Image Mosaicing for Tele-Reality Applications

Richard Szeliski

Digital Equipment Corporation,
Cambridge Research Lab,
One Kendall Square, Bldg. 700,
Cambridge, MA 02139

Abstract

This paper presents some techniques for automatically deriving realistic 2-D scenes and 3-D geometric models from video sequences. These techniques can be used to build environments and 3-D models for virtual reality application based on recreating a true scene, i.e., *tele-reality* applications. The fundamental technique used in this paper is *image mosaicing*, i.e., the automatic alignment of multiple images into larger aggregates which are then used to represent portions of a 3-D scene. The paper first examines the easiest problems, those of flat scene and panoramic scene mosaicing. It then progresses to more complicated scenes with depth, and concludes with full 3-D models. The paper also discusses a number of novel applications based on tele-reality technology.

1 Introduction

Virtual reality is currently creating a lot of excitement and interest in the computer graphics community. Typical virtual reality systems use immersive technologies such as head-mounted or stereo displays and data gloves. The intent is to convince users that they are interacting with an alternate physical world, and also often to give insight into computer simulations, e.g., for fluid flow analysis or molecular modeling.

Rather than being based on simulations, a different class of virtual reality applications attempts to recreate true reality as convincingly as possible. Examples of such applications include flight simulators, interactive multi-player games, and medical simulators. Since the reality being recreated is usually distant, we use the term *tele-reality* in this paper to refer to such virtual reality scenarios based on real imagery.[1] Tele-reality applications which could be built using the techniques described in this paper include scanning a whiteboard in your office to obtain a high-resolution image, walkthroughs of existing buildings for re-modeling or selling, participating in a virtual classroom, and browsing through your local supermarket aisles from home.

While most focus in virtual reality today is on input/output devices and the quality and speed of rendering, perhaps the biggest bottleneck standing in the way of widespread tele-reality applications is the slow and tedious model-building process. This problem also affects many other areas of computer graphics, such as computer animation, special effects, and CAD. For small objects, say 0.2–2m, laser-based rangefinders, which provide registered depth and colored texture maps, are a good solution Wohlers94. Such rangefinders been used extensively in the entertainment industry for special effects and computer animation. Unfortunately, laser-based rangefinders are fairly expensive, limited in resolution, and most importantly, limited in range (e.g., they cannot be used to scan in a building).

The image-based ranging techniques we develop in this paper have the potential to overcome these limitations. Imagine walking through an environment such as a building interior and filming a video sequence of what you see. By registering and compositing the images in the video together into large mosaics of the scene, image-based ranging can achieve an essentially unlimited resolution.[2] Since the images can be acquired using any optical technology (from microscopy, through hand-held videocams, to satellite photography), the range or scale of the scene being reconstructed is not an issue. As desktop video becomes ubiquitous in our computing environments—initially for videoconferencing, and later as an advanced user interface tool—image-based scene and model acquisition will become accessible to all users.

In this paper, we develop novel techniques for extracting large 2-D textures and 3-D models from image sequences and also present some potential applications. After a review of related work (Section 2) and of the basic image formation equations (Section 3), we present our technique for registering pieces of a planar scene (Section 4). We then show how the same technique can be used to mosaic panoramic scenes obtained by rotating the camera around its center of projection (Section 5). Section 6 discusses how to recover depth in scenes. Section 7 discusses the most general and difficult problem, that of building full 3-D models from multiple images. Section 8 presents some novel applications of the tele-reality technology developed in this paper.

[1] The term *telepresence* is also often used, especially for applications such as tele-medicine where two-way interactivity.

[2] The traditional use of the term *image compositing* [1] is for blending images which are already registered. We use the term *image mosaicing* in this paper to avoid confusion with this previous work.

0-8186-6410-X/94 $4.00

2 Related work

The extraction of geometric information from multiple images has long been one of the central problems in computer vision and photogrammetry. However, many of the techniques used are based on feature extraction, produce sparse descriptions of shape, and often require manual assistance. Certain techniques, such as stereo, produce depth or elevation maps which are inadequate to model true 3-D objects. In computer vision, the recovered geometry is normally used either for object recognition or for grasping and navigation (robotics). Relatively little attention has been paid to building 3-D models registered with intensity (texture maps), which are necessary for computer graphics and virtual reality applications. However, some recent papers have looked at compositing multiple 2-D images to obtain higher resolution images [2, 3].

In computer graphics, compositing multiple image streams together to create larger format (*Omnimax*) images is discussed in [4]. However, in this application, the relative position of the cameras was known in advance. Compositing video into *salient still* based on affine transformations is discussed in [5, 6]. The registration techniques developed in this paper are related to *image warping* [7] since once the images are registered, they can be warped into a common reference frame before being composited. Recently, image warping based on z-buffer depths and camera motion has been applied to view interpolation as a quick alternative to full 3-D rendering [8]. The techniques we develop in Section 6 can be used to compute the depth maps necessary for this approach directly from real-world image sequences.

3 Basic imaging equations

Many of the ideas in this paper have their simplest expression using projective geometry. However, rather than relying on these results, we use standard methods and notations from computer graphics, and prove whatever simple results we require in [9]. Throughout, we use *homogeneous coordinates* to represent points, i.e., we denote 2-D points in the image plane as (x, y, w), with $(x/w, y/w)$ being the corresponding *Cartesian coordinates* [1]. Similarly, 3-D points with homogeneous coordinates (x, y, z, w) have Cartesian coordinates $(x/w, y/w, z/w)$.

