel.com/research/mrl/research/**opencv**/.

FIGURE LEGENDS

Figure 1. We present a system that introduces live three-dimensional content into the real world. The observer views the scene through a head mounted display (HMD) with a camera attached to the front. The system measures the position of the card marker relative to this camera and draws the appropriate view of the captured subject into the world. To the observer it appears that the subject is a real 3-dimensional part of the world.

Figure 2. System Diagram. Fifteen cameras film the subject from different angles. Five computers pre-process the image to find the silhouettes and pass the data to the rendering server. The mixed reality client machine takes the camera output from the head mounted display and calculates the pose of the marker (see Figure 2). It then passes this information to the rendering server that returns the appropriate image of the subject. This is rendered into the users view in real time.

Figure 3. Marker detection and pose estimation. The image is thresholded and connected components are identified. Edge pixels are located, and corner positions are accurately measured. These positions determine the orientation of the virtual content. Region size, number of corners and template similarity are used to reject other dark areas in the scene.

Figure 4. Correction of the estimated vectors to the vanishing points in the x- and y-direction. In principle these should be perpendicular, but in practice they are not due to image processing errors. We symmetrically correct the vectors to ensure that they are normal to one another. This in turn ensures that the non-linear constraints on the components of the estimated rotation matrix are enforced.

Figure 5. Measured error in estimating the pose of the fiducial markers. The abcissa shows the error in arc minutes, and the ordinate shows the absolute distance to the marker. This is illustrated for a number of absolute marker orientations.

Figure 6. Virtual viewpoint generation by shape from silhouette. An explicit depth search is carried out along the ray passing through each virtual pixel. Points along this ray that project into the background in any camera are rejected. The points from A to C have already been processed and project to background in both images, so are marked as unoccupied. Points yet to be processed are marked in yellow. Point D is in the background in the silhouette from camera 2, so it will be marked as unoccupied and the search will proceed outward along the line.

Figure 7. The difference between the visual hull and the actual 3-D shape. The point on the visual hull does not correspond to a real surface point, so neither sample from the real cameras is appropriate for virtual camera pixel B. In this case, the closer real camera is preferred, since its point of intersection with the object is closer to the correct one.

Figure 8. Example results. The system produces highly realistic looking results at a fast frame rate that are suitable for video conferencing in either a table-top or life-size context.

Figure 9. We can also record sequences to disk and play them back at a later time. Example applications might include training for complex tasks such as karate or dancing (top row). Users can pause the action at any point and examine the frozen model from any angle. We have also developed an augmented book containing conventional graphics and 3D-Live content (bottom row). Notice the occlusion by the conventional graphics in the last frame which can be achieved easily as our algorithm naturally provides a depth estimate at each pixel.

Figure 10. An AR entertainment application. The user views a collaborator who is immersed in a virtual reality space. The user can move the collaborator from one virtual environment to another using the tangible card marker. The user then attempts to drop objects onto the subject who jumps out of the way in real time.

