Image-Based Interactive Exploration of Real-World Environments

Matthew Uyttendaele, Antonio Criminisi, Sing Bing Kang, Simon Winder, and Richard Szeliski
Microsoft Research

Richard Hartley
Australian National University and National ICT Australia

An image-based acquisition and rendering system lets users interactively explore remote real-world locations. A multisensor omnidirectional camera helps provide a compelling sense of presence.

Interactive scene walk-throughs have long been an important computer graphics application area. Starting with Fred Brooks' pioneering work,[1] efficient rendering algorithms have emerged for visualizing large architectural databases. More recently, researchers have developed techniques for constructing photorealistic 3D architectural models from real-world images.[2] Real-world tours based on panoramic images also exist, as we describe in the "Panoramic Imaging" sidebar. These systems all aim to create a real sense of *being there*—a sense of virtual presence that lets users experience a space or environment in an exploratory, interactive manner.

This article presents an image-based rendering system that brings us a step closer to a compelling sense of being there. Whereas many previous systems have used still photography and 3D scene modeling, we avoid explicit 3D reconstruction because it tends to be brittle. Instead, we film a tour of an environment that we wish to explore and then use image-based rendering techniques to replay the tour interactively, as Figure 1 shows. We call such experiences *interactive visual tours*. In designing this system, our goal was to let users move freely along a set of predefined tracks, choose between different directions of motion at decision points, and look in any direction. In addition, the displayed views should have a high resolution and dynamic range. The system easily incorporates multimedia objects such as navigation maps, video textures,[3] audio enhancements, and pop-up stills.

Our system is not the first to propose interactive video-based tours (see the "Panoramic Imaging" sidebar). We believe, however, that our system is the first to deliver fully interactive, photorealistic image-based tours on a personal computer at or above broadcast video resolutions and frame rates. Moreover, to our knowledge, no other tour provides the same rich set of interactions or visually complex environments.

System overview

To provide high-resolution interactive image-based tours, we designed an integrated acquisition, author-

(a) (b) (c)

1 Navigating an interactive virtual tour. (a) Users can virtually explore large environments from their sofas. (b) A standard wireless gamepad lets the user navigate easily through the captured environment. (c) The system displays the environment on a large wall screen.

Published by the IEEE Computer Society

Panoramic Imaging

The history of panoramic photography using mechano-optical means stretches back to the 19th century. Computer software for stitching photographs has been around for a few decades, but only gained widespread commercial use with the release of Apple's QuickTime VR system in 1995.[1] Since then, the basic system has seen many improvements, including mechanisms for constructing full-view panoramas, dealing with moving objects, and using panoramas for image-based rendering and architectural reconstruction. (A collection of recent papers in this area is available elsewhere.[2]) Most mid- to high-end digital cameras now ship with image-stitching software, and more than a dozen commercial stitching products are on the market today (http://www.panoguide.com/software lists several).

Kodak first introduced panoramic (360-degree) movies at the world's fair in 1964. Andrew Lippman first demonstrated interactive video tours in his MovieMaps project,[3] which stored video clips of the streets of Aspen, Colorado, on an optical videodisc and let viewers interactively navigate through the clips. In mobile robotics, 360-degree video cameras based on reflecting curved mirrors (catadioptric systems) have long been used for robot navigation.

More recently, researchers have used such systems to present virtual tours and build 3D image-based environment models. For example, Boult[4] developed a campus tour based on reprojecting catadioptric spherical images streaming from a camcorder to let users look around while driving through campus. Coorg and Teller[5] built detailed 3D architectural models of the MIT campus based on high-resolution still panoramic images taken from a specially instrumented cart with a pan-tilt head. Aliaga and Carlbom[6] built image-based interactive walk-throughs of indoor rooms based on a dense sampling of omnidirectional images taken from a mobile robot platform. Taylor's VideoPlus system[7] uses a similar setup but relies more on a sparse traversal of a set of connected rooms to explore a larger region of interest.

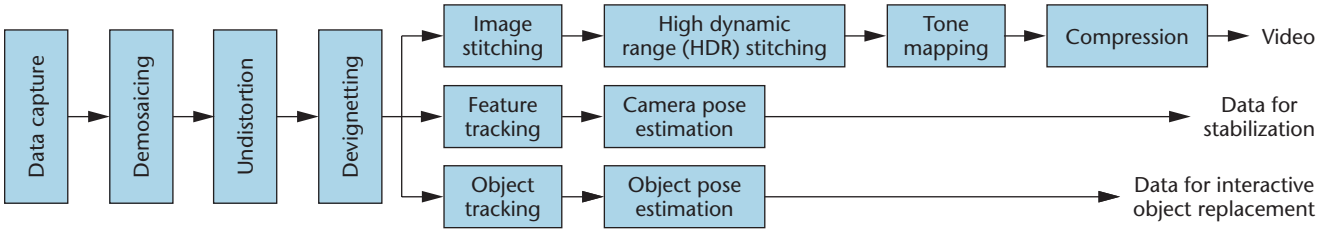
Daniilidis' Omnidirectional Vision Web page (http://www.cis.upenn.edu/~kostas/omni.html) lists about 50 research projects and commercial systems based on omnidirectional cameras. Most of the systems use curved mirrors and a single camera, and a few use multiple cameras. One clever design (http://www.fullview.com) uses mirrors to make the optical centers coincident, which removes the parallax problems but results in a slightly bulkier system. The system built by iMove (http://www.imoveinc.com) is the most similar to ours in that it uses a collection of outwardly pointing video cameras but has a lower spatial resolution than the Ladybug.

