IEEE COMPUTER GRAPHICS AND APPLICATIONS (c) 1996 IEEE
Vol. 16, No. 2: MARCH 1996, pp. 22-30

Video Mosaics for Virtual Environments

Richard Szeliski, ***Microsoft Corporation***

By panning a camera over a scene and automatically compositing the video frames, this system creates large panoramic images of arbitrary shape and detail. Depth recovery from motion parallax also enables limited 3D rendering.

The use of photographic imagery as part of the computer graphics creation process is a well established and popular technique. Still imagery can be used in a variety of ways, including the manipulation and compositing of photographs inside video paint systems, and the texture mapping of still photographs onto 3D graphical models to achieve photorealism. Although laborious, it is also possible to merge 3D computer graphics seamlessly with video imagery to produce dramatic special effects. As computer-based video becomes ubiquitous with the expansion of transmission, storage, and manipulation capabilities, it will offer a rich source of imagery for computer graphics applications.

This article looks at one way to use video as a new source of high-resolution, photorealistic imagery for these applications. In its current broadcast-standard forms, video is a low-resolution medium that compares poorly with computer displays and scanned imagery. It also suffers, as do all input imaging devices, from a limited field of view. However, if you walked through an environment, such as a building interior, and filmed a video sequence of what you saw, you could subsequently register and composite the video images together into large mosaics of the scene. In this way, you can achieve an essentially unlimited resolution. Furthermore, since you can acquire the images using any optical technology (from microscopy to hand-held videocams to satellite photography), you can reconstruct any scene regardless of its range or scale.

Video mosaics can be used in many different applications, including the creation of virtual reality environments, computer-game settings, and movie special effects. Such applications commonly use an *environment map*—that is, a 360-degree spherical image of the environment—both to serve as a backdrop and to correctly generate reflections from shiny objects.[1]

In this article, I present algorithms that align images and composite scenes of increasing complexity—beginning with simple planar scenes and progressing to panoramic scenes and, finally, to scenes with depth variation. I begin with a review of basic imaging equations and conclude with some novel applications of the virtual environments created using the algorithms presented.

Basic imaging equations

The techniques developed here are all based on the ability to align different pieces of a scene (tiles) into a larger picture of the scene (mosaic) and then to seamlessly blend the images together. In many ways, this resembles current image morphing techniques,[2] which use a combination of image warping [3] and image blending.[4] To automatically construpt virtual environments, however, we must automatically derive the alignment (warping) transformations directly from the images, rather than relying on manual intervention.

Before proceeding, we need to consider the geometric transformations that relate the images to the mosaic. To do this, we use *homogeneous coordinates* to represent points, that is, we denote 2D points in the image plane as (*x*, *y*, *w*). The corresponding Cartesian coordinates are (*x/w*, *y/w*).[4] Similarly, 3D 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 2D planar projective transformations using matrix multiplication:

$$\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} = \begin{bmatrix} m_0 & m_1 & m_2 \\ m_3 & m_4 & m_5 \\ m_6 & m_7 & m_8 \end{bmatrix} \begin{bmatrix} x \\ y \\ w \end{bmatrix} \quad \text{or} \quad \mathbf{u}' = \mathbf{M}\mathbf{u} \tag{1}$$

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. Forms for the rigid and affine transformation matrix **M** are

$$\mathbf{M}_{\text{rigid}-2\text{D}} = \begin{bmatrix} \cos\theta & -\sin\theta & t_x \\ \sin\theta & \cos\theta & t_y \\ 0 & 0 & 1 \end{bmatrix},$$

$$\mathbf{M}_{\text{affine}-2\text{D}} = \begin{bmatrix} m_0 & m_1 & m_2 \\ m_3 & m_4 & m_5 \\ 0 & 0 & 1 \end{bmatrix}$$

with 3 and 6 degrees of freedom, respectively, while projective transformations have a general **M** matrix with 8 degrees of freedom. (Note that two **M** matrices are equivalent if they are scalar multiples of each other. We remove this redundancy by setting $m_8 = 1$.)

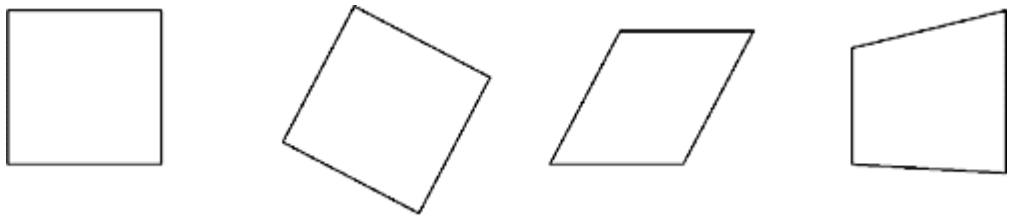


Figure 1. Square and rigid, affine, and projective transformations.

The same hierarchy of transformations exists in 3D, with rigid, similarity, affine, and full projective transformations having 6, 7, 12, and 15 degrees of freedom, respectively. The **M** matrices in this case are 4 × 4. Of particular interest are the rigid (Euclidean) transformation

$$\mathbf{E} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ 0^T & 1 \end{bmatrix} \tag{2}$$

where **R** is a 3 × 3 orthonormal rotation matrix and **t** is a 3D translation vector, and the 3 × 4 viewing matrix

$$\hat{\mathbf{V}} = [\mathbf{V} \ \mathbf{0}] = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1/f & 0 \end{bmatrix} \tag{3}$$

which projects 3D points through the origin onto a 2D projection plane a distance f along the z axis.[4]

(Note that a more general camera model, where **V** is an upper triangular matrix, can also account for aspect ratio, an offset optical center, and skew. A real camera might also have optical distortions that do not follow the pinhole model.)

