Real-Time Rendering of Real World Environments

David K. McAllister, Lars Nyland, Voicu Popescu, Anselmo Lastra, Chris McCue

University of North Carolina at Chapel Hill, Department of Computer Science[1]

Abstract: One of the most important goals of interactive computer graphics is to allow a user to freely walk around a virtual recreation of a real environment that looks as real as the world around us. But hand-modeling such a virtual environment is inherently limited and acquiring the scene model using devices also presents challenges. Interactively rendering such a detailed model is beyond the limits of current graphics hardware, but image-based approaches can significantly improve the status quo.

We present an end-to-end system for acquiring highly detailed scans of large real world spaces, consisting of forty to eighty million range and color samples, using a digital camera and laser rangefinder. We explain successful techniques to represent these large data sets as image-based models and present contributions to image-based rendering that allow these models to be rendered in real time on existing graphics hardware without sacrificing the high resolution at which the data sets were acquired.

Keywords: image-based rendering, range data, 3D free-form objects, automatic object modeling

1 Introduction

Our goal is to walk freely in an interactively rendered, high-resolution, convincing environment acquired from the real world. We intend the prototype system presented here to be a point of reference from which others and we may advance the same goal. We have implemented an end-to-end image-based system to:

- rapidly and robustly acquire a high resolution image-based scene model,
- process the data as necessary to create a reasonably frugal representation, and
- render the scene with full detail, in real-time, onto a high-resolution display.

Our system acquires scenes consisting of a single large room containing free-form solid objects, intricate surfaces, and complex lighting. We first capture a color and shape representation of the whole scene using a laser rangefinder combined with a high-quality color CCD camera. We place the scanning rig in the scene and acquire a range scan with about ten million samples over 180° horizontally by 76° vertically. We then take a panorama of about ten color images from the same location as the rangefinder. We repeat this process for up to ten manually chosen positions. Each laser range scan takes about 20 to 30 minutes, followed by a few extra minutes to swap the camera and the rangefinder and take the color images.

The raw data must be processed in several steps to prepare a scene description suitable for real-time rendering. We first register each range scan against the others. using a straightforward method in which a user interactively selects three planes that two range scans have in common, and the rigid-body transformation is computed in closed form. This is done for each range scan in turn.

[1] E-mail: {davemc,nyland,popescu,lastra,mccue}@cs.unc.edu

delivered by EUROGRAPHICS DIGITAL LIBRARY www.eg.org diglib.eg.org

We next register each color image with its range scan. Once this rigid-body transformation is known, the range data is projected onto the color image, converting it from scattered data in spherical coordinates to a regular parameterization on the plane, yielding color images with per-pixel depth.

One key aspect of our method is that we represent and process the scene in 2D instead of 3D. Three-dimensional scenes may be represented as a set of projections of the scene onto two-manifolds, such as image planes. This has the important storage and processing advantage that each sample only requires a single coordinate (typically depth) to be stored and computed since the other two coordinates are induced by the parameterization. Were it not for this image-based approach, we would never have been able to process and render scene models consisting of eighty million vertices.

Our depth images require several more phases of processing. We interpolate over small dropouts in the range data and flag larger missing regions (usually caused by absorption or specular reflection of the laser light) as low confidence. We approach the familiar problem of detecting surface discontinuities in a depth image using both a heuristic filter and a novel image reprojection approach.

We now merge the several depth and color images into a single image-based scene model. Again, we use a 2D projection-based approach because of its speed and its easily induced connectivity constraints. We project each depth image onto every other image, using depth comparisons to measure where the images contain the same surface and where they are distinct. For redundant regions we choose the better surface sampling and cull the others. The resulting scene representation is a set of partial depth images whose union is the set of all surfaces scanned from any position.