Using homogeneous coordinates, we can describe the class of 2-D planar transformations using matrix multiplication

$$\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} = \begin{bmatrix} m_{00} & m_{01} & m_{02} \\ m_{10} & m_{11} & m_{12} \\ m_{20} & m_{21} & m_{22} \end{bmatrix} \begin{bmatrix} x \\ y \\ w \end{bmatrix} \quad (1)$$

or $\mathbf{u}' = \mathbf{M_{2D}}\mathbf{u}$.[3] The simplest transformations in this general class are pure translations, followed by translations and rotations (rigid transformations), plus scaling (similarity transformations), affine transformations, and full projective transformations. Figure 1 shows a square and possible rigid, affine, and projective deformations.

In 3-D, we have the same hierarchy of transformations, with rigid, similarity, affine, and full projective transformations. The $\mathbf{M_{3D}}$ matrices in this case are 4×4. Of particular interest are the rigid (Euclidean) transformation,

$$\mathbf{E} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0}^T & 1 \end{bmatrix}, \quad (2)$$

where $\mathbf{R}$ is a 3×3 orthonormal rotation matrix and $\mathbf{t}$ is a 3-D translation vector, and the 3×4 viewing matrix

$$\mathbf{V} = \begin{bmatrix} \hat{\mathbf{V}} & \mathbf{0} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1/f & 0 \end{bmatrix}, \quad (3)$$

which projects 3-D points through the origin onto a 2-D projection plane a distance f along the z axis [10]. The 3×3 $\hat{\mathbf{V}}$ matrix can be a general matrix in the case where the internal camera parameters are unknown, but the last column of $\mathbf{V}$ is always zero for central projection.

The combined equations projecting a 3-D world coordinate $\mathbf{p} = (x, y, z, w)$ onto a 2-D screen location $\mathbf{u} = (x', y', w')$ can thus be written as

$$\mathbf{u} = \mathbf{VEp} = \mathbf{M_{cam}p}, \quad (4)$$

where $\mathbf{M_{cam}}$ is a 3×4 *camera matrix*. This equation is valid even if the camera calibration parameters and/or the camera orientation are unknown.

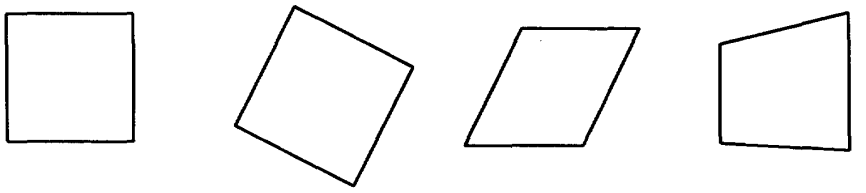
4 Planar image mosaicing

The simplest possible set of images to mosaic are pieces of a planar scene such as a document, whiteboard, or flat desktop. Imagine that we have a camera fixed directly over our desk. As we slide a document under the camera, different portions of the document are visible. Any two such pieces are related to each other by a translation and a rotation (2-D rigid transformation).

Now imagine that we are scanning a whiteboard with a hand-held video camera. The class of transformations relating two pieces of the board is the family of 2-D projective transformations (just imagine how a square or grid in one image can appear in another). These transformations can be computed without any knowledge of the internal camera calibration parameters such as focal length and optical center, or knowledge of the relative camera motion between frames. The fact that 2-D projective transformations capture all such possible mappings is a basic result of projective geometry (see [9] for a simple proof).

Given this knowledge, how do we compute the transformations relating the images? A variety of techniques are possible, some more automated than others. For example, we

[3] Two $\mathbf{M_{2D}}$ matrices are equivalent if they are scalar multiples of each other. We remove this redundancy by setting $m_{22} = 1$. or $\sum_j m_{2j}^2 = 1$.



original square rigid (rotation) affine deformation projective deformation

Figure 1: Rigid, affine, and projective transformations

could manually identify four or more corresponding points between the two views. We could also iteratively adjust the relative positions of input images using either a blink comparator or transparency. Unfortunately, these kinds of manual approaches are too tedious to be useful for large tele-reality applications.

The approach we use in this paper is to directly minimize the discrepancy in intensities between pairs of images after applying the transformation we are recovering. This has the advantage of not requiring any easily identifiable feature points, and of being statistically optimal once we are in the vicinity of the true solution [11]. Let us re-write our 2-D transformations as

$$x_i' = \frac{m_0 x_i + m_1 y_i + m_2}{m_6 x_i + m_7 y_i + 1}, \; y_i' = \frac{m_3 x_i + m_4 y_i + m_5}{m_6 x_i + m_7 y_i + 1}. \quad (5)$$

Our technique minimizes the sum of the squared intensity errors

$$E = \sum_i [I'(x_i', y_i') - I(x_i, y_i)]^2 = \sum_i e_i^2 \quad (6)$$

over all corresponding pairs of pixels i which are inside both images $I(x, y)$ and $I'(x', y')$ (pixels which are mapped outside image boundaries do not contribute). Once we have found the best transformation $\mathbf{M}_{2D}$, we can *warp* image I' into the reference frame of I using $\mathbf{M}_{2D}^{-1}$ and then blend the two images together [7]. To reduce visible artifacts, we weight images being blended together more heavily towards the center, using a bilinear weighting function.

To perform the minimization, we use the Levenberg-Marquardt iterative non-linear minimization algorithm [12]. This algorithm requires the computation of the partial derivatives of e_i with respect to the unknown motion parameters $\{m_0 \ldots m_7\}$. These are straightforward to compute, e.g.,

$$\frac{\partial e_i}{\partial m_0} = \frac{x_i}{D_i}\frac{\partial I'}{\partial x'}, \ldots, \frac{\partial e_i}{\partial m_7} = -\frac{y_i}{D_i}\left(x_i'\frac{\partial I'}{\partial x'} + y_i'\frac{\partial I'}{\partial y'}\right), \quad (7)$$

where D_i is the denominator in (5), and $(\partial I'/\partial x', \partial I'/\partial y')$ is the image intensity gradient of I' at (x_i', y_i'). From these partials, the Levenberg-Marquardt algorithm computes an approximate Hessian matrix $\mathbf{A}$ and the weighted gradient vector $\mathbf{b}$ with components

$$a_{kl} = \sum_i \frac{\partial e_i}{\partial m_k}\frac{\partial e_i}{\partial m_l}, \quad b_k = -\sum_i e_i \frac{\partial e_i}{\partial m_k}, \quad (8)$$

and then updates the motion parameter estimate $\mathbf{m}$ by an amount $\Delta\mathbf{m} = (\mathbf{A} + \lambda\mathbf{I})^{-1}\mathbf{b}$, where λ is a time-varying stabilization parameter [12]. The advantage of using Levenberg-Marquardt over straightforward gradient descent is that it converges in fewer iterations. For more details on the exact implementation, see [11].