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

$$\mathbf{u} = \hat{\mathbf{V}}\mathbf{E}\mathbf{p} = \mathbf{P}\mathbf{p}$$

where **P** is a 3 × 4 camera matrix. This equation is valid even if the camera calibration parameters and/or the camera orientation are unknown.

Planar image mosaics

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

Now imagine scanning a whiteboard with a hand-held video camera that you can move to any position. The class of transformations relating two pieces of the board, in this case, is the full family of 2D 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 of the relative camera motion between frames. The fact that 2D projective transformations capture all such possible mappings (at least for an ideal pinhole camera) is a basic result of projective geometry (see sidebar).

Given this knowledge, how do we compute the transformations relating the various scene pieces so that we can paste them together? A variety of techniques are possible, some more automated than others. For example, we could manually identify four or more corresponding points between the two views, which is enough information to solve for the eight unknowns in the 2D projective transformation. We could also iteratively adjust the relative positions of input images using either a blink comparator (alternating between the two images at a high rate) or transparency. Unfortunately, these kinds of manual approaches are too tedious to be useful for large compositing applications.

Local image registration

The approach used here directly minimizes the discrepancy in intensities between pairs of images after applying the recovered transformation. This has the advantages of not requiring any easily identifiable feature points and of being statistically optimal, that is, giving the maximum likelihood estimate once we are in the vicinity of the true solution. Let's rewrite our 2D transformations as

$$x' = \frac{m_0 x + m_1 y + m_2}{m_6 x + m_7 y + 1},$$

$$y' = \frac{m_3 x + m_4 y + m_5}{m_6 x + m_7 y + 1}$$

(5)

Our technique minimizes the sum of the squared intensity errors

$$E = \sum \left[I'(x', y') - I(x, y) \right]^2 = \sum e^2$$

(6)

over all corresponding pairs of pixels i inside both images $I(x, y)$ and $I'(x', y')$. (Pixels that are mapped outside image boundaries do not contribute.) Since (x', y') generally do not fall on integer pixel coordinates, we use bilinear interpolation of the intensities in I' to perform the resampling.

To perform the minimization, we use the Levenberg-Marquardt iterative nonlinear minimization algorithm.[5] This algorithm requires 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. For example,

$$\frac{\partial e}{\partial m_0} = \frac{x}{D} \frac{\partial I'}{\partial x'}, \ \cdots, \ \frac{\partial e}{\partial m_7} = -\frac{y}{D} \left(x' \frac{\partial I'}{\partial x'} + y' \frac{\partial I'}{\partial y'} \right)$$

(7)

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

$$a_{kl} = \sum \frac{\partial e}{\partial m_k} \frac{\partial e}{\partial m_l}, \quad b_k = -\sum e \frac{\partial e}{\partial m_k}$$

(8)

and then updates the motion parameter estimate **m** by an amount $\mathbf{\Delta m} = (\mathbf{A} + \lambda\mathbf{I})^{-1}\mathbf{b}$, where λ is a time-varying stabilization parameter.[5] The advantage of using Levenberg-Marquardt over straightforward gradient descent is that it converges in fewer iterations.

The complete registration algorithm thus consists of the following steps:

1. For each pixel i at location (x_i, y_i),

   (a) compute its corresponding position in the other image (x', y') using Equation 5;

   (b) compute the error in intensity between the corresponding pixels $e = I'(x', y') - I(x, y)$ (Equation 6) and the intensity gradient ($\partial I'/\partial x', \partial I'/\partial y'$) using bilinear intensity interpolation on I';

   (c) compute the partial derivative of e_i with respect to the m_k using

$$\frac{\partial e}{\partial m_k} = \frac{\partial I'}{\partial x'}\frac{\partial x'}{\partial m_k} + \frac{\partial I'}{\partial y'}\frac{\partial y'}{\partial m_k}$$

   as in Equation 7;

   (d) add the pixel's contribution to **A** and **b** as in Equation 8.

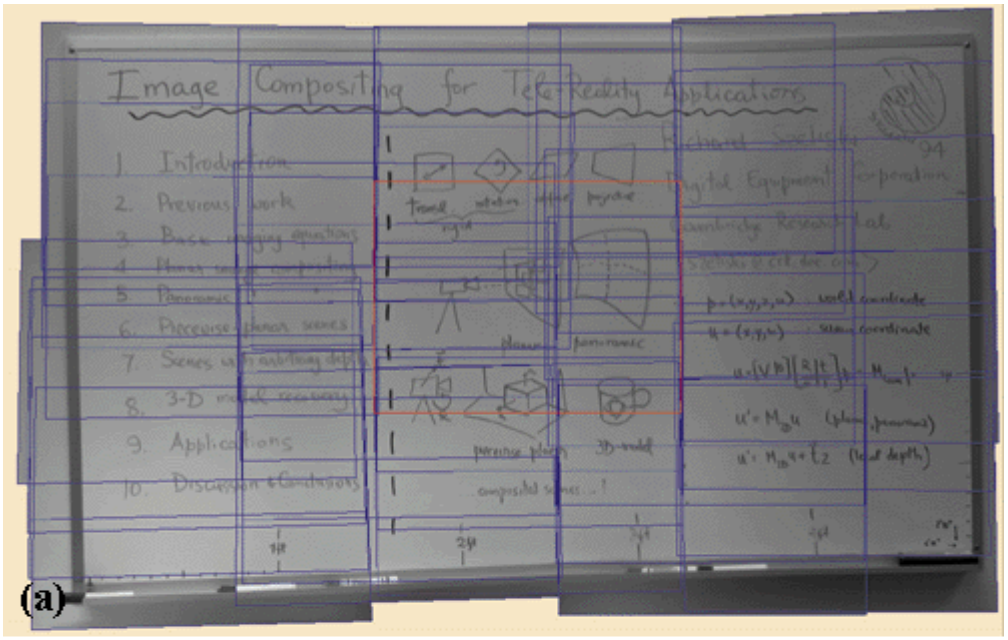
2. Solve the system of equations $(\mathbf{A} + \lambda\mathbf{I})\mathbf{\Delta m} = \mathbf{b}$ and update the motion estimate $\mathbf{m}^{(t+1)} = \mathbf{m}^{(t)} + \mathbf{\Delta m}$.
3. Check that the error in Equation 6 has decreased; if not, increment λ (as described in Press et al.[5]) and compute a new $\mathbf{\Delta m}$.
4. Continue iterating until the error is below a threshold or a fixed number of steps has been completed.