We render this image-based scene model using several methods – point primitives that approximate splats, triangle meshes with compositing by confidence, and a Voronoi region primitive with novel properties. The renderer runs well on commercial OpenGL-compliant hardware, but can take advantage of special-purpose primitives on PixelFlow [Eyles 1997] to obtain higher quality rendering and better performance.

The data sets shown in this paper are taken from rooms in our building – a reading room, a laboratory and a cluttered office (see Appendix). Each is a single room in which we controlled the lighting. The scenes contain plants as examples of our technique's ability to acquire free-form surfaces if sampled adequately. A major goal of our research is to acquire outdoor scenes as well as indoor. We believe our current system could be used outdoors under strict conditions described in the conclusions.

Following a section on related work, we describe the system in the order in which it is used: data acquisition, processing, and then rendering. We then present performance results and conclude with some observations and future directions.

2 Related Work

Several previous systems have shared our goal of acquiring a model of a real environment for purposes that include rendering novel views. The systems acquire geometry using either computer vision methods or laser range scanning. Of the vision methods, [Sara 1998], [Teller 1998], and [Debevec 1996] use photogrammetry or sparse correspondence to derive shape from camera images. [Laveau 1994] and [Rander 1997] both use dense correspondence to compute depth or disparity. [Laveau 1994], [Teller 1998], and [Debevec 1996] all use significant manual effort to create the model. All these systems yield both color and geometry, usually represented as a coarse polygonal model with texture maps of the surfaces, which are assumed to be diffuse. [Laveau 1994] and [Teller 1998] instead create an image-based model.

The other common method of acquiring an environment's geometry is a scanning laser rangefinder. The system of [El-Hakim 1997] is probably the most similar to our own, with a major difference being that we use far fewer measurement devices – only the laser and one color camera, and depend on these two devices to provide all registration parameters. Range data acquisition systems for environments are commercially available from K^2T, Cyra, and Acuity Research, each of which sells time-of-flight laser rangefinders. All of these have range accuracy in millimeters, however their maximum measurable range varies from 15m to 100m. The Acuity rangefinder is a kit, but Cyra's Cyrax scanner and K^2T's SceneModeler both come as complete products with software to produce 2D and 3D standard format environment models. These two systems do not currently provide color, but only point clouds or simplified polygonal models.

Another class of scanning systems scans objects, rather than whole environments. This distinction will become important in our approaches to several aspects of this work. The most advanced object scanning systems are probably those of [Rushmeier 1998] and [Sato 1997]. Both are able to provide more surface detail than just diffuse color, and both use controlled lighting. [Sato 1997] returns coarse specular surface properties, which is an important step, but since the system acquires shape from silhouette, currently the object must be small enough to fit on a turntable. The [Rushmeier 1998] system cleverly acquires bump maps by controlling the lighting, then simplifies the geometry heavily, leaving the detail in the texture and bump maps. Unlike these two systems, we will make no attempt to derive surface material properties of the many objects in the scenes that we scan since our current goal of acquiring a scene and rendering it exactly as it originally appeared does not require relighting. These two systems as well as commercial object scanning systems from Cyberware, Real3D, and Minolta typically use range image registration systems like [Besl 1992], and important variants such as [Turk 1994], followed by meshing systems like [Curless 1996]. Promising recent work by [Whitaker 1998] robustly generalizes this method, but typically imposes surface smoothness constraints.

We represent and render our scene using an image-based model, rather than the typical simplified polygonal mesh with textures. The Light Field [Levoy 1996] and Lumigraph [Gortler 1996] image-based models consist of a dense set of rays regularly parameterized over four dimensions, typically a 2D grid of 2D images. These models have been acquired for real objects, but not for real environments. When they are, we expect the results to be very pleasing, since they will automatically reproduce view-dependent lighting effects for the original lighting.