Unfortunately, both gradient descent and Levenberg-Marquardt only find locally optimal solutions. If the motion between successive frames is large, we must use a different strategy to find the best registration. We have implemented two different techniques for dealing with this problem. The first technique, which is commonly used in computer vision, is *hierarchical matching*, which first registers smaller, subsampled versions of the images where the apparent motion is smaller [13, 14]. Motion estimates from these smaller coarser levels are then used to initialize motion estimates at finer levels, thereby avoiding the local minimum problem (see [11] for details). While this technique is not guaranteed to find the correct registration, we have observed empirically that it works well when the initial misregistration is only a few pixels.

For larger displacements, we use *phase correlation* [15]. This technique estimates the 2-D translation between a pair of images by taking 2-D Fourier transforms of each image, computing the phase difference at each frequency, performing an inverse Fourier transform, and searching for a peak in the magnitude image. In our experiments, this technique has proven to work remarkably well, providing good initial guesses for image pairs which overlap by as little as 50%.

To demonstrate the performance of our algorithm, we digitized an image sequence with a camera panning over a whiteboard. Figure 2 shows the final mosaic of the whiteboard, with the location of a constituent image shown as a white outline. This mosaic is 1300×2046 pixels, based on compositing 39 NTSC (640×480) resolution images.

To compute this mosaic, we developed an interactive image mosaicing tool which lets the user coarsely position successive frames relative to each other. The tool also has an *automatic mosaicing* option which computes the initial rough placement of each image with respect to the previous one using phase correlation. Our algorithm then refines the location of each image by minimizing (6) using the current mosaic as $I(x, y)$ and the input frame being adjusted as $I'(x', y')$. The images in Figure 2 were automatically composited with-

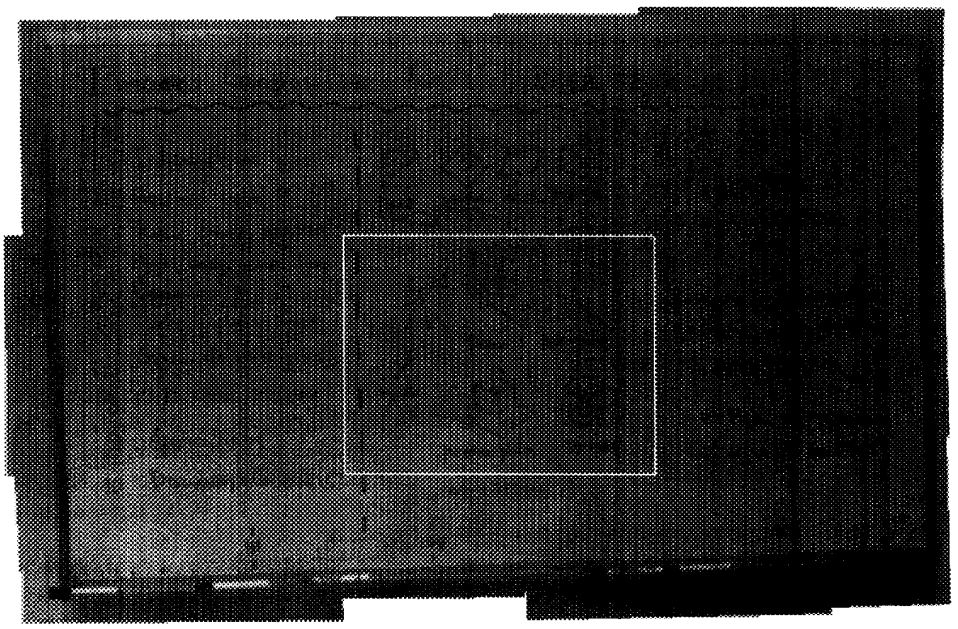


Figure 2: Whiteboard image mosaic example
The central square shows the size of one input image (*tile*).