FIGURE 1

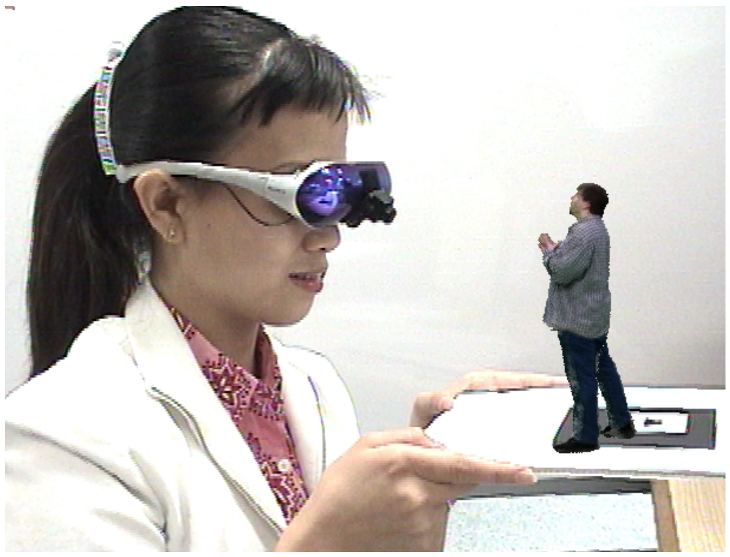


FIGURE 2

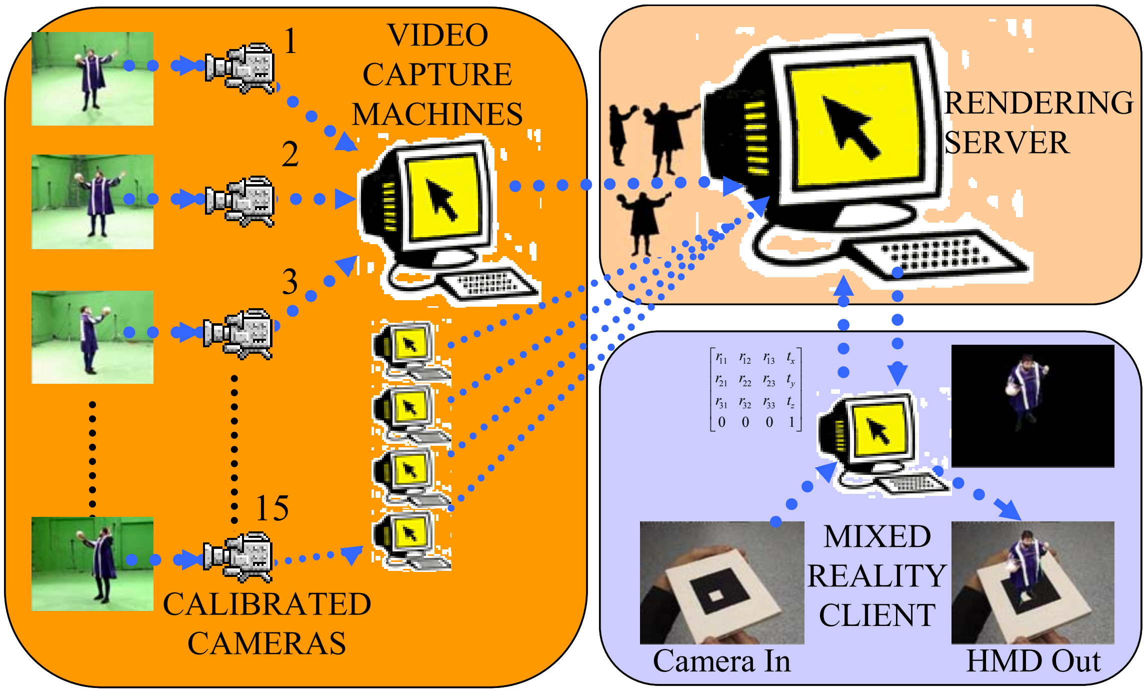

1
VIDEO CAPTURE MACHINES
2
3
15
CALIBRATED CAMERAS
RENDERING SERVER
$\begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$
MIXED REALITY CLIENT
Camera In
HMD Out