The steps in this algorithm are similar to the operations performed when warping images,[2,3] with additional operations for correcting the current warping parameters based on local intensity error and its gradients. For more details on the exact implementation, see Szeliski and Coughlan.[6]

Once we have found the best transformation **M**, we can blend the resampled image $I'(x', y')$ together with the reference image $I(x_i, y_i)$. To reduce visible artifacts—that is, to hide the edges of the component images—we use a weighted average with pixels near the center of each image contributing more to the final composite. The weighting function is a simple bilinear function:

$$w(x', y') = w_t(x')w_t(y') \tag{9}$$

where w_t is a triangle (hat) function that goes to zero at both edges of the image. In practice, this approach completely eliminates edge artifacts (see Figure 2), although a low-frequency "mottling" might still remain if the individual tiles have different exposures.



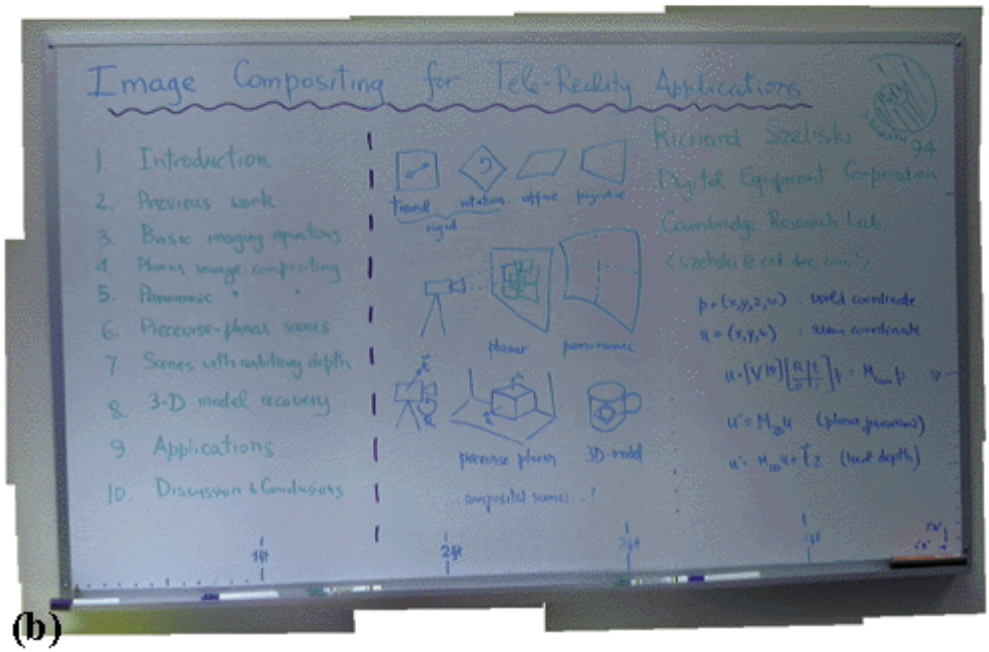


Figure 2. Whiteboard image mosaic example: (a) mosaic with component locations shown as colored outlines, (b) complete color mosaic (the central square shows the size of one input tile).

Global image registration

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. Two different techniques can be used to handle 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. Motion estimates from these smaller, coarser levels are then used to initialize motion estimates at finer levels, thereby avoiding the local minimum problem (see Szeliski and Coughlan[6] for details). While this technique is not guaranteed to find the correct registration, it has proved empirically to work well when the initial misregistration is only a few pixels (the exact domain of convergence depends on the intensity pattern in the image).

For larger displacements, you can use *phase correlation*.[7] This technique estimates the 2D translation between a pair of images by taking 2D 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. I have found this technique to work remarkably well in experiments, providing good initial guesses for image pairs that overlap by as little as 50 percent, even when there are moderate projective distortions (such as those that occur when using wide-angle lenses). The technique will not work if the interframe motion has large rotations or zooms, but this does not often occur in practice.

Results

To demonstrate the performance of the algorithm developed above, I digitized an image sequence with a camera panning over a whiteboard. Figure 2a shows the final mosaic of the whiteboard with the constituent images outlined in color. Figure 2b shows the final mosaic with the location of a single image shown as a white outline. This mosaic is 1,300 × 2,046 pixels, based on compositing 39 NTSC (640 × 480) resolution images.