References

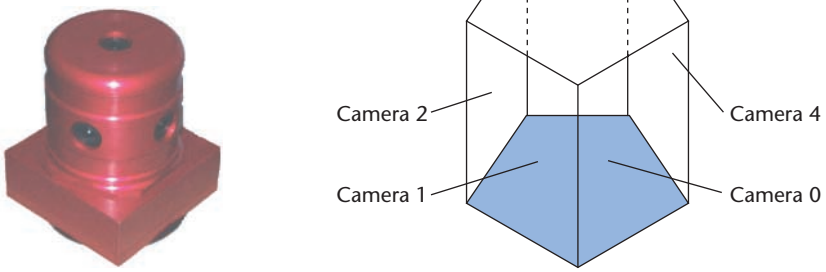
1. S.E. Chen, "QuickTime VR: An Image-Based Approach to Virtual Environment Navigation," *Proc. Siggraph*, ACM Press, 1995, pp. 29-38.
2. *PanoramicVision: Sensors, Theory, and Applications*, R. Benosman and S.B. Kang, eds., Springer, 2001.
3. A. Lippman, "Moviemaps: An Application of the Optical Videodisc to Computer Graphics," *Proc. Siggraph*, ACM Press, vol. 14, no. 3, 1980, pp. 32-43.
4. T.E. Boult, "Remote Reality via Omnidirectional Imaging," *Proc. Siggraph 1998 Technical Sketch*, ACM Press, 1998, p. 253.
5. S. Coorg and S. Teller, "Extracting Textured Vertical Facades from Controlled Close-Range Imagery," *Proc. IEEE Computer Soc. Conf. Computer Vision and Pattern Recognition* (CVPR 99), vol. 1, IEEE CS Press, 1999, pp. 625-632.
6. D.G. Aliaga and I. Carlbom, "Plenoptic Stitching: A Scalable Method for Reconstructing 3D Interactive Walkthroughs," *Proc. Siggraph*, ACM Press, 2001, pp. 443-450.
7. C.J. Taylor, "Videoplus: A Method for Capturing the Structure and Appearance of Immersive Environments," *IEEE Trans. Visualization and Computer Graphics*, vol. 8, no. 2, Apr.-June 2002, pp. 171-182.

ing, and playback system from the ground up, as Figure 2 illustrates. The novel components we developed as part of this system include the following:

- a multicamera high-resolution video head connected to a small, portable redundant array of independent disks (RAID) and power supply;
- an acquisition platform to conveniently transport the sensor and capture the space we wish to explore;
- a suite of image-processing algorithms to denoise, demosaic, unwarp, and devignette the images;
- a spatial stitching/deghosting algorithm to minimize visible parallax in overlap regions;
- a high dynamic range (HDR) video-stitching algorithm that merges successive frames taken at different exposures with a moving camera;
- a feature-tracking and pose-estimation algorithm to stabilize the acquired video and aid navigation;
- a video-compression scheme designed for selective (partial) decompression and random access;
- authoring tools for inserting branch points into the tour, adding sound sources, and augmenting the



2 Simplified block diagram of our video-processing and rendering pipelines.



(a) (b)

3 Omnidirectional camera design: (a) A six-sensor omnidirectional camera. We arranged the six images taken by our camera horizontally around a pentagon, with a sixth sensor looking straight up. During processing, our system merges the overlapping fields of view and resamples them. During rendering, these images become (b) the texture maps used in the 3D environment map.

video with synthetic and hybrid elements; and
- an interactive viewer that lets users easily navigate and explore the captured space.

The system's overall complexity is hidden from the end user, who intuitively navigates compelling interactive tours of remote locations using a simple game control pad.

Omnidirectional camera design

Sensor resolution and bandwidth limitations cause many current systems to display panoramic images and videos in small windows. When these systems display a wide field-of-view scene in a small portion of the observer's field of view, artifacts such as *swimming* (wavy motion during virtual panning) occur, detracting from the immersive nature of panoramic imaging. Interactive 3D computer games, however, are typically rendered as full-screen experiences, giving the user a more immersive feel of the environment. Thus, one of our effort's primary goals was to let users explore large real-world environments with full-screen fidelity. Computer games also let users explore an environment with continuous translational and rotational motion, and we wanted our real-world tours to have similar characteristics. Furthermore, we wanted the flexibility to capture a wide range of environments, including traditional architecture and rugged outdoor environments.

The first step in achieving these goals is to design a capture system. To efficiently acquire immersive environments, the system must capture full spherical environments at high resolution. One approach is to capture many high-resolution stills and stitch them together, repeating this step at many locations throughout the environment.[4] Although this yields high-resolution panoramas, it's a time-consuming process. We decided instead on omnidirectional capture at video rates.

A common approach to omnidirectional capture is to use catadioptric systems consisting of mirrors and lenses. Mirror-based designs use either curved mirrors with a single lens and sensor or multiple planar mirrors with associated cameras. Systems based on these designs capture a large cross-section of a spherical environment and require no complex stitching because they have a single optical center. To enable full-screen experiences, our system must capture enough pixels to provide at least a 640 × 480 resolution in any given 60-degree field of view.

A curved mirror design would need at least a 2,000 × 2,000 video-rate sensor to approach these requirements, but the desired resolution would only be within a fraction of the system's field of view. This is because these designs have a nonuniform image-resolution distribution, which drops significantly toward the pole. In addition, these designs have blind spots at the pole where the camera itself blocks the field of view.

A multiple-camera approach requires arranging the cameras so they share a common center of projection. The user postprocesses the captured data with a more complex stitching and parallax compensation step to produce seamless panoramas. However, packing the cameras tightly together can minimize parallax. On the plus side, such a system can capture most of a full-viewing sphere, we can make it high resolution using multiple sensors, and we can build it using a rugged and compact design. We therefore selected this design.

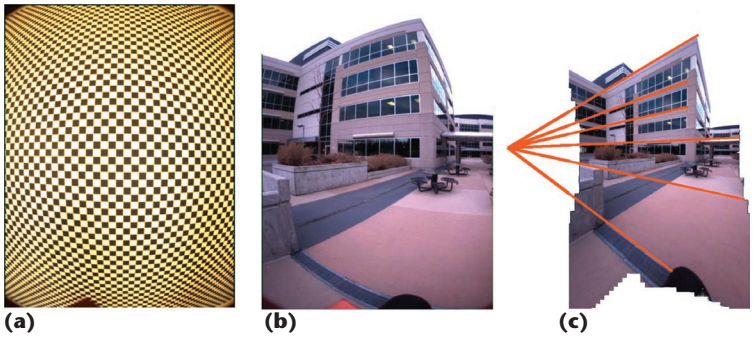
Because no commercially available system met all of our requirements, we designed our own multicamera system and contracted Point Grey Research to build it. The result is the Ladybug camera (http://www.ptgrey.com/products/ladybug), shown in Figure 3.