FIGURE 3

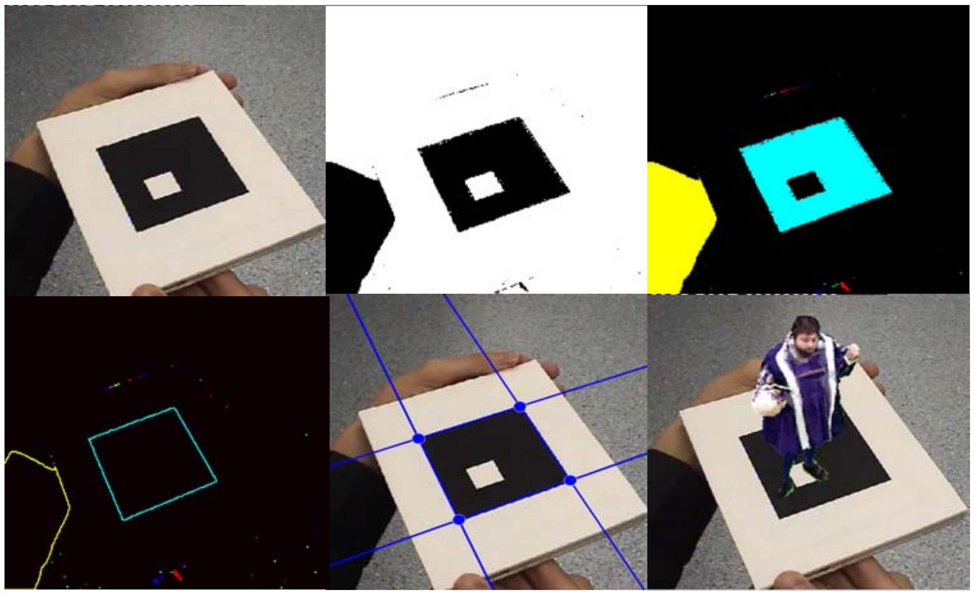


FIGURE 4

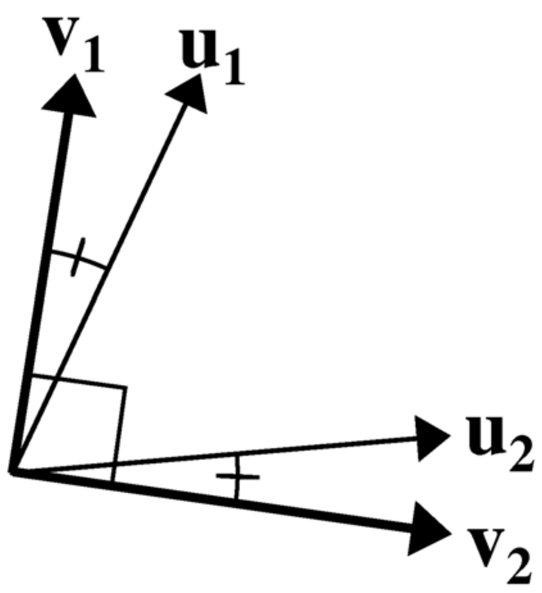
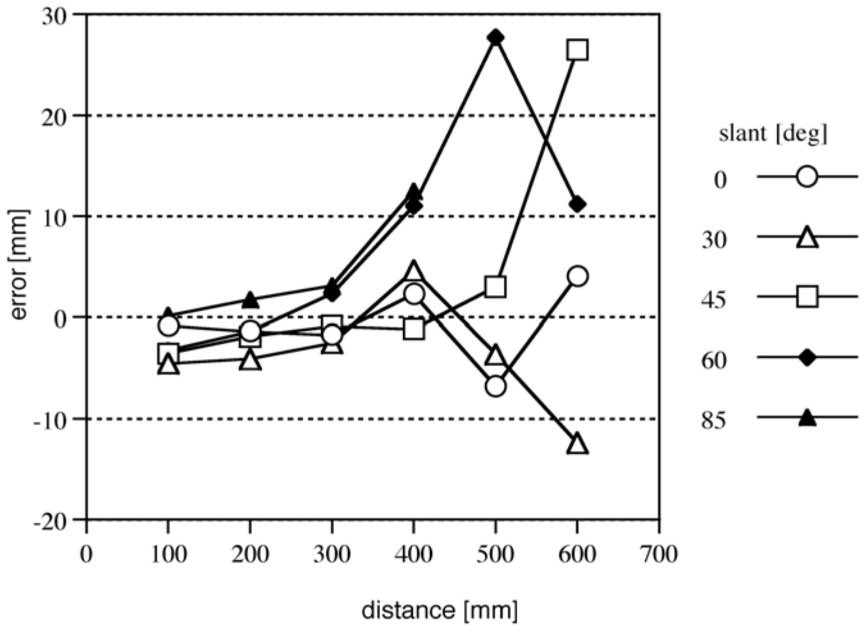