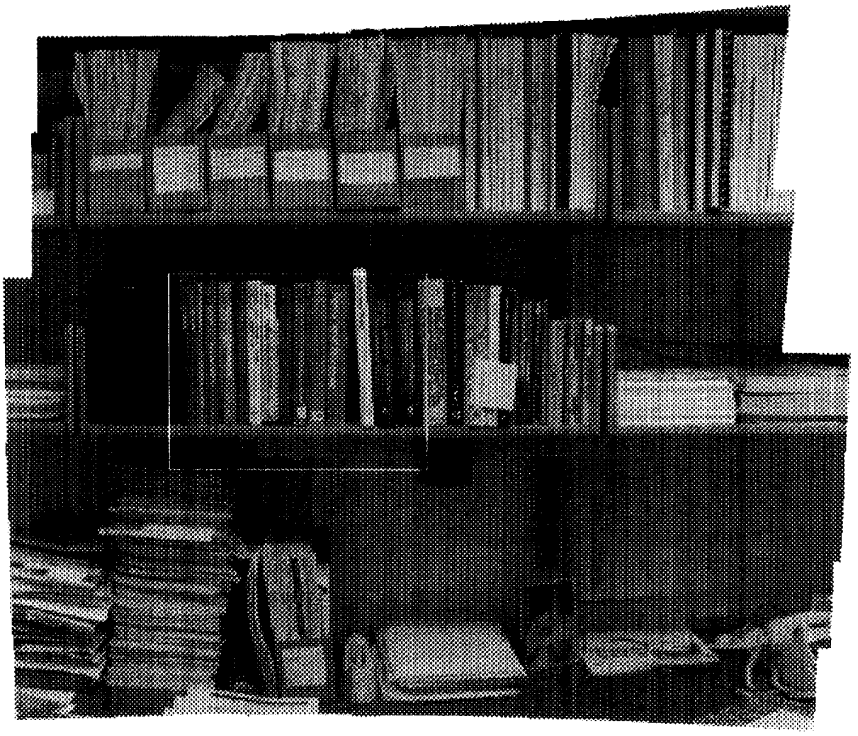


Figure 3: Panoramic image mosaic example (bookshelf and cluttered desk)
These images were pasted onto a planar viewing surface.

out user intervention using the middle frame (center of the image) as the *anchor* image (no deformations). As we can see, our technique works well on this example.

5 Panoramic image mosaicing

Another image mosaicing problem which turns out to be equally simple to solve is panoramic image mosaicing. In this scenario, we rotate a camera around its optical center in order to create a full "viewing sphere" of images. This is similar to the action of panoramic still photographic cameras where the rotation of a camera on top of a tripod is mechanically coordinated with the film transport. In our case, however, we can mosaic multiple 2-D images of arbitrary detail and resolution, and we need not know the camera motion. Examples of applications include constructing true scenic panoramas (say of the view at the rim of the Grand Canyon), or limited virtual environments (recreating a meeting room or office as seen from one location).

Images taken from the same viewpoint with a stationary optical center are related by 2-D projective transformations, just as in the planar scene case [9]. Intuitively, we cannot tell the relative depth of points in the scene as we rotate (there is no *motion parallax*), so they may as well be located on any plane; in projective geometry, we could say they lie on the *plane at infinity*, $w = 0$. For many applications, this is viewed as a deficiency, i.e., we cannot recover scene depth from rotational camera motion. For image mosaicing, this is actually an advantage since we just want to composite a large scene and be able to "look around" from a static viewpoint.

More formally, the 2-D transformation denoted by $\mathbf{M_{2D}}$ is related to the viewing matrices $\hat{\mathbf{V}}$ and $\hat{\mathbf{V}}'$ and the inter-view rotation $\mathbf{R}$ by [9]

$$\mathbf{M_{2D}} = \hat{\mathbf{V}}'\mathbf{R}\hat{\mathbf{V}}^{-1}. \tag{9}$$

In the case of a calibrated camera (known center of projection), this has a particularly simple form,

$$\mathbf{M_{2D}} = \begin{bmatrix} r_{00} & r_{01} & fr_{02} \\ r_{10} & r_{11} & fr_{12} \\ r_{20}/f' & r_{21}/f' & fr_{22}/f' \end{bmatrix}, \tag{10}$$

where f and f' are the scaled focal lengths in the two views, and r_{kl} are the entries in the rotation matrix $\mathbf{R}$. In this case, we only have to recover five independent parameters (or three if the f values are known) instead of the usual eight.

How do we represent a panoramic scene composited using our techniques? One approach is to divide the viewing sphere into several large, potentially overlapping regions, and to represent each region with a plane onto which we paste the images. Another approach is to compute the relative position of each frame relative to some base frame, and to then recompute an arbitrary view on the fly from all visible pieces, given a particular view direction $\mathbf{R}$ and zoom factor f. We have implemented both approaches and find that they produce similar results.

Figure 3 shows a mosaic of a bookshelf and cluttered desk, which is obviously a non-planar scene. The images were obtained by tilting and panning a video camera mounted on a tripod, without any special steps taken to ensure that the rotation was around the true center of projection. The complete scene is registered quite well. Figure 4 shows a complete circular panorama of an office, unrolled onto a cylindrical surface.

6 Scenes with arbitrary depth

Mosaicing flat scenes or panoramic scenes may be interesting for certain tele-reality applications, e.g., transmitting whiteboard or document contents, or viewing an outdoor panorama. However, for most applications, we must recover the depth associated with the scene to give the illusion of a 3-D environment, e.g., through nearby view synthesis [8]. Two possible approaches are to model the scene as piecewise-planar or to recover dense 3-D depth maps.

The first approach is to assume that the scene is piecewise-planar, as is the case with many man-made environments such as building exteriors and office interiors. The image mosaicing technique developed in Section 4 can then be applied to each of the planar regions in the image. Once the independent planar pieces have been composited, we could, in principle, recover the relative geometry of the various planes and the camera motion [9]. However, rather than pursuing this approach in this paper, we will purse the second, more general solution which is to recover a full depth map, i.e., to infer the missing z component associated with each pixel in a given image sequence.

When the camera motion is known, the problem of depth map recovery is called *stereo reconstruction* (or multi-frame stereo if more than two views are used). This problem has been extensively studied in photogrammetry and computer vision. When the camera motion is unknown, we have the more difficult *structure from motion* problem [10]. In this section, we present our solution to this latter problem based on recovering *projective depth*, which is particularly simple and robust and fits in well with the methods already developed in this paper.

To formulate the projective structure from motion recovery problem, we note that the coordinates corresponding to a pixel $\mathbf{u}$ with projective depth w in some other frame can be written as

$$\mathbf{u}' = \mathbf{V}'\mathbf{E}\mathbf{p} = \hat{\mathbf{V}}'\mathbf{R}\hat{\mathbf{V}}^{-1}\mathbf{u} + w\hat{\mathbf{V}}'\mathbf{t} = \mathbf{M_{2D}}\mathbf{u} + w\hat{\mathbf{t}}, \tag{11}$$

where $\mathbf{V}$, $\mathbf{E}$, $\mathbf{R}$, and $\mathbf{t}$ are defined in (2–3), and $\mathbf{M_{2D}}$ and $\hat{\mathbf{t}}$ are the computed planar projection matrix and epipole direction [9]. To recover the parameters in $\mathbf{M_{2D}}$ and $\hat{\mathbf{t}}$ for each frame along with the depth values w (which are the same for all frames), we use the same Levenberg-Marquardt algorithm as before.