To compute this mosaic, I developed an interactive image-manipulation tool that lets the user coarsely position successive frames relative to each other. The tool includes an automatic registration option that uses phase correlation to compute the initial rough placement of each image with respect to the previous one. The algorithm then refines the location of each image by minimizing Equation 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 without user intervention by employing the middle frame (center of the image) as the *base* image (no deformation). As you can see, the technique works well on this example.

Panoramic image mosaics

To build a panoramic image mosaic or *environment map*,[1] you can rotate a camera around its optical center. This resembles the action of panoramic still photographic cameras where the rotation of a camera on a tripod is mechanically coordinated with the film transport.[8] In our case, however, we can mosaic multiple 2D 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 Bryce Canyon) or limited virtual environments (a recreated meeting room or office as seen from one location).

Images taken from the same viewpoint with a stationary optical center are related by 2D projective transformations, just as in the planar scene case. (The sidebar presents a quick proof.) Because there is no motion parallax, you cannot see the relative depth of points in the scene as you rotate, so the images might as well be located on any plane.

More formally, the 2D transformation denoted by **M** is related to the viewing matrices **V** and **V**' and the inter-view rotation **R** by

$$\mathbf{M} = \mathbf{V}'\mathbf{R}\mathbf{V}^{-1} \tag{10}$$

(see sidebar). For a calibrated camera, we only have to recover the three independent rotation parameters (or five parameters if the focal length f values are unknown) instead of the usual eight.

How do we represent a panoramic scene composited using these 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.[1] Figure 3 shows a mosaic of a bookshelf and cluttered desk composited onto a single plane (the highlighted central square forms the base relative to which all other images are registered). The images were obtained by tilting and panning a video camera mounted on a tripod, without taking any special steps to ensure that the rotation was around the true center of projection. As you can see, the complete scene is registered quite well.

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

Another approach is to compute the relative position of each frame relative to some base frame and to periodically choose a new base frame for doing the alignment. (Note that the algebraic properties of the 2D projective transformation group—that is, the associativity of matrix multiplication—make it possible to always compute the transformation between any two frames. However, to represent arbitrary views (including 90-degree rotations) requires replacing the condition $m_8 = 1$ in Equation 1 with $m_6^2 + m_7^2 + m_8^2 = 1$.) We can then recompute an arbitrary view on the fly from all visible pieces, given a particular view direction **R** and zoom factor f. This is the approach used to composite the large wide-angle mosaic of Bryce Canyon shown in Figure 4.

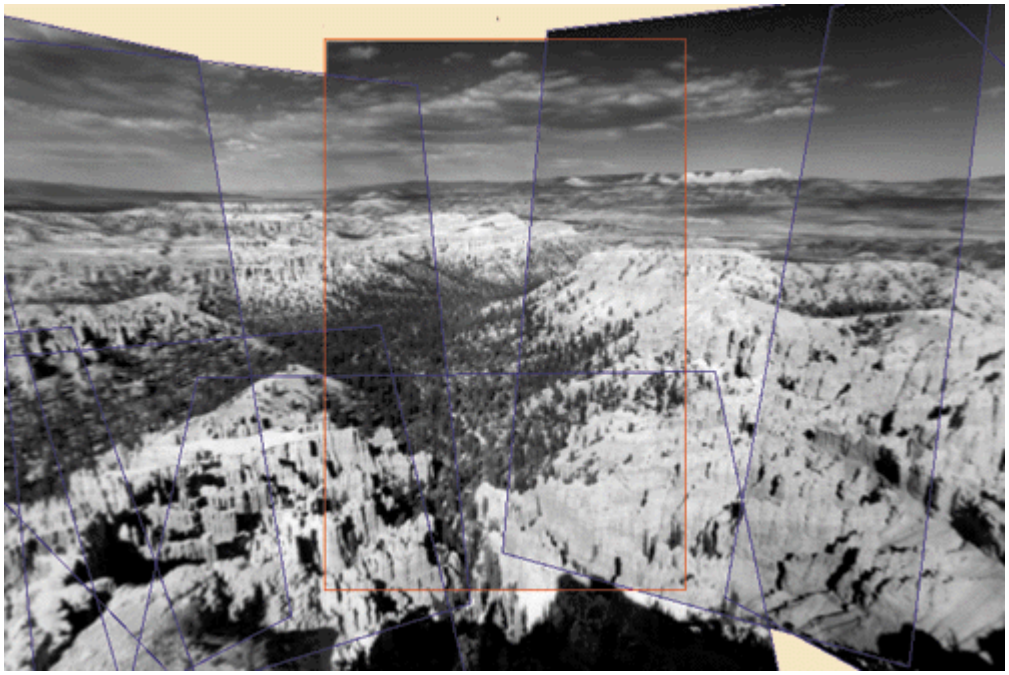


Figure 4. A portion of the Bryce Canyon mosaic. Because of the large motions involved, a single plane cannot represent the whole mosaic. Instead, different tiles are selected as base images.

A third approach is to use a cylindrical viewing surface to represent the image mosaic.[9-12] In this approach, we map world coordinates $\mathbf{p} = (x, y, z, w)$ onto 2D cylindrical screen locations $\mathbf{u} = (\theta, v)$, $\theta \in (-\pi, \pi]$ using

$$\theta = \tan^{-1}(x/z), \quad v = y/\sqrt{x^2 + z^2} \tag{11}$$

Figure 5 shows a complete circular panorama of an office unrolled onto a cylindrical surface. To build this panorama, each image is first mapped into cylindrical coordinates (using a known focal length and assuming the camera was horizontal). Then, the complete sequence is registered and composited using pure translations. The focal length of the camera can, if necessary, be recovered from images registered on a planar viewing surface. Figure 6 shows a similar panorama taken on the banks of the Charles River in Cambridge.