FIGURE 5



30
20
10
0
-10
-20
error [mm]
0
100
200
300
400
500
600
700
distance [mm]
slant [deg]
0
30
45
60
85

FIGURE 6

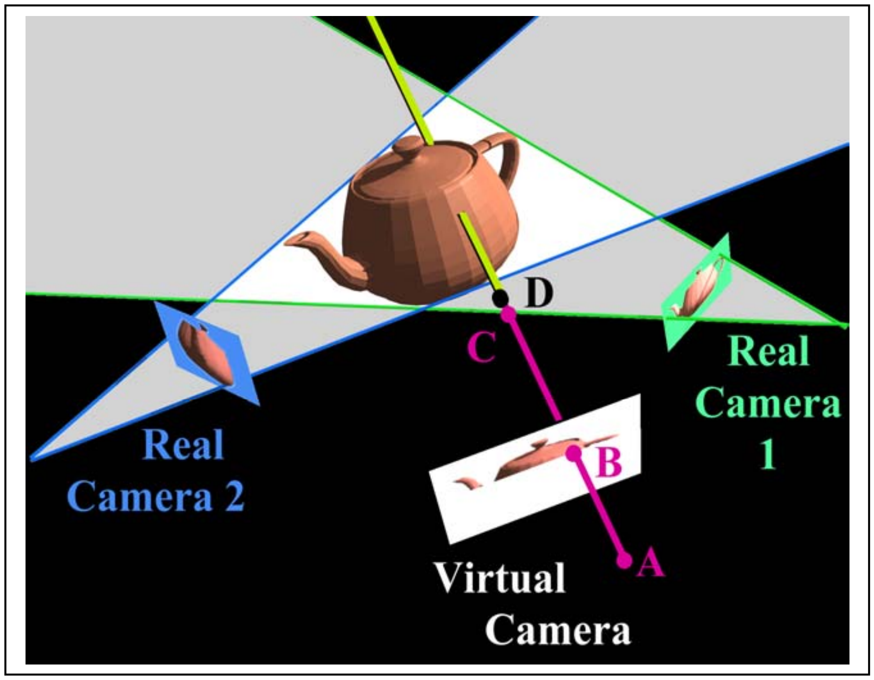