Figure 4: Circular panoramic image mosaic example (office interior)
These images were pasted onto a cylindrical viewing surface.

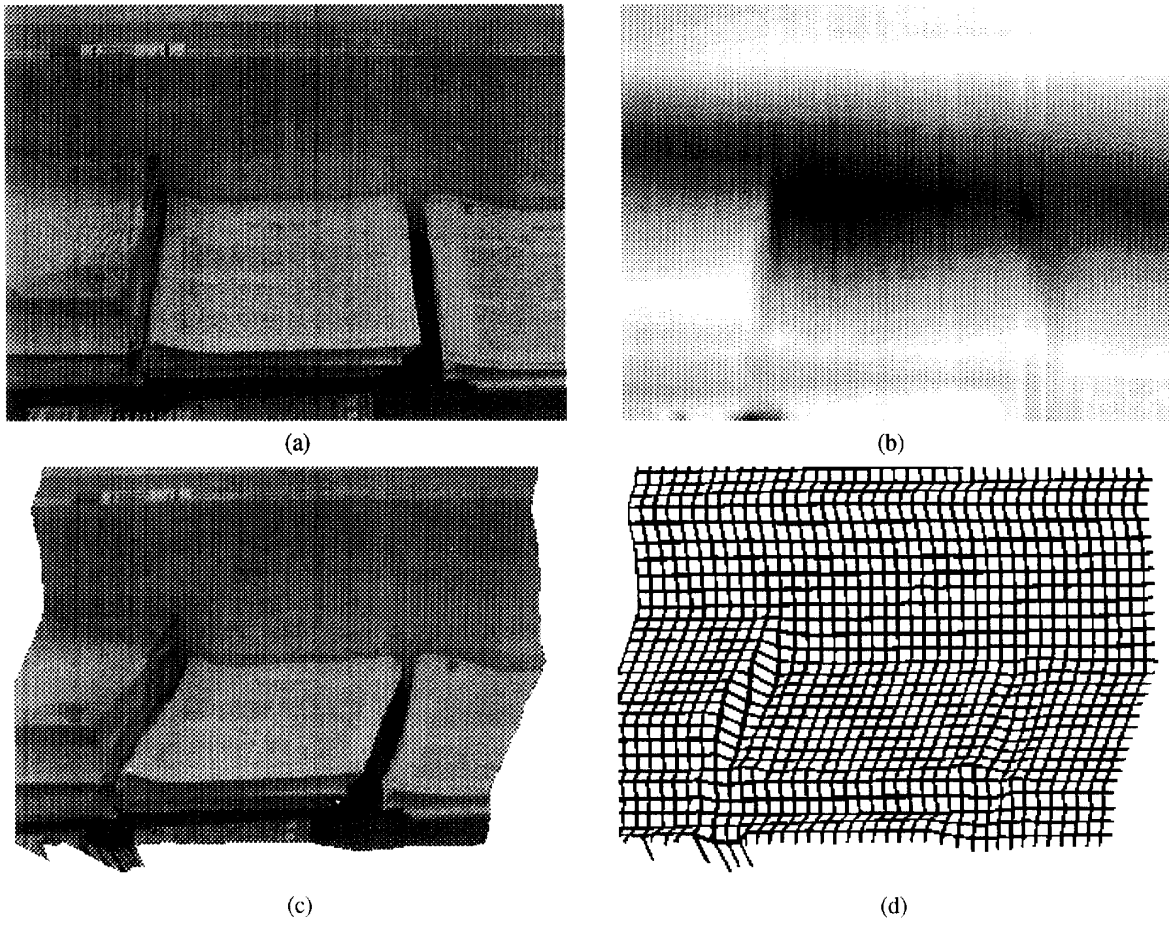


Figure 5: Depth recovery example: table with stacks of papers
(a) input image, (b) intensity-coded depth map (dark is farther back) (c) texture-mapped surface seen from novel viewpoint, (d) gridded surface.

In more detail, we write the projection equation as

$$x_i' = \frac{m_0 x_i + m_1 y_i + t_0 w_i + m_2}{m_6 x_i + m_7 y_i + t_2 w_i + 1},$$
$$y_i' = \frac{m_3 x_i + m_4 y_i + t_1 w_i + m_5}{m_6 x_i + m_7 y_i + t_2 w_i + 1}. \quad (12)$$

We compute the partial derivatives of x_i' and y_i' w.r.t. the m_k and t_k (which we concatenate into the motion vector $\mathbf{m}$) as before in (7). Similarly, we compute the partials of x_i' and y_i' with respect to w_i, i.e.,

$$\frac{\partial x_i'}{\partial w_i} = \frac{D_i t_0 - t_2}{D_i^2}, \quad \frac{\partial y_i'}{\partial w_i} = \frac{D_i t_0 - t_2}{D_i^2}, \quad (13)$$

where D_i is the denominator in (12).

To estimate the unknown parameters, we alternate iterations of the Levenberg-Marquardt algorithm over the motion parameters $\{m_0, \ldots, t_2\}$ in $\mathbf{m}$ and the depth parameters $\{w_i\}$, using the partial derivatives defined above to compute the approximate Hessian matrices $\mathbf{A}$ and the weighted error vectors $\mathbf{b}$ as in (8). In our current implementation, in order to reduce the total number of parameters being estimated, we represent the depth map using a tensor-product spline, and only recover the depth estimates at the spline control vertices (the complete depth map is available by interpolation) [11].