Figure 5. Circular panoramic image mosaic example (office interior). A total of 36 images

are pasted onto a cylindrical viewing surface.

Figure 6. Circular panoramic image mosaic example (exterior scene). A total of 29 images are pasted onto a cylindrical viewing surface.

In addition to constructing large, single-resolution mosaics, we can also build mosaics with spatially varying amounts of resolution, for example, to zoom in on areas of interest. The modifications to the algorithm described so far are relatively straightforward and affect only the image-blending portion of it. As more images are added at varying resolutions, we can use the last image already registered as the new base image (since it is likely to be close in size to the new image). To create the new composite mosaic, we can use a generalization of the pyramidal parametrics used in texture mapping.[13]

Projective depth recovery

While mosaics of flat or panoramic scenes are useful for many virtual reality and office applications, such as scanning whiteboards or viewing outdoor panoramas, some applications need the depth associated with the scene to give the illusion of 3D. Once the depth has been recovered, nearby views can be generated using view interpolation.[14] Two possible approaches are to model the scene as piecewise-planar or to recover dense 3D depth maps.

The first approach assumes that the scene is piecewise-planar, as is the case with many constructed environments such as building exteriors and office interiors. The mosaicing technique developed above for planar images can then be applied to each of the planar regions in the image. The segmentation of each image into its planar components can be done either interactively (for example, by drawing the polygonal outline of each region to be registered) or automatically by associating each pixel with one of several global motion hypotheses. Once the independent planar pieces have been composited, we could, in principle, recover the relative geometry of the various planes and the camera motion. However, rather than pursuing this approach here, we will develop the second, more general solution, which is to recover a full depth map. That is, we will 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 multiframe stereo if more than two views are used). This problem has been extensively studied in photogrammetry and computer vision.[15] When the camera

motion is unknown, we have the more difficult *structure-from-motion* problem.[15] This section presents a solution to the latter problem based on recovering *projective depth*. The solution is simple and robust, and fits in well with the methods already developed in this article.

Formulation

To formulate the projective structure-from-motion recovery problem, note that the coordinates corresponding to a pixel **u** with projective depth *w* in some other frame can be written as

$$\mathbf{u}' = \hat{\mathbf{V}}'\mathbf{E}\mathbf{p} = \mathbf{V}'\mathbf{R}^{-1}\mathbf{u} + w\,\mathbf{V}'\mathbf{t} = \mathbf{M}\mathbf{u} + w\tilde{\mathbf{t}} \tag{12}$$

where $\hat{\mathbf{V}}$, **E**, **R**, and **t** are defined in Equations 2 and 3, and **M** and $\tilde{\mathbf{t}}$ are the computed planar projective motion matrix and the epipole, that is, where the center of projection appears in the other camera (see Equation 24 in the sidebar). To recover the parameters in **M** and $\tilde{\mathbf{t}}$ for each frame together with the depth values *w* (which are the same for all frames), we can use the same Levenberg-Marquardt algorithm as before.[5] Once the projective depth values are recovered, they can be used directly in viewpoint interpolation (using new **M** and $\tilde{\mathbf{t}}$ matrices), or they can be converted to true Euclidean depth using at least four known depth measurements.[15]

In more detail, we write the projection equation as

$$x' = \frac{m_0 x + m_1 y + t_0 w + m_2}{m_6 x + m_7 y + t_2 w + 1}, \qquad y' = \frac{m_3 x + m_4 y + t_1 w + m_5}{m_6 x + m_7 y + t_2 w + 1} \tag{13}$$

with $\tilde{\mathbf{t}} = (t_0, t_1, t_2)$. We compute the partial derivatives of x', y', and e_i with respect to the m_k and t_k (which we concatenate into the motion vector **m**) as before in Equation 7. Similarly, we compute the partials of x' and y' with respect to w_i. That is,

$$\frac{\partial x'}{\partial w} = \frac{t_0 - x' t_2}{D}, \qquad \frac{\partial y'}{\partial w} = \frac{t_1 - y' t_2}{D}$$

where D_i is the denominator in Equation 13.

To estimate the unknown parameters, we alternate iterations of the Levenberg-Marquardt algorithm over the motion parameters $\{m_0, ..., t_2\}$ and the depth parameters $\{w_i\}$, using the partial derivatives defined above to compute the approximate Hessian matrices **A** and the weighted error vectors **b** as in Equation 8.

In the current implementation of this technique, the total number of parameters being estimated is decreased by using a tensor-product spline to represent the depth map and only recovering the depth estimates at the spline control vertices.[6] (The complete depth map is computed by interpolation.)

Results

Figure 7 shows an example of using the projective depth recovery algorithm. The image sequence was taken by moving the camera up and over a table with stacks of papers (Figure 7a). The resulting depth map is shown in Figure 7b as intensity-coded range values. Figure 7c shows the original color image texture mapped onto the surface seen from a side viewpoint that is not part of the original sequence (an example of view extrapolation). Figure 7d shows a set of grid lines overlaid on the recovered surface to better judge its shape. The shape is recovered reasonably well in areas where there is sufficient texture to give depth cues (uniform intensity areas give none), and the extrapolated views look reasonable.