FIGURE 7

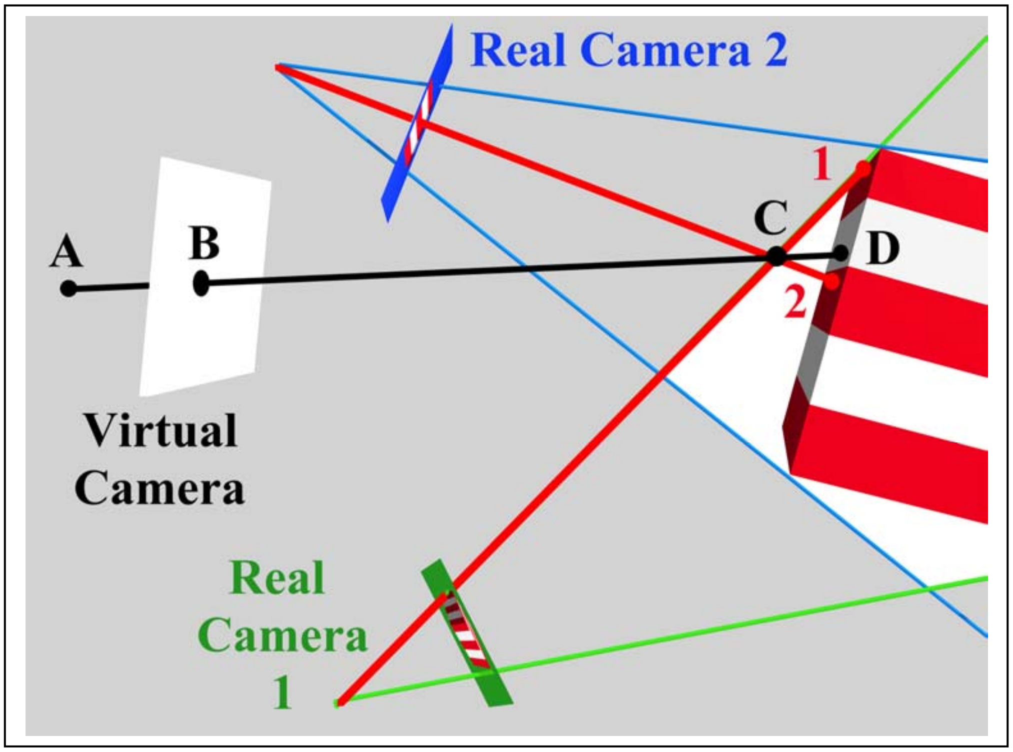


FIGURE 8

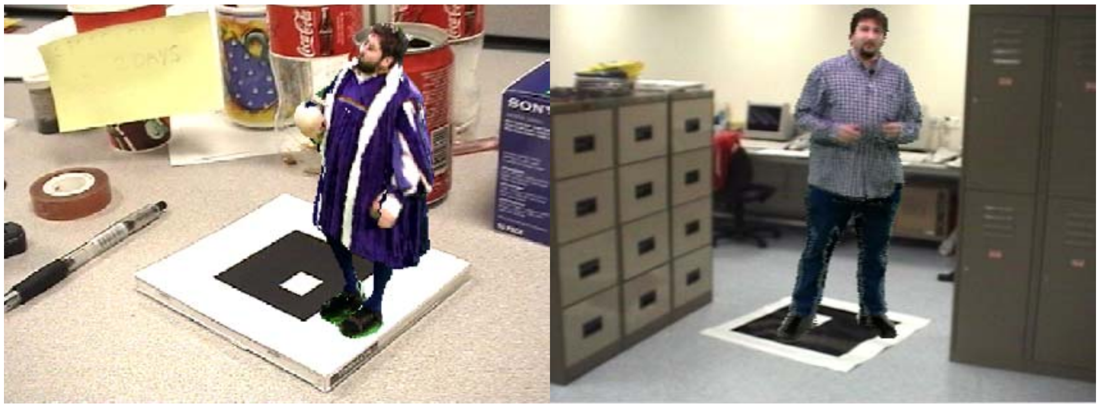


FIGURE 9

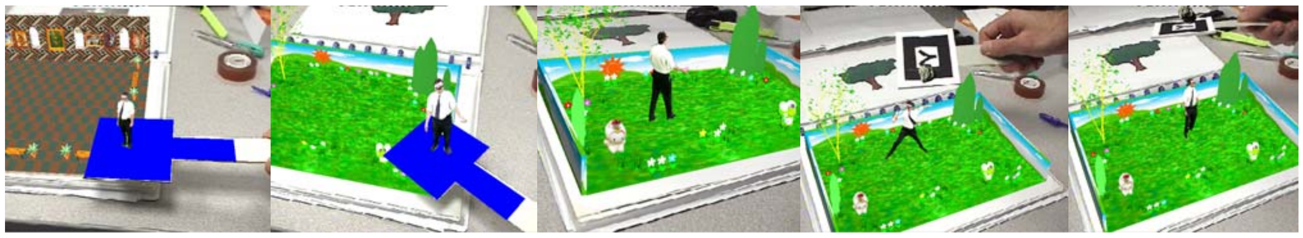


FIGURE 10