Figure 5 shows an example of using our projective depth recovery algorithm. The image sequence was taken by moving the camera up and over the scene of a table with stacks of papers (Figure 5a). The resulting depth-map is shown in Figure 5b as intensity-coded range values. Figure 5c shows the original intensity image texture mapped onto the surface seen from a side viewpoint which is not part of the original sequence (an example of view extrapolation). Figure 5d shows a set of grid lines overlayed on the recovered surface to better judge its shape. The shape is recovered reasonably well in areas where there is sufficient texture and the extrapolated views look reasonable.

7 Full 3-D model recovery

Recovering depth maps may be adequate for many tele-reality applications, e.g., scanning library bookshelves or supermarket aisles, but for other applications, i.e., when we need to see the back side of an object, full 3-D models are required. This is the most difficult image mosaicing problem, since not only do we have to recover depth, but we also have to merge (register and composite) multiple depth maps, and represent objects given no *a priori* knowledge about their rough shape or topology.

For simple topologies and shapes, deformable physically-based models can do a good job of recovering the unknown geometry [16]. For general topologies, the problem is more difficult, but techniques do exist for extracting 3-D shape from multiple views. For example, we can recover a volumetric description from the binary *silhouettes* of an object against its background [17], compute local optic flow (pixel motion) estimates and convert these into sparse 3-D point estimates [18], or track the occluding contours of an object to generate 3-D space curves [19]. A complete survey of 3-D shape extraction techniques is beyond the scope of this paper. Instead, we present the results of two of our previously developed algorithms applied to an image sequence of a cup rotating on a calibrated turntable (Figure 6a).

Our first technique converts each image into a binary silhouette by comparing the image with an empty background image. Each cube in the octree volumetric 3-D model is then projected (using the known camera position) into the silhouette, and cubes that fall outside the silhouette are culled. Cubes which fall partially into the silhouette are marked for later subdivision, and the process is repeated after each complete revolution at successively finer resolutions [17]. The resulting octree is shown in Figure 6b.

Our second technique extracts silhouette (extremal) edges from the image sequence, and uses a multi-frame stereo algorithm to reconstruct their 3-D position (internal albedo edges are also extracted and reconstructed) [19]. Figure 6c shows one of the edge images used by the algorithm, and Figure 6d shows the collection of 3-D curves estimated by the algorithm. Figure 6e shows the 3-D *epipolar curves* connecting these to form a mesh (only a portion of the cup is shown for clarity).

To produce a complete smooth 3-D surface, we must interpolate and smooth the geometry available with these techniques. We do this using the particle-based surface modeler developed in [20] to smooth out the rough geometry provided by the volumetric (octree) recovery technique. Once a detailed surface description is available, we then apply *inverse texture mapping* to associate a color with each point in our surface representation, by projecting the points into each input image, and computing the average color over all frames where a given point is visible. A view of the final texture-mapped particle-based model is shown in Figure 6f.

These examples demonstrate some of our algorithms for reconstructing an isolated object undergoing known motion. Similar techniques can be used to solve the more general 3-D scene recovery problem where the camera motion is unknown. Methods for determining the motion include the projective motion algorithm presented in the previous section, as well as techniques described in [10, 21] and elsewhere.

8 Applications

Given automated techniques for building 2-D and 3-D scenes from video sequences, what can we do with these models? In this section, we describe a number of potential applications, including whiteboard and document scanning, 3-D model acquisition for inverse CAD, model acquisition for computer animation and special effects, supermarket shopping at home, interactive walkthroughs of historical buildings, and live tele-reality (telepresence) applications.