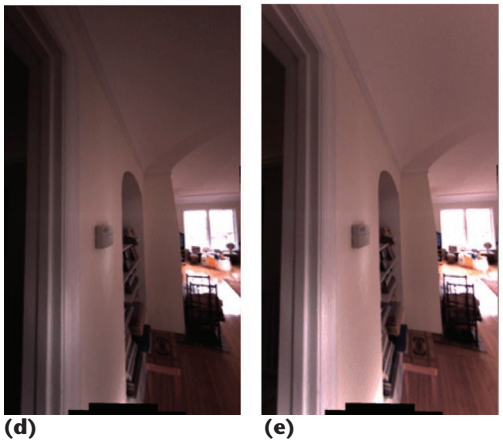
We divided the acquisition system into two parts: a *head unit* containing the cameras and control logic, and a *storage unit* tethered to the head via a fiber-optic link. The head unit contains six 768 × 1,024 video-rate sensors and associated lenses: five point out along the equator and one points up, giving complete coverage from the system's north pole to 50 degrees below its equator. We chose the lenses so that a small overlap would exist between adjacent cameras, allowing for stitching of the captured data. The synchronized sensors run at a video rate of 15 frames per second (fps).

The storage unit is a portable RAID array consisting of four 40-Gbyte hard drives, which can store up to 20 minutes of uncompressed raw video. A small portable battery powers all components, and the complete system can fit in a backpack or on a small robot.

Data acquisition

We used two different mobile platforms to capture our omnidirectional videos. We designed both to avoid imaging themselves and the camera operators.





4 Radial distortion and vignetting correction: (a) image of calibration grid; (b) original photograph of a building showing lens distortion; (c) corrected image, with straight lines converging on a vanishing point; (d) original image showing vignetting effect; and (e) image after vignetting correction.

Our first design was a tripod mounted on a dolly. This platform is moved along the planned path by a human crouched next to it. This works if the ground is flat. To handle arbitrary terrain, we used the second design—that is, a sky-diving helmet with the camera mounted on top. We can use this platform practically anywhere (we've even used it while sky-diving), and the pace taken along the path during acquisition is reasonably rapid. Unfortunately, camera jitter is a significant issue with this design.

Image preprocessing

The first step in our offline processing pipeline is converting the raw video values from our sensor chip to color images and correcting for radial distortion and vignetting.

Demosaicing

We copy images directly from the sensors onto the RAID array as raw Bayer color filter array values (that is, individual green, red, or blue samples arranged in a checkerboard array consisting of 50 percent green pixels and 25 percent each red and blue pixels). We use Chang et al.'s[5] algorithm to interpolate to full RGB images while preserving edge detail and minimizing spurious color artifacts. We also undo the camera gamma so the rest of the processing can occur in a linear color space.

Radial distortion

Our cameras' large field of view (approximately 100 degrees) introduces significant lens distortion, which we must remove to achieve seamless stitching where images overlap.

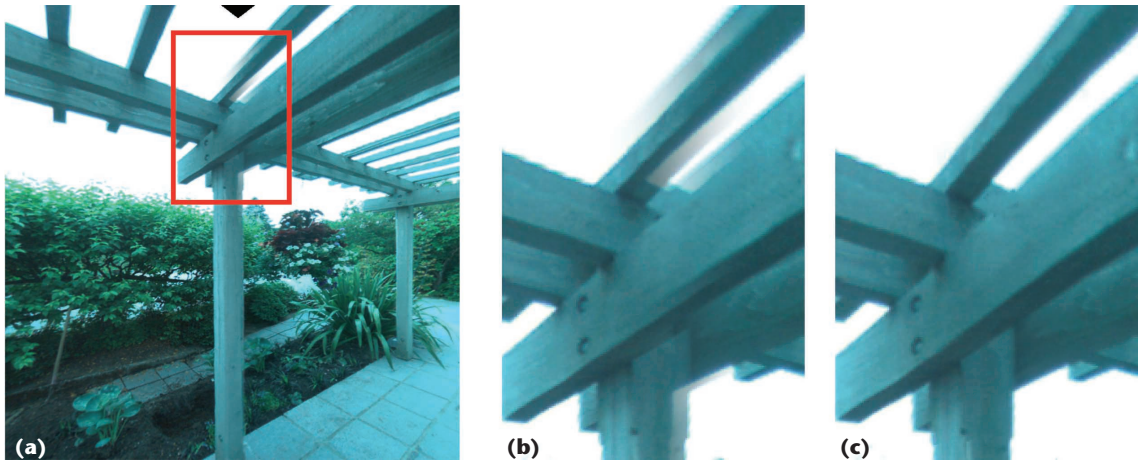
We achieved the required level of lens accuracy by combining a parametric radial distortion model (a traditional plumb line technique) and a point-based matching algorithm to remove the residual error. The algorithm simply finds all corners of a grid's image (Figure 4a) and maps them back to their original locations. Interpolating the output displacements with a bilinear spline produces the final pixel-based correction flow. This approach produces considerably more pixel-accurate results than traditional plumb line techniques.

Vignetting

Cameras with wide fields of view also suffer from a vignetting effect—an intensity drop-off away from the center. Both optical (off-axis illumination) and geometric (aperture) factors cause this drop-off. We compensate for these effects by first capturing an image within an integrating sphere whose internal surface is almost perfectly Lambertian, and then fitting a global function that accounts for this effect. The function we use is of the form $I(r) = I_0(r)[1 - \alpha r]/[1 + (r/f)^2]^2$, where r is the radial distance from the principal point (assumed to be the image center), $I(r)$ is the sampled image, $I_0(r)$ is the hypothesized unattenuated intensity image, f is the camera focal length, and α is the geometric attenuation factor.

Geometric calibration

After undoing radial distortion for each camera so that they fit a linear perspective model, we must compute the linear transforms relating camera pixels to world rays relative to the camera head. One approach is to track a large number of points while rotating the head and then perform a full bundle adjustment (struc-



5 Parallax compensation in image overlap: (a) direct feathering introduces blurring, (b) magnified version of the marked region in (a), and (c) stitching results after multiperspective plane sweep, producing much cleaner edges.

ture from motion reconstruction to recover both camera pose and 3D point location).

We chose a simpler approach. We take an image of an outdoor scene containing negligible parallax and compute the camera's intrinsic parameters and rotations using an image-stitching algorithm.[6] We then estimate the intercamera translations from the design specifications. Although these aren't the exact displacements between the optical centers, they're close enough that we can compensate for intercamera parallax.

Image stitching and parallax compensation

Because the camera centers of our omnidirectional capture system aren't coincident, the observed parallax can be significant in image overlap regions, especially when portions of the scene are nearby. Under such conditions, using simple feathering or blending to produce a stitched omnidirectional image isn't acceptable, as Figures 5a and 5b illustrate.