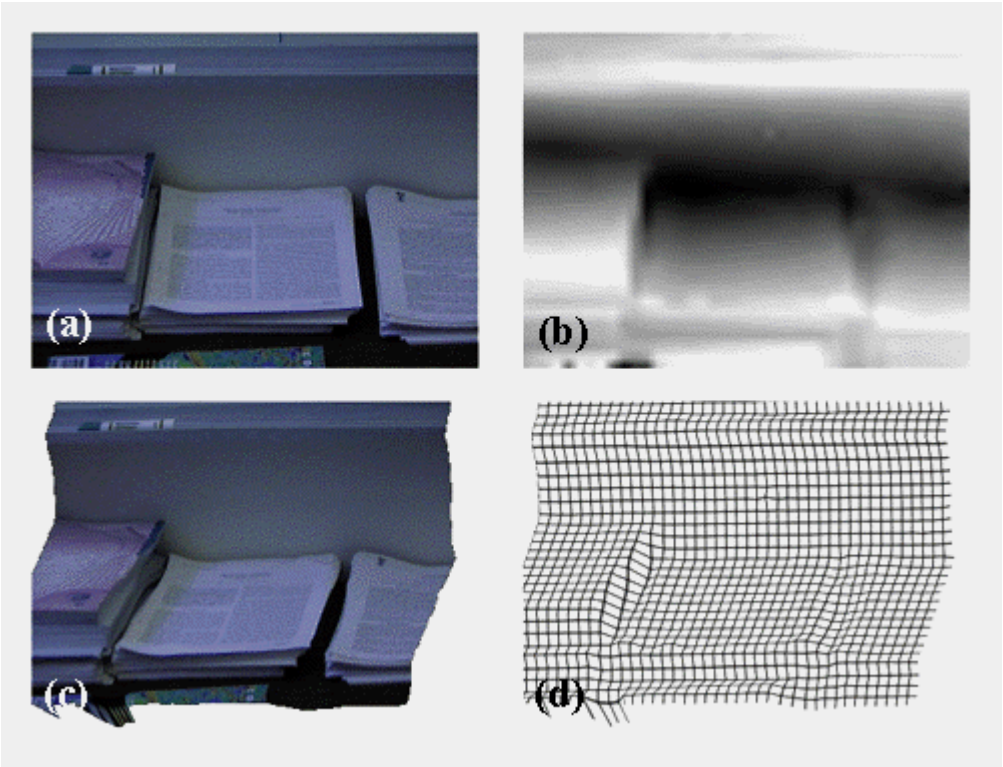


Figure 7. Depth recovery example—table with stacks of papers: (a) input image, (b) intensity-coded depth map (dark is farther back), (c) texture-mapped surface seen from a novel viewpoint, and (d) gridded surface.

Figure 8a shows results from another sequence—an outdoor scene of some trees. Again, the geometry of the scene is recovered reasonably well, as indicated in the depth map in Figure 8b.

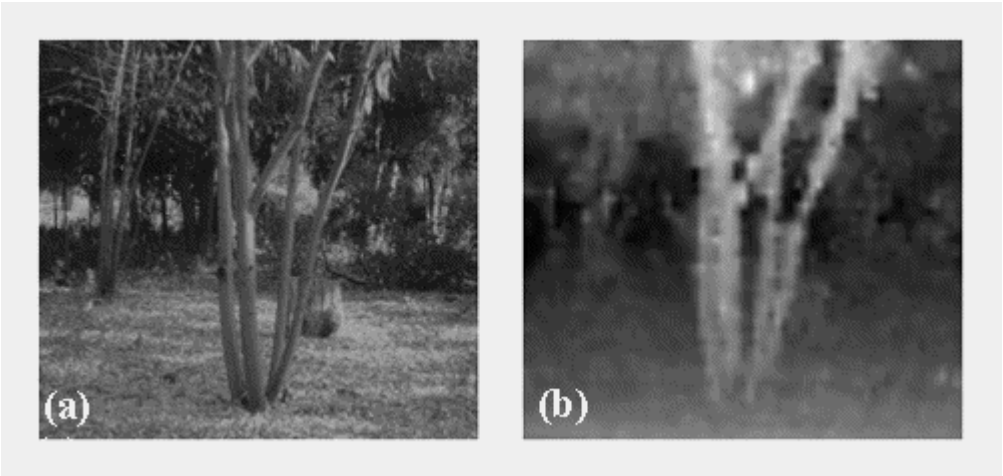


Figure 8. Depth recovery example—outdoor scene with trees: (a) input image, (b) intensity-coded depth map (dark is farther back).

Applications

Given automated techniques for building 2D and 3D scenes from video sequences, what can we do with these models? This section describes several potential applications of video mosaics, including whiteboard and document scanning, environment and backdrop acquisition for special effects and video games, supermarket shopping at home, interactive walkthroughs of historical buildings, and live telepresence applications.

The most straightforward application of image mosaics 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 article can be used with any video camera attached to a computer. In specialized situations, for example, in classrooms, a computer-controlled camera could do the scanning, removing the need for automatic registration of the images. In general, combinations of image mosaicing and superresolution can be used to produce photographic stills of extremely high resolution.[9,16]

Document scanning is another obvious application of this technology. Hand-held scanners currently perform this function quite well. However, because 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 2D 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 (for example, column borders).

The ability to mosaic video frames in real time opens up additional possibilities, such as using the camera as a "paintbrush" to generate large composite scenes interactively. The combination of such mosaics with live video streams produces interesting effects, where the newest frames appear to be "live," while older frames (presumably in other areas of the composite image) appear to be "frozen" in time.

A potential mass-market application is home shopping access to complete stores, such as your local supermarket. This has the advantage of a familiar look and organization, and lets you plan 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 by a video-based 3D model-building process.[10] In the future, these models will be available directly from the manufacturer. The shopper can then stroll down the aisles, pick out individual items, and look at their ingredients and prices.