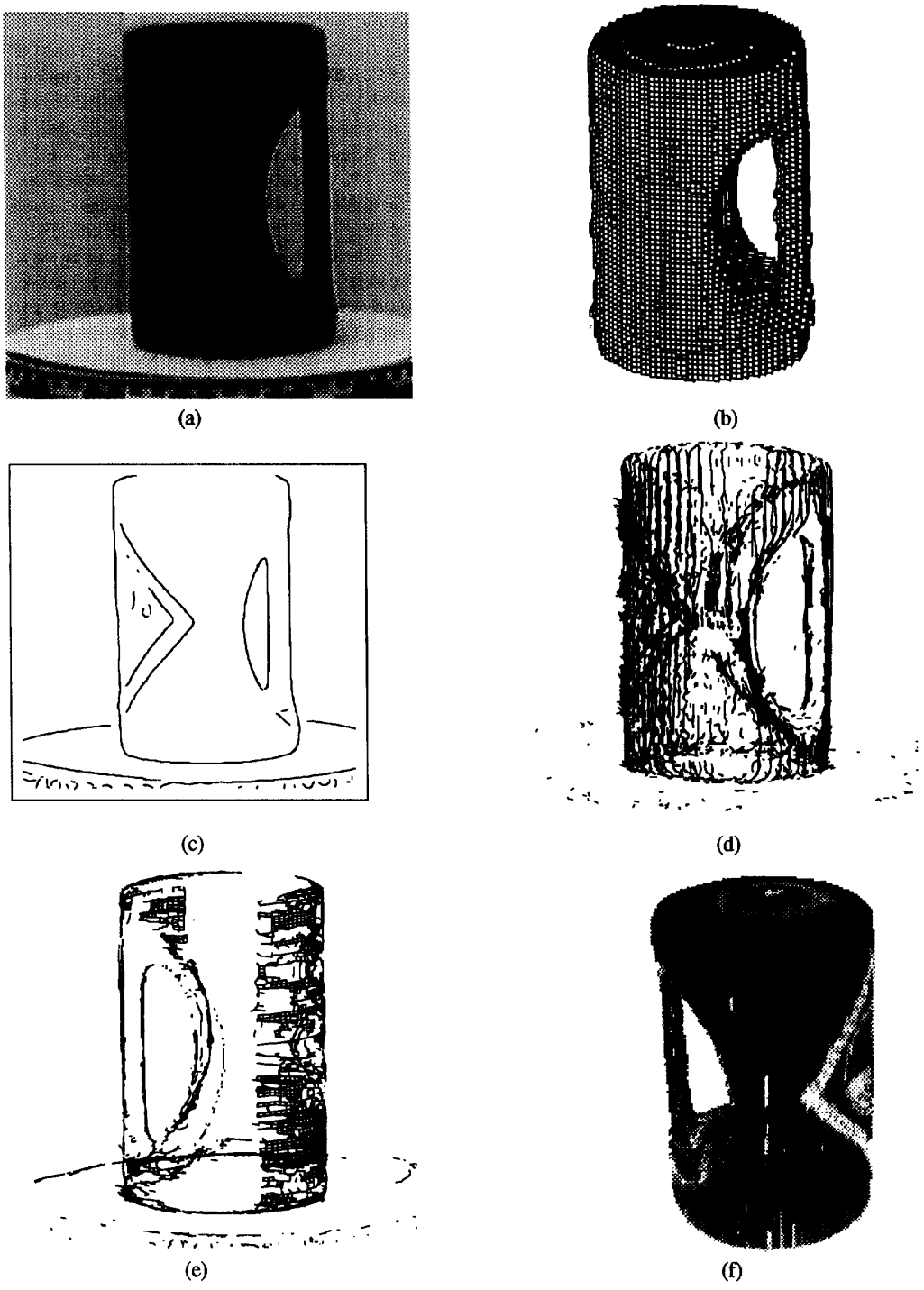


Figure 6: 3-D model recovery example
(a) input image, (b) octree recovered from silhouettes (c) 2-D edges, (d) 3-D extremal contours (side view), (e) portion of surface mesh made from profile and epipolar curves, (f) inverse texture-mapped 3-D model (oblique view),

The most straightforward application of image mosaicing is scanning whiteboards or blackboards as an aid to videoconferencing or as an easy way to capture ideas. Scanning can produce images of much greater resolution than single wide-angle lens shots. The techniques developed in this paper enable any video camera attached to a computer to be used.

Another obvious application of this technology is for document scanning. Hand-held scanners currently perform this function quite well. However, since they are based on linear CCDs, they are subject to "skewing" problems in each strip of the scanned image which must be corrected manually. Using 2-D video images as input removes some of these problems. A final global skew may still be needed to make the document square, but this can be performed automatically by detecting document edges or internal horizontal and vertical edges (e.g., column borders).

The 3-D model building technology, while more difficult to automate reliably, has even greater potential. For example, we could construct a *3-D fax* which would scan 3-D objects at one end using a video camera and either mechanical or user-controlled motion, and a 3-D graphics display at the other end [22]. This fax could be used to transmit 3-D models to a remote site for viewing and design revision, or to input rough CAD models for reverse engineering or clay mock-up applications.

Rapid 3-D model building is also critical in the computer animation and special effects industries. Currently, it can take days or weeks to build by hand each computer model used in a feature-length film. The ability to build such models rapidly from real objects or physical mock-ups would be of great utility, especially when freed from the range and resolution limitations of active rangefinding systems.

A more mass-market application is home shopping applied not to single objects but to complete stores, such as your local supermarket. This has the advantage of having a familiar look and organization, and enables the pre-planning of your next shopping trip. The images of the aisles (with their current contents) can be digitized by rolling a video camera through the store. More detailed geometric models of individual items can be acquired either by a 3-D model building process, or, in the future, directly from the manufacturer. The shopper can then stroll down the aisles, pick out individual items, and look at their ingredients and prices.

Walkthrough of existing building can have a number of applications. For example, interactive 3-D walkthroughs of your home, built by walking a video camera through the rooms and mosaicing the image sequences, could be used for selling your house (an extension of existing still-image based systems), or for re-modeling or renovations. Walkthroughs of historic building (e.g., palaces or museums) can be used for educational and entertainment purposes. A museum scenario might include the ability to look at individual 3-D objects such as sculptures, and to bring up related information in a hypertext system.

The ultimate in tele-reality systems is dynamic tele-reality (sometimes called *telepresence*), which composites video from multiple source in real-time to create the illusion of being in a dynamic (and perhaps reactive) 3-D environment. An example of such an application might be to view a 3-D version of a concert with control over the camera shots, even being able to see the concert from the musicians' point of view. Other examples might be to participate or consult in a surgery from a remote location (*tele-medicine*), or to remotely participate in a *virtual classroom*. Building such dynamic 3-D models at frame rates is beyond the processing power of today's high-performance superscalar workstations, but it could be achieved using a collection of such machines or special-purpose stereo hardware.

9 Discussion

Image mosaicing provides a powerful new way of creating the detailed 3-D models and scenes needed for tele-reality applications. By registering multiple images together, we can create scenes of extremely high resolution and at the same time recover partial or full 3-D geometric information. While we use techniques from computer vision to perform the registration, our focus is different: traditional vision techniques are designed for inspection, recognition, and robot control, while our techniques are designed to produce realistic 3-D models for computer graphics and virtual reality applications.

The approach we use, namely direct minimization of intensity differences between warped images, has a number of advantages over more traditional vision techniques, which are based on tracking features from frame to frame [10, 21]. Our techniques produce dense estimates of shape, work in highly textured areas where features may not be reliably observed, and make statistically optimal use of all the information [11]. Our approach is similar to [14], who also use intensity differences. However, we use full 2-D projective models of motion instead of instantaneous quadratic flow fields, and we also use a projective formulation of structure from motion, which eliminates the need for calibrated cameras. It is also very similar to [3], who also use planar projective motion estimates and have furthermore demonstrated superresolution results, i.e., the ability to obtain higher resolution images by simply jittering the camera rather than panning.