To handle this problem, we adapted a stereo-matching process to recover a *multiperspective image* in the overlap region. This process matches each column in the overlap region using a slightly different virtual viewpoint. For a given intermediate viewpoint, we apply stereo to find the most photoconsistent appearance given two input images, I_1 and I_2—that is, we search along a set of predetermined depths (*plane sweeping*) and find the minimum difference between projected local colors from the input images at those depths. This technique, which we call *multiperspective plane sweep* (MPPS),[7] eliminates abrupt visual discontinuities in the stitched image. Figure 5c shows the resulting image when we apply MPPS.

High dynamic range capture and viewing

One recurring difficulty in capturing real-world environments is that the dynamic range can be high, especially if both interior architecture and exterior scenery are visible simultaneously. (Omnidirectional imaging only exacerbates this problem.) For this reason, we modified our sensor to capture multiple exposures in quick succession and developed new algorithms to compute both HDR and tone-mapped versions of each frame. The "High Dynamic Range Photography" sidebar details this process and briefly reviews HDR imaging.

Camera pose estimation and bifurcation handling

Despite recent research efforts, no existing vision algorithms for camera pose recovery (ego motion) robustly handle large-scale environments such as those our system captures. The main problem is that the error accumulation over long sequences degrades camera translation accuracy.

Fortunately, we don't need full-motion estimation. In fact, accurate rotations are sufficient to obtain compelling 3D walk-throughs. Moreover, whereas traditional camera systems blur the distinction between small translations and small rotations, omnidirectional acquisition systems such as ours make this difference clear.

Point tracking

Our approach to orientation estimation starts with robust, omnidirectional feature tracking. Our algorithm follows a standard vision approach in terms of corner detection and robust matching. However, unlike previous approaches, our tracker follows feature points that move from one camera viewpoint to an adjacent viewpoint. Our algorithm proceeds as follows:

1. Using a Harris corner detector,[8] it detects feature points to subpixel precision in all six views.
2. Because the calibration parameters are known, the algorithm converts 2D corner positions into 3D ray vectors with respect to a unique coordinate frame at the camera head center, as Figure 6a illustrates.
3. For each 3D ray, the algorithm computes points of intersection with all cameras. The algorithm chooses the image in which the point of intersection is closest to the image center, along with a 5×5 window of pixels centered on the point. This process minimizes the distortion of sampled local appearance.

High Dynamic Range Photography

The real world contains more brightness variation than most cameras' digital sensors can capture. An indoor photograph's radiance range can contain as many as six orders of magnitude between the dark areas under a desk to the sunlit views seen through a window. Typical charge-coupled device or complementary metal-oxide semiconductor sensors capture only two to three orders of magnitude and thus can't capture details in both the dark and the bright areas at the same time.

Many solutions to this problem exist. One way to capture greater dynamic range of still scenes is to shoot multiple exposures, which appropriately capture tonal detail in dark and bright regions in turn. Combining these images can create a high dynamic range (HDR) image. Automatically combining such images requires knowing the camera's response curve as well as the camera settings used for each exposure. Other work details techniques used to create HDR images from multiple exposures.[1] Each technique produces an HDR image with at least 16 bits per color component.

An alternate method is to combine these images directly in an image-editing tool such as Adobe Photoshop. Two Web sites offer tutorials on how to do this: http://www.luminous-landscape.com/tutorials/digital-blending.shtml and http://www.digitalsecrets.net/secrets/DynamicRanger.html.

Viewing or displaying HDR images is also a problem. A typical computer monitor has a dynamic range of about two orders of magnitude, and printers have even less. To display an HDR image, its range must be compressed to match that of the display. Other researchers have explored this operation, known as tone mapping or tone reproduction.[2,3]

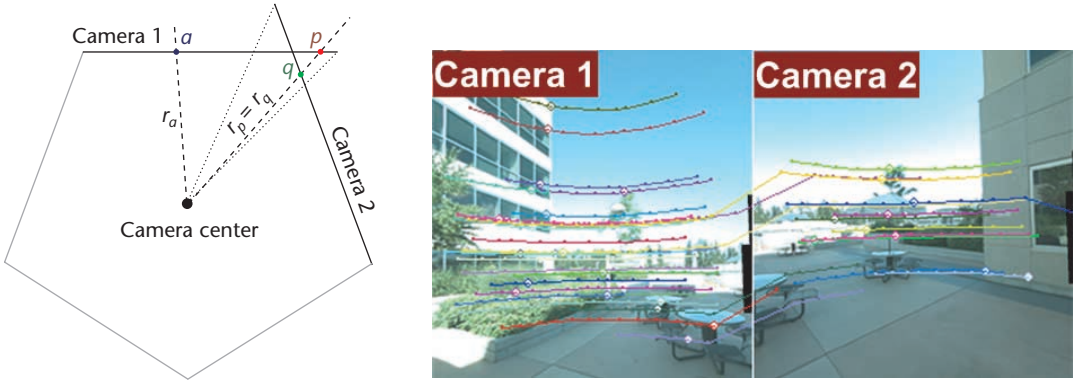
With our system, we generate HDR video—that is, we generate one HDR image at each time interval. This is challenging because we can't capture all the desired exposures simultaneously. Furthermore, different image regions might be in motion. We describe our solution to this problem elsewhere:[4] here, we provide a brief overview.

Our camera lets the exposure settings change at every frame time. We use this feature to capture a video consisting of alternating long and short exposures. We then solve the problem of registering these frames to each other so we can synthesize the missing long and short exposure frames at each given instant. The matching task is difficult because frames captured at different exposures are quite dissimilar. We facilitate the matching process by increasing the brightness of the lower exposure images to match the higher exposure images. After registration is complete, we compute a full radiance map by appropriately combining warped versions of the original video frames, using only the pixels with the most reliable correspondence and radiance values. We subsequently tone map the radiance map for viewing.

We adapted Reinhard et al.'s technique[3] to operate on HDR video, modifying it to use statistics from neighboring frames to produce a tone-mapped output without any temporal artifacts.