Panoramic mosaics (environment maps) can enhance special effects used in movies, for example, computing illumination maps and reflections or filling in backgrounds. Such panoramas could also serve as backdrops for video games. A collection of such panoramas could be used in a limited form of virtual reality, such as showing the views inside different

rooms in a museum or historic building.[11] A museum scenario might include the ability to look at individual 3D objects such as sculptures and to bring up related information in a hypertext system.

Building true 3D models of buildings, perhaps using piecewise-planar representations, opens up many more possibilities. For example, interactive 3D walkthroughs of your home, built by walking a video camera through the rooms and processing the image sequences, could be used for selling your house (an extension of existing still-image based systems) or for remodeling or renovations. However, building complete models of room interiors, including furniture, requires true 3D model reconstruction techniques (see Szeliski[10] for some recent results).

You can also create walk- or fly-throughs of general outdoor 3D scenes. Example applications include flight and driving simulators, and virtual travel (*teletourism*). Such 3D scene models can have extremely high complexity and require solutions to problems in representing the images, employing partial 3D models, and switching between different levels of resolution.

The ultimate in virtual reality systems is dynamic virtual reality (sometimes called *telepresence*), which composites video from multiple sources in real time to create the illusion of being in a dynamic (and perhaps reactive) 3D environment. An example application might be to view a 3D version of a concert or sporting event with control over the camera shots, even seeing the event from a player's point of view. Other examples might be to participate or consult in a surgery from a remote location (*telemedicine*) or to participate remotely in a virtual classroom. Building such dynamic 3D 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.

Discussion

Video mosaics provide a powerful new way of creating the detailed environments needed for virtual reality applications. By quickly registering multiple images together, it is possible to create scenes of extremely high resolution and simultaneously recover partial 3D geometric information. The approach used here—namely, direct minimization of intensity differences between warped images—has a number of advantages over more traditional techniques, which are based on tracking features from frame to frame.[15] The techniques here produce dense estimates of shape. They work in highly textured areas where features may not be reliably observed, and they make statistically optimal use of all the information. In addition, the depth map recovery algorithm, described in the section "Projective depth recovery," makes it possible to perform view interpolation directly on real-world scenes, rather than just on synthetically generated graphics.

While the techniques described in this article have worked well in the scenes in which I have tried them, I remain cautious about their general applicability. Intensity-based

techniques 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 and proper image blending can remove many of these problems. Registration 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 reported here were obtained with uncalibrated cameras). Other potential sources of error include limited depth of field and imperfect alignment of rotational axes in panoramic scene compositing.

The depth extraction techniques rely on the presence of texture in the image. Even in areas of sufficient texture, the registration/matching algorithm can still compute erroneous depth estimates, for example, due to repetitive texture patterns or occlusions. Where texture is absent, interpolation must be used, and this can also lead to erroneous depth estimates.

Conclusion

The techniques presented here automatically register video frames into 2D and partial 3D scene models. Truly realistic virtual environments will require 3D object models as well as environment maps. The automatic construction of such models directly from video is the subject of ongoing investigations.[10]

The creation of realistic high-resolution scenes from video imagery opens up many new applications. Ultimately, as processing speeds and reconstruction algorithms improve further, video mosaics and related techniques will enable an even more exciting range of interactive computer graphics, telepresence, and virtual reality applications.

References

1. N. Greene, "Environment Mapping and Other Applications of World Projections," *IEEE CG&A*, Vol. 6, No. 11, Nov. 1986, pp. 21-29.
2. T. Beier and S. Neely, "Feature-Based Image Metamorphosis," *Computer Graphics* (Proc. Siggraph), Vol. 26, No. 2, July 1992, pp. 35-42.
3. G. Wolberg, *Digital Image Warping*, IEEE Computer Society Press, Los Alamitos, Calif., 1990.
4. J.D. Foley et al., *Computer Graphics: Principles and Practice*, 2nd edition, Addison-Wesley, Reading, Mass., 1990.
5. W.H. Press et al., *Numerical Recipes in C: The Art of Scientific Computing*, 2nd edition, Cambridge Univ. Press, Cambridge, England, 1992.
6. R. Szeliski and J. Coughlan, "Hierarchical Spline-Based Image Registration," *Proc. IEEE Computer Soc. Conf. Computer Vision and Pattern Recognition (CVPR 94)*, IEEE CS Press, Los Alamitos, Calif., 1994, pp. 194-201.
7. C.D. Kuglin and D.C. Hines, "The Phase Correlation Image Alignment Method,"

IEEE Conf. on Cybernetics and Society, IEEE, New York, 1975, pp. 163-165.

8. H.E. Malde, "Panoramic Photographs," *American Scientist,* Vol. 71, No. 2, Mar.-Apr. 1983, pp. 132-140.

9. S. Mann and R.W. Picard, "Virtual Bellows: Constructing High-Quality Images from Video," *Proc. First Int'l Conf. Image Processing,* Vol. I, IEEE CS Press, Los Alamitos, Calif., 1994, pp. 363-367.

10. R. Szeliski, "Image Mosaicing for Tele-Reality Applications," *Proc. IEEE Workshop on Applications of Computer Vision,* IEEE CS Press, Los Alamitos, Calif., 1994, pp. 44-53.

11. S.E. Chen, "QuickTime VR_An Image-Based Approach to Virtual Environment Navigation," *Proc. Siggraph 95,* ACM, New York, 1995, pp. 29-38.