While our techniques have worked well in the scenes in which we have tried them, we must be cautious about their general applicability. The intensity-based techniques we use are sensitive to image intensity variation, such as those caused by video camera gain control and vignetting (darkening of the corners at wide lens apertures); working with band-pass filtered images can remove most of these problems. All vision techniques are also sensitive to geometric distortions (deviations from the pinhole model) in the optics, so careful calibration is necessary for optimal accuracy (the results in this paper

were obtained with uncalibrated cameras).

The depth extraction techniques we use rely on the presence of texture in the image. In areas of sufficient texture, it is still possible for the registration/matching algorithm to compute an erroneous depth estimate (such gross errors are less common in active rangefinders). Where texture is absent, interpolation must be used, and this can lead to erroneous (hallucinated) depth estimates. Other visual cues, such as occluding contours or shading, can be used to mitigate these problems in some cases, but a general-purpose reliable vision-based ranging system remains very much an open research problem.

10 Conclusions

In this paper, we have presented a hierarchy of scene models, ranging from 2-D planar and panoramic mosaics through full 3-D object models, and developed a collection of associated scene recovery algorithms. The creation of realistic high-resolution 3-D scenes from video imagery opens up many new applications for tele-reality technology. These include office applications such as whiteboard, document, and bookshelf scanning, simulated meeting spaces, engineering applications such as reverse engineering and collaborative design, and computer graphics applications such as 3-D model building. More importantly, this technology enables mass market applications such as home shopping, education (the virtual classroom), and entertainment (virtual travel). Ultimately, as processing speeds and reconstruction algorithms improve further, we will see dynamic real-time 3-D scene and model recovery being used to provide an even more exciting range of telepresence and tele-reality applications.

Acknowledgements

Bob Thomas introduced me to the term *tele-reality*, which he used to describe the live concert telepresence scenario. James Coughlan developed the image registration code. David Tonnesen, Demetri Terzopoulos, Sing Bing Kang, and Richard Weiss were collaborators on some of the 3-D reconstruction algorithms described in this paper.

References

[1] J. D. Foley, A. van Dam, S. K. Feiner, and J. F. Hughes. *Computer Graphics: Principles and Practice*. Addison-Wesley, Reading, MA, 2 edition, 1990.

[2] M. Irani and S. Peleg. Improving resolution by image registration. *Graphical Models and Image Processing*, 53(3):231–239, May 1991.

[3] S. Mann and R. W. Picard. Virtual bellows: constructing high-quality images from video. In *First IEEE International Conference on Image Processing (ICIP'94)*, November 1994.

[4] N. Greene and P. Heckbert. Creating raster Omnimax images from multiple perspective views using the elliptical weighted average filter. *IEEE Computer Graphics and Applications*, 6(6):21–27, June 1986.

[5] L. Teodosio and W. Bender. Salient video stills: Content and context preserved. In *ACM Multimedia 93*, pages 39–46, Anaheim, California, August 1993.

[6] L. Teodosio and W. Bender. Panoramic overview for navigating real-world scenes. In *ACM Multimedia 93*, pages 359–364, Anaheim, California, August 1993.

[7] G. Wolberg. *Digital Image Warping*. IEEE Computer Society Press, Los Alamitos, California, 1990.

[8] S. Chen and L. Williams. View interpolation for image synthesis. *Computer Graphics (SIGGRAPH'93)*, pages 279–288, August 1993.

[9] R. Szeliski. Image mosaicing for tele-reality applications. Technical Report 94/2, Digital Equipment Corporation, Cambridge Research Lab, June 1994.

[10] O. Faugeras. *Three-dimensional computer vision: a geometric viewpoint*. MIT Press, Cambridge, Massachusetts, 1993.

[11] R. Szeliski and J. Coughlan. Hierarchical spline-based image registration. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'94)*, pages 194–201, Seattle, Washington, June 1994. IEEE Computer Society.

[12] W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling. *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge University Press, Cambridge, England, second edition, 1992.

[13] L. H. Quam. Hierarchical warp stereo. In *Image Understanding Workshop*, pages 149–155, New Orleans, Louisiana, December 1984. Science Applications International Corporation.

[14] J. R. Bergen, P. Anandan, K. J. Hanna, and R. Hingorani. Hierarchical model-based motion estimation. In *Second European Conference on Computer Vision (ECCV'92)*, pages 237–252, Santa Margherita Liguere, Italy, May 1992. Springer-Verlag.

[15] C. D. Kuglin and D. C. Hines. The phase correlation image alignment method. In *IEEE 1975 Conference on Cybernetics and Society*, pages 163–165, New York, September 1975.

[16] D. Terzopoulos, A. Witkin, and M. Kass. Symmetry-seeking models and 3D object reconstruction. *International Journal of Computer Vision*, 1(3):211–221, October 1987.

[17] R. Szeliski. Rapid octree construction from image sequences. *CVGIP: Image Understanding*, 58(1):23–32, July 1993.

[18] R. Szeliski. Shape from rotation. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'91)*, pages 625–630, Maui, Hawaii, June 1991. IEEE Computer Society Press.

[19] R. Szeliski and R. Weiss. Robust shape recovery from occluding contours using a linear smoother. In *Image Understanding Workshop*, pages 939–948, Washington, D. C., April 1993. Morgan Kaufmann Publishers. A longer version is available as CRL TR 93/7.

[20] R. Szeliski and D. Tonnesen. Surface modeling with oriented particle systems. *Computer Graphics (SIGGRAPH'92)*, 26(2):185–194, July 1992.

[21] R. Szeliski and S. B. Kang. Recovering 3D shape and motion from image streams using nonlinear least squares. *Journal of Visual Communication and Image Representation*, 5(1):10–28, March 1994.

[22] I. Carlbom et al. Modeling and analysis of empirical data in collaborative environments. *Communications of the ACM*, 35(6):74–84, April 1992.