References

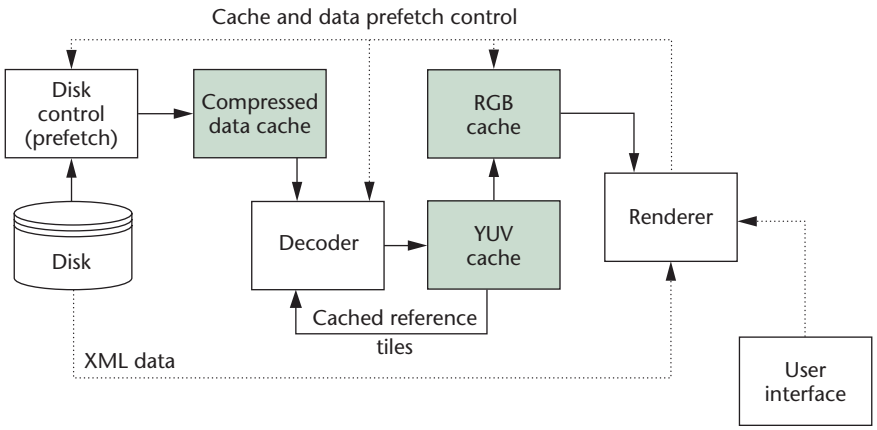
1. P.E. Debevec and J. Malik, "Recovering High Dynamic Range Radiance Maps from Photographs," *Proc. Siggraph*, ACM Press, 1997, pp. 369-378.
2. G.W. Larson, H. Rushmeier, and C. Piatko, "A Visibility Matching Tone Reproduction Operator for High Dynamic Range Scenes," *IEEE Trans. Visualization and Computer Graphics*, vol. 3, no. 4, 1997, pp. 291-306.
3. E. Reinhard et al., "Photographic Tone Reproduction for Digital Images," *ACM Trans. Graphics*, vol. 21, no. 3, 2002, pp. 267-276.
4. S.B. Kang et al., "High Dynamic Range Video," *ACM Trans. Graphics*, vol. 22, no. 3, July 2003, pp. 319-325.



(a) (b) (c)

6 Point tracking with omnidirectional cameras. (a) Plan view of the photo-bubble (omnidirectional image) model. Point *a* in camera 1 corresponds to a 3D ray r_a through the camera center. Points *p* and *q* on the overlap between two cameras correspond to the same feature in space. (b and c) The system detects and tracks feature points across time and across adjacent cameras. The differently colored curves show the positions of tracked points at different time instances; the larger circles show the tracked point's position in the current frame.

4. The algorithm matches pairs of 3D rays based on cross correlation. It uses the matched pairs to initialize a random sample consensus (RANSAC) process for robustly estimating the essential matrix and out-



7 Block diagram of playback engine showing decoder and caching configuration.

lier removal. Once we know the essential matrix between two frames, we use it to guide the matching of unmatched corners.
5. The algorithm concatenates the set of point matches along time to produce the long point tracks.

Using 3D rays instead of 2D corner features (step 2 above) allows feature matching across cameras, facilitating construction of long feature tracks across time and cameras. In Figures 6a and 6b, the red track corresponding to a point on the table is an example of tracking across cameras.

The large number of frames in a complete sequence (almost 10,000 frames in the smallest of our demos) and the enormous number of matched points (over 2 million tracked points) complicate the simultaneous recovery of structure and motion using standard bundle adjustment. However, estimating just the rotation component suffices for our application.

Orientation estimation and data stabilization

We use the robust essential matrices produced in step 4 of the point-tracking algorithm to estimate pair-wise interframe rotations. After initializing the orientation estimates by chaining together the interframe rotation estimates, we use nonlinear least squares to compute an optimal estimate of the camera orientations.[7] Because of accumulated errors, rotation estimates will inevitably drift from their correct values. To reduce this effect, we correct the drift at frames where the path self-intersects and use vanishing points, when possible, as constraints on each frame's absolute orientation.[4]

To reduce the jitter introduced at acquisition, we apply a low-pass filter to the camera rotations to obtain a smoother trajectory (damped stabilization) and subtract the original orientations (using quaternion division) to obtain per-frame correction. We can perform stabilization at authoring time by resampling the omnicamera mosaics onto a new set of viewing planes (potentially increasing compression efficiency), or at runtime, by modifying the 3D-viewing transformation in the viewer. Our current system uses the latter approach because it results in less imagery resampling.

With our current setup, compensating for the rotational component only achieves satisfactory video stabilization, although a small amount of lower-frequency up-and-down nodding (from the walking motion) remains. (Translational motion can't be compensated for without estimating 3D scene structure.) We could have used additional hardware components for jitter detection and removal (such as accelerometers), but we didn't pursue this. Buehler, Bosse, and McMillan present a related technique for stabilizing regular, monocular video sequences.[9]

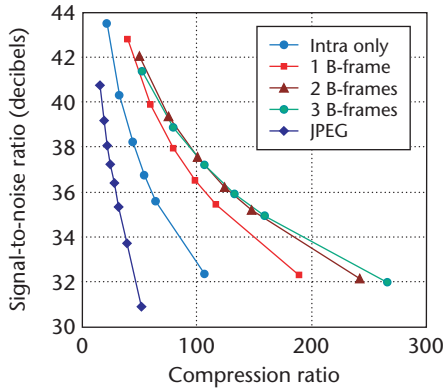
Compression and selective decompression

Real-time interactive playback of our tour content requires fetching image data from storage and rendering it rapidly. We therefore developed our own compression scheme that minimizes disk bandwidth and uses caching to support optimal performance. This scheme supports both

- *temporal* random access, so users can play the content forward and backward, as well as jump to arbitrary frames with minimal delay; and
- *spatial* random access, so the system can decompress currently viewed subregions of each frame independently, reducing bandwidth to the graphics card.

When rendering the interactive tour, we set up a six-plane Direct3D environment map that matches the camera configuration (Figure 3b) and treats the processed video streams as texture maps. To allow for spatial random access, we divide each $768 \times 1{,}024$ frame into 256×256 pixel tiles and compress them independently. We chose tile size based on a compromise between compression efficiency (a large tile allows for good spatiotemporal prediction) and random access granularity. Our codec uses both intra- (I) and inter- (B) coded tiles. The B-tiles are bidirectionally predicted and are similar to MPEG's B-frames. These tiles get their predictions from nearby past and future I-frames.

We divide tiles into 16×16 macroblocks. For I-tiles, we use discrete cosine transform (DCT) coding with spatial prediction of AC and DC coefficients in a manner similar to MPEG-4. B-macroblocks can be forward, backward, or bidirectionally predicted. Forward and backward motion vectors are independently spatially predicted.



8 Rate-distortion curves comparing our codec with JPEG compression. We used a 500-frame home-tour sequence, varying the number of B-frames between each I-frame. Compression ratio is the ratio of the total bitstream size to the total number of bits in the sequence when using the uncompressed RGB format.

Figure 7 shows our caching strategy, which minimizes the amount of repeated decompression necessary during typical viewing. The system decodes tiles for the current viewpoint only, caching them in case the user pans around. In addition, the system caches I-frame YUV tiles because they're used repeatedly during the decoding of consecutive B-frames. When decoding a B-tile, the decompressor determines which I-tiles are needed for motion compensation and requests them from the cache. If they aren't available, the cache calls upon the decoder to decompress additional I-tiles.