12. L. McMillan and G. Bishop, "Plenptic Modeling: An Image-Based Rendering System," *Proc Siggraph 95,* ACM, New York, 1995, pp. 39-46.

13. L. Williams, "Pyramidal Parametrics," *Computer Graphics* (Proc. Siggraph), Vol. 17, No. 3, July 1983, pp. 1-11.

14. S. Chen and L. Williams, "View Interpolation for Image Synthesis," *Proc. Siggraph 93,* ACM, New York, 1993, pp. 279-288.

15. O. Faugeras, *Three-Dimensional Computer Vision: A Geometric Viewpoint,* MIT Press, Cambridge, Mass., 1993.

16. L. Teodosio and W. Bender, "Panoramic Overview for Navigating Real-World Scenes," *Proc. ACM Multimedia 93,* ACM Press, New York, 1993, pp. 359-364.

Richard Szeliski *is a senior researcher in the Advanced Technology and Research division of Microsoft Corporation. His research interests include 3D computer vision, motion estimation, and virtual environments. He received a BEng in electrical engineering from McGill University, Montreal, in 1979, an MEng in electrical engineering from the University of British Columbia, Vancouver, in 1981, and a PhD in computer science from Carnegie Mellon University, Pittsburgh, in 1988. Szeliski is the author of* Bayesian Modeling of Uncertainty in Low-Level Vision *(Kluwer). He is a member of ACM, IEEE, and Sigma Xi and associate editor of the* IEEE Transactions on Pattern Analysis and Machine Intelligence.

2D projective transformations

Planar scene views are related by projective transformations. This is true in the most general case—that is, for any 3D-to-2D mapping

$$\mathbf{u} = \mathbf{P}\mathbf{p}$$

(i)

where **p** = (*x*, *y*, *z*, *w*) is a 3D world coordinate, **u** = (*x*',*y*',*w*') is the 2D screen location, and **P** is the 3 × 4 camera matrix defined in Equation 4.

Since **P** is of rank 3, we have

$$\mathbf{p} = \mathbf{M} * \mathbf{u} + s\mathbf{m}$$

(ii)

where $\mathbf{M}^* = \mathbf{P}^T(\mathbf{P}\mathbf{P}^T)^{-1}$ is the pseudoinvrse of **P** and **m** is in the null space of **P**, that is, **Pm** = 0. The equation of a plane in world coordinates can be written as

$$ax + by + cz = d \quad \text{or} \quad \mathbf{n} \cdot \mathbf{p} = 0$$

(iii)

from which we can conclude that

$$\mathbf{n}^T\mathbf{M} * \mathbf{u} + s\mathbf{n} \cdot \mathbf{m} = 0 \quad \text{or} \quad s = -(\mathbf{n}^T\mathbf{M} * \mathbf{u})/(\mathbf{n} \cdot \mathbf{m})$$

(iv)

and hence

$$\mathbf{p} = (\mathbf{I} - (\mathbf{n} \cdot \mathbf{m})^{-1}\mathbf{m}\mathbf{n}^T)\mathbf{M}^*\mathbf{u} = \tilde{\mathbf{M}}\mathbf{u}$$

(v)

From any other viewpoint, we have

$$\mathbf{u}' = \mathbf{P}'\tilde{\mathbf{M}}\mathbf{u} = \mathbf{M}\mathbf{u}$$

(vi)

which is a 2D planar projective transformation.

In the absence of prior information about the camera location, we can assume that the world coordinate system coincides with the first camera position, that is, $\mathbf{E} = \mathbf{I}$ and $\mathbf{P} = \hat{\mathbf{V}} = [\,\mathbf{V} \;\; \mathbf{0}\,]$, and therefore $\mathbf{M}^* = \mathbf{V}^{-1}$ and $\mathbf{m} = (0, 0, 0, 1)^T$ in Equations ii through v. An image point **u** then corresponds to a world coordinate **p** of the form

$$\mathrm{p} = \begin{bmatrix} \mathbf{V}^{-1}\mathbf{u} \\ w \end{bmatrix}$$

(vii)

where *w* is the unknown fourth coordinate of **p** (called *projective depth*), which determines how far the point is from the origin.

For panoramic image mosaics, we can rotate the point **p** around the camera optical center to obtain

$$\mathrm{p}' = \begin{bmatrix} \mathbf{R} & 0 \\ 0^T & 1 \end{bmatrix} \mathrm{p} = \begin{bmatrix} \mathbf{R}\mathbf{V}^{-1}\mathbf{u} \\ w \end{bmatrix}$$

(viii)

with a corresponding screen coordinate

$$\mathbf{u}' = \mathbf{V}'\mathbf{R}\mathbf{V}^{-1}\mathbf{u} = \mathbf{M}\mathbf{u}$$

(ix)

Thus, the mapping between the two screen coordinate systems can be described by a 2D projective transformation.

For projective depth recovery, each image point **u** corresponds to a 3D point **p** with an unknown projective depth *w* as given in Equation vii. In some other frame, whose relative orientation to the first frame is denoted by **E**, we have

$$\mathbf{u}' = \mathbf{V}'\mathbf{E}\mathbf{p} = \mathbf{V}'\mathbf{R}\mathbf{V}^{-1}\mathbf{u} + w\mathbf{V}'\mathbf{t} = \mathbf{M}\mathbf{u} + w\tilde{\mathbf{t}}$$

(x)

Thus, the motion between the two frames can be described by our familiar 2D planar projective motion, plus an amount of motion proportional to the projective depth along the direction $\tilde{\mathbf{t}}$ (which is called the *epipole*).