We also implemented a tile-based JPEG method with no temporal prediction. This decodes faster but offers much lower compression efficiency than our new scheme. Figure 8 plots the rate-distortion curves for our codec. As the figure's graph indicates, the JPEG compressed sequence has the worst performance. Our intracodec performs quite well, but adding predicted B-frames improves its performance, with two Bs for every I-frame being about optimal for this sequence. Using bidirectional coding, we obtain compression ratios of 60:1 with high visual quality.

Currently we maintain about 10 fps with our new codec but can achieve 30 fps with the less efficient JPEG compression. Because our codec provides access to lower data bandwidths, we're investigating the possibility of playing back our data sets from standard DVD media.

Multimedia experience authoring

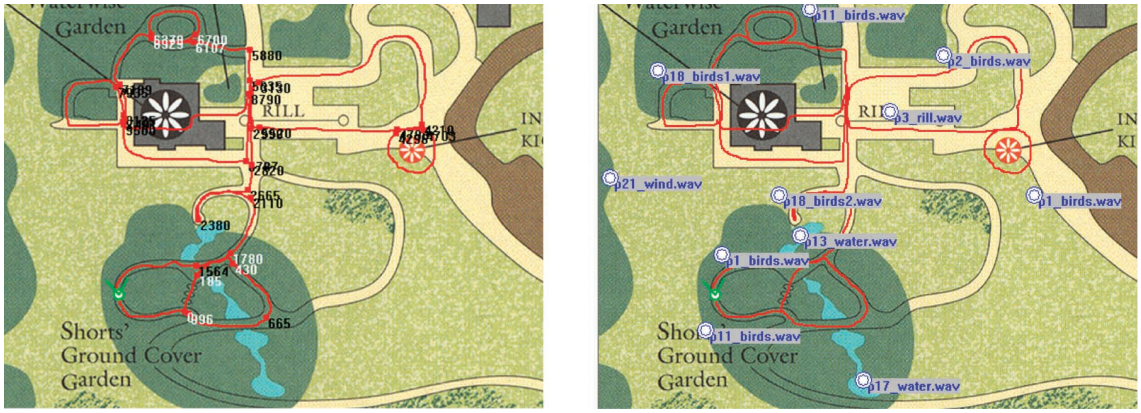
Adding multimedia elements using simple semiautomatic authoring tools enhances the final walk-throughs.

Bifurcation selection and map control

Whenever maps of the visited site are available, we incorporate them into our viewer to enhance the user experience. For instance, to help users navigate the space being visited, we display the current location and orientation on a map, as in Figure 9.

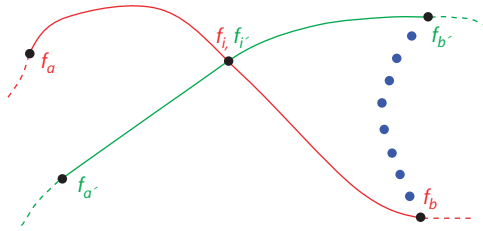
To enable the location display feature, we manually draw an approximate acquisition path on the map (red lines in Figure 9a) during the authoring stage. We then manually associate frame numbers with different key locations along the path (see Figure 9a). (We could also automate this process by correlating visual motion estimates with the hand-drawn map.) At viewing time, given the current view (with known frame number), we can estimate the observer's 2D position on the map and update it using simple linear interpolation.

Where the acquisition path's two branches intersect (for example, the red and green paths in Figure 10), we manually select two ranges of corresponding frames for the transition (for example, $[f_a, f_b]$ and $[f_{a'}, f_{b'}]$ in Figure 10). Our system then automatically estimates the best intersection—that is, the best pair of corresponding frames ($f_i, f_{i'}$ in Figure 10, next page). Instead of using explicit position information, we seek the pair of frames with the minimum visual distance[7] between them. Our algorithm constructs a matrix of such distances between each omnidirectional image (or photo bubble) in the range $[f_a, f_b]$ and every other photo bubble in the range $[f_{a'}, f_{b'}]$. The best pair, $f_i, f_{i'}$, has the smallest distance $d(f_i, f_{i'})$.



(a) **(b)**

9 Maps can enhance the navigation experience. The red line indicates the approximate camera path. The current viewer position and orientation is marked in green. (a) We mapped some key frame numbers to positions on the path and (b) placed some audio sources captured in situ on the map.



10 Bifurcation handling. Given a self-intersecting path, we want to find the best bifurcation—that is, the best pair of frames, f_i, $f_{i'}$, corresponding to the same map location.

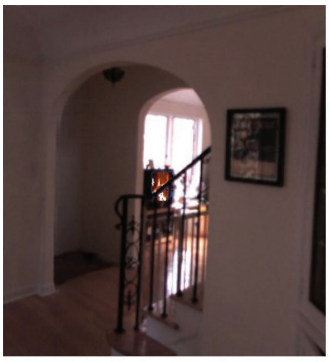
In our current implementation, the user must traverse a bifurcation node (f_i,$f_{i'}$) in Figure 10) to move to a new path. An alternative is to use new-view synthesis to generate novel intermediate views, letting the user move from one path to another using a synthesized path (the dotted blue path in Figure 10) and possibly resulting in a more seamless transition. (View interpolation might also compensate for the fact that paths might not exactly intersect in 3D.) However, because avoiding artifacts is difficult due to occlusions and nonrigid motions such as specularities and transparency in new-view synthesis, we haven't yet investigated this alternative.

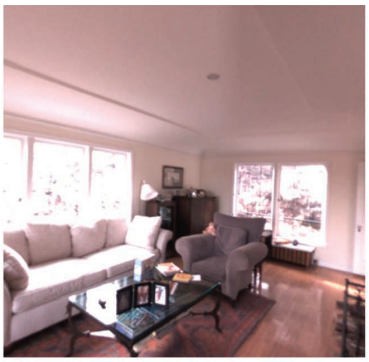
Object tracking and replacement

Interactive and dynamic elements can enhance the richness and realism of real-world environment visualizations. To support the insertion of such elements, we strategically position planar blue grids during data capture—for example, in front of a TV screen or inside a fireplace. During postprocessing, our tracking algorithm automatically estimates the target's position together with an occlusion mask, and the system stores this data along with the video (in a separate alpha channel). We can then replace the selected object with other still images, streaming video, or video textures (for repetitive elements such as fire or waterfalls[3]) during visualization while dealing correctly with possible occlusion events. For example, we can replace the paintings in a room or change the channel and display live video on the television set. Note that the blue grid is only necessary if the surface can be partially occluded in some views, as in Figure 11a. In easier cases, we can track and replace regular geometric objects such as picture frames without any special targets in the scene.

Audio placement and mixing

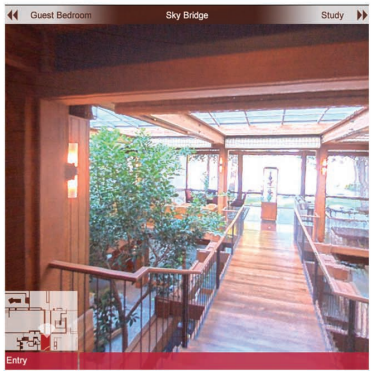
Adding spatialized sound can also make the tour richer. We acquire audio data in situ with a directional microphone, clean it up using a standard audio editing application, and then place it on the map to achieve a


(a)

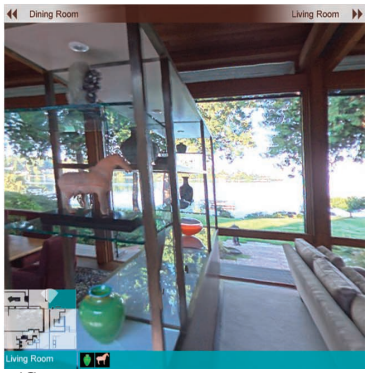

(b)


(c)


(d)



(e)



(f)

11 Snapshots of our demos. (a) Section of the dining area in our first home tour (see the "Results" section). (b) Living room in the home tour with no high dynamic range imaging. (c) Our Bellevue botanical garden model with a superimposed map. (d) At bifurcations in the garden tour, arrows appear on the ground to show allowed directions. (e) In the enhanced home tour demo, the top bar shows available choices at a given bifurcation. (f) Living room with the indoor and outdoor details simultaneously visible, thanks to our HDR video technique.

desired effect (Figure 9b). At rendering time, we attenuate the volumes of the audio files based on the inverse square distance rule and mix them using the audio card. This simple technique increases the experience's realism by conveying the feeling of moving closer or farther away from objects such as waterfalls, swaying trees, and pianos. In future work, we hope to investigate the use of true acoustic modeling for more realistic auralization.

Scene specification and rendering

The system's final component, the interactive viewing software (viewer), handles user input, the GUI (visual overlays), and video data rendering. For each walk-through, the authoring stage outputs a file in our XML-based scene description language (SDL). This file contains sound positions, stabilization information, bifurcation points, object locations, map information, and object tracking data. We developed SDL to allow flexible overlaying of a user interface onto the raw video data. It lets us rapidly prototype new user interfaces as the project evolves.

The viewer takes user input from a mouse or gamepad (Figure 1). The gamepad controls are mapped similarly to driving games. The left joystick lets the user pan left, right, up, or down, whereas the left and right trigger buttons let the user move forward and backward along the path.

Rendering occurs completely on the PC's 3D graphics hardware. As Figure 3b shows, the environment model is a set of six texture-mapped planes corresponding to the six camera focal planes. SDL specifies the planes' positions. This model works well for our particular camera geometry; a user could specify a sphere or parabaloid via SDL for other camera designs. We place the virtual camera at the center of this geometry and control its rotation via the joystick.

As the user navigates the environment, the viewer computes the appropriate data to request from the selective decompressor. The input video is often traversed nonsequentially. For fast motion, the stride through the input frames can be greater than one frame. In a bifurcation, the next frame requested can be in a totally different section of the original video. To accommodate this, we use a two-stage pipeline. During a given display time, the system computes the next frame to display and requests it from disk while the viewer decompresses and renders the frame already in the compressed cache. The system loads all the compressed bits for a given surround frame into the compressed cache. However, only a subset of these bits actually needs to be decompressed and rendered for any given view. The viewer intersects the view frustum with the environment geometry according to the user's viewing direction and computes the appropriate subset. When the appropriate decompressed tiles are in the RGB cache, the viewer texture maps the visible planes with the tiles.

At each frame, the viewer also applies the stabilization information in the SDL file, using a simple transformation of the current view matrix. SDL also specifies additional user interface elements to display. The viewer monitors the bifurcation points and displays the appropriate overlay as a user approaches. Sample screen shots of the viewer are shown in Figure 11. Figures 11a and 11b are screen shots of a simple home tour, Figures 11c and 11d are of a botanical garden, and Figures 11e and 11f are of an enhanced home tour. The navigation bar at the top of Figures 11e and 11f indicates a branch point: Navigating to the left leads to the dining room, to the right leads to the living room. Figure 11d shows an alternative user interface in which arrows are overlaid on the ground. Figures 11e and 11f show an information bar at the bottom of the screen. We tag certain segments of the environment (*hotspots*) as containing additional information. In Figure 11f, the art case is in view, so icons displayed in the bottom information bar indicate additional information about the art pieces. Selecting an icon via the gamepad causes a high-resolution still image and audio annotation to pop up. This user interface also contains a map, indicating the user's current position within the tour. When users select a different room using the gamepad, they're quickly transported to the new position. We specify these elements' positions and styles using SDL.

Results

We captured three different environments and built interactive experiences around each. A video of the results is available at http://research.microsoft.com/vision/ImageBasedRealities/IBR/default.htm.

Simple indoor environment

Our first demo was a basic home tour (Figures 11a and 11b). The viewer acquired the data along a single loop through the house's ground floor. The final experience contains some dynamic elements such as video playing on the TV and flames in the fireplace. The setup process—path planning and grid placement—took approximately 45 minutes. Acquisition took 80 seconds using our omnidirectional camera mounted on a rolling tripod. Processing the raw data to the final compressed format took several hours.

We used our interactive viewer for the demos described in this section. The viewer lets users experience captured environments. The viewer runs on a 2-GHz Pentium PC with an Nvidia GeForce2 graphics card in full-screen 1,280 × 1,024 mode with a horizontal field of view of about 80 degrees, with the compressed video data streaming from the hard drive. This system lets us maintain a rendering speed of 20 fps during translation and 60 fps during pure rotation. The difference in frame rates occurs because during translation, the system fetches frames from disk, whereas during rotation, the compressed frame is already in memory.

In this first demo, users didn't need to arbitrarily navigate the environment (away from the acquired path) to get a sense of the 3D space. However, they wanted to branch at obvious locations in the house, for example, up the stairs or down a hallway.

Complex outdoor environment

The next environment we captured was an outdoor botanical garden, shown in Figures 11c and 11d. For this more rugged location, we mounted the camera on a helmet, which let us navigate staircases and uneven paths. In this case, we did more advanced planning and captured a large portion of the gardens. The planned path

crossed itself in several locations. At these positions, we placed small markers on the ground to guide the camera operator through these desired intersection points. This process, performed in a single pass, took about 15 minutes. We also collected audio data from various locations in the garden.

Unfortunately, the helmet mount introduced much more jitter in the video. The stabilization technique we described earlier removes much of this unwanted motion. This increased video-processing time, which took about 24 hours unattended. Some manual authoring was also involved: We had to coarsely register video frames to locations on the map and indicate the positions of audio sources relative to that map. Another step of the authoring phase involved roughly indicating the branch points. Our automated bifurcation-selection system then computed the best branch near the user-selected point.

To indicate bifurcations to the user, the viewer rendered arrows into the scene indicating available directions (see Figure 11d). Unfortunately, this method was somewhat confusing as the arrows were sometimes hard to find. The viewer application also rendered a blend of distance-modulated sound sources. Adding sound considerably enhanced the feeling of presence in this virtual environment.

Enhanced home tour

Our final demo was a tour of a high-end home, shown in Figures 11e and 11f. To capture staircases and outside areas, we again used a head-mounted camera. Preparation took about 45 minutes, and mainly involved planning the capture route and again placing small markers. Capture was accomplished in a single 15-minute pass.

For this tour, we also addressed the dynamic range issue by shooting with the omnidirectional camera in HDR video mode as described in the "High Dynamic Range Photography" sidebar. Throughout the tour, users can simultaneously see details inside and outside the home, resulting in much more pleasing imagery than in the first demo, in which window regions were saturated.

To alleviate the confusion around available branch points in this tour, we adopted a heads-up display user interface. As Figure 11e shows, a separate area at the top of the display shows the branch points. Users had an easier time navigating the environment using this interface.

Conclusion

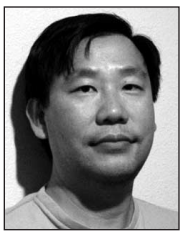
Our system occupies a unique design point compared to previous work in image-based visual tours, making it particularly suitable to delivering interactive high-quality visualization of large real-world environments. In the project's next stage, we plan to reconstruct 3D surfaces and layered models wherever we can do so reliably and without loss of visual fidelity, and to use these models to enable more general movement and more sophisticated interactions and to increase compression efficiency. ■

References

1. F.P. Brooks, "Walkthrough—A Dynamic Graphics System for Simulating Virtual Buildings," *Proc. Workshop Interactive 3D Graphics*, ACM Press, 1986, pp. 9-21.
2. P.E. Debevec, C.J. Taylor, and J. Malik, "Modeling and Rendering Architecture from Photographs: A Hybrid Geometry- and Image-Based Approach," *Proc. Siggraph*, ACM Press, 1996, pp. 11-20.
3. A. Schödl et al., "Video Textures," *Proc. Siggraph*, ACM Press, 2000, pp. 489-498.
4. S. Teller et al., "Calibrated, Registered Images of an Extended Urban Area," *Int'l J. Computer Vision*, vol. 53, no. 1, June 2003, pp. 93-107.
5. E. Chang et al., "Color Filter Array Recovery Using a Threshold-Based Variable Number of Gradients," *SPIE vol. 3650, Sensors, Cameras, and Applications for Digital Photography*, SPIE, Mar. 1999, pp. 36-43.
6. R. Szeliski and H.-Y. Shum, "Creating Full View Panoramic Image Mosaics and Texture-Mapped Models," *Proc. Siggraph 97*, ACM Press, 1997, pp. 251-258.
7. M. Uyttendaele et al., *High-Quality Image-Based Interactive Exploration of Real-World Environments*, tech. report MSR-TR-2003-61, Microsoft Research, Oct. 2003.
8. C. Schmid, R. Mohr, and C. Bauckhage, "Comparing and Evaluating Interest Points," *Proc. 6th Int'l Conf. Computer Vision* (ICCV 98), IEEE CS Press, 1998, pp. 230-235.
9. C. Buehler, A. Bosse, and L. McMillan, "Non-metric Image-Based Rendering for Video Stabilization," *Proc. IEEE Computer Soc. Conf. Computer Vision and Pattern Recognition* (CVPR 01), vol. 2, IEEE CS Press, 2001, pp. 609-614.

Matthew Uyttendaele *is a member of Microsoft Research's Interactive Visual Media Group. His research interests include image-based modeling and rendering, omnidirectional imaging, video compression, and graphics hardware. Uyttendaele has a BS and an MS in electrical engineering from Rensselaer Polytechnic Institute.*

Antonio Criminisi *is a research fellow at Clare Hall College, Cambridge. His research interests include image-based modeling of spaces, image and video analysis and editing, one-to-one teleconferencing, 3D reconstruction from single and multiple images with applications to virtual reality, forensic science, image-based rendering, and art history. Criminisi has a degree in electronics engineering from the University of Palermo and a PhD in computer vision from the University of Oxford.*



Sing Bing Kang *is a researcher at Microsoft Research. His research interests include environment modeling from images. Kang has a PhD in robotics from Carnegie Mellon University.*

Simon Winder *is a member of Microsoft Research's Interactive Visual Media Group. His research interests include image-based rendering and other digital imaging research. Winder has a first-class degree and an MS in electronic engineering and a PhD in computer science from the University of Bath, England.*

Richard Szeliski *is a senior researcher in the Interactive Visual Media Group at Microsoft Research. His research interests include 3D computer vision, video scene analysis, image-based rendering, and constructing photorealistic 3D scene models from multiple images and video. Szeliski has a PhD in computer science from Carnegie Mellon University. He is on the editorial board of the* International Journal of Computer Vision *and is a senior member of the IEEE.*

Richard Hartley *is a professor in the Research School of Information Sciences and Engineering at the Australian National University, and program leader of the Autonomous Systems and Sensor Technology Program of National ICT Australia. His research interests include video analysis, image synthesis, and geometric techniques in computer vision. Hartley has a PhD in mathematics from the University of Toronto.*

Readers may contact Matthew Uyttendaele at Microsoft Research, One Microsoft Way, Redmond, WA 98052-6399; mattu@microsoft.com.