Other image-based models represent a scene using sparse images, augmented with camera parameters or range images. These systems store far less data by assuming that surface radiance is view-independent. [Chen 1993] and [Chen 1995] constrain the viewer to positions for which the model can accurately reconstruct a view. [McMillan 1995], [Rademacher 1998] and [Shade 1998] allow any viewpoint to be reconstructed, but exhibit artifacts due to an incomplete model. The latter two systems and ours each use different means of representing a more complete model than the original system.

3 Data Acquisition

To acquire color and range scans of real environments, we have assembled a system consisting of a rangefinder, a high-resolution digital camera and a pan-tilt unit. All of these components are controlled by a PC and were put together on a cart, with the rangefinder placed approximately at eye-level, (see Appendix). We manually

place the scanning rig in locations that attempt to maximize the amount of visible detail. The range and color data have a relatively high resolution of 20 to 25 samples per degree, requiring about 250MB for the raw data from a single 180° x 90° panorama. The total cost of the hardware was less than $35,000. To share our experience in building this acquisition system we will describe each component.

3.1 Scanning, Panning Laser Rangefinder System

The scanning laser rangefinder is an infrared 8mw Acuity Research LIR-4000. It is a time-of-flight (as opposed to triangulation) rangefinder that modulates the laser light to set up an interference pattern, determining the distance from the frequency of the best interference. The rangefinder can measure distances from 0 to 15m at up to 50,000 range readings per second, with precision better than 1cm. A mirror, 45° off-axis, rotates about a shaft with a 2000-position encoder to sweep out a plane.

The pan-tilt unit, a Directed Perception PTU-70-15, has a very clean interface, making programming very simple. The stepping resolution is 14,000 steps in 180°, or about 78 positions per degree, but we typically take data at every third or fourth position. The spinning mirror and panning motor combine to allow the laser to sweep out a longitude-latitude sphere.

The data produced for each rangefinder sample includes range, intensity of the returned laser light, the ambient light subtracted out, and the shaft-encoder position. Using a slower sampling rate and shorter maximum measurable range improves the range data quality, since it can find the proper interference frequency more quickly. We typically acquire 16k to 25k samples per second, with a maximum measurable range of 6m, and the mirror spinning approximately 4 RPS.

During acquisition, raw data is streamed to disk. This consists of the range, the angles of the mirror and panning unit, and the strength of the reflected laser light. The samples are not regular in longitude and latitude, since the rangefinder and scanning mirror are not synchronized.

As with any rangefinder, system calibration has taken considerable effort. For instance, we have determined that the pan-tilt unit includes about 14,039 steps in 180°, not 14,000. The 45° mirror actually has an angle of 44.8°. While these discrepancies may seem small, an error of several centimeters occurs for objects only 3m away. Other values that require calibration are the shaft encoder position of the horizon and the rotation of the pan-tilt unit caused by the weight of the rangefinder. We have experiments to calibrate these factors and we apply corrections in software.

3.2 Digital Camera

In order to capture color imagery, we remove the laser from the pan-tilt unit and replace it with a Canon EOS2000 digital camera (also sold as the Kodak DCS520). The camera is built on a 35mm single-lens reflex body, has a resolution of 1728 by 1152 pixels with up to 12 bits of color data per channel, and an IEEE 1394 (Firewire) interface. We use a Canon 14mm flat-field lens for several reasons. First, the camera's CCD sensor is not as large as a 35mm-film image, so any lens on this camera will have a narrower field of view than on a film camera. Second, since we fix the focus and aperture for all images in the panorama, we require a large depth-of-field. Our typical setting is f/11, allowing us to focus the lens such that everything between 0.5m and infinity is in focus.

We built a mount that makes the camera's nodal point coincide with that of the laser and both devices rotate about this point. This was key to achieving high quality

registration of color and range data (see Section 4.2). We mount the camera vertically and pan the camera to acquire color images covering the field scanned by the laser.

3.3 Camera Calibration

Knowing the camera parameters is crucial for registration of color and range and for rendering with the resulting images. Although the camera parameters could be determined during the registration of the color and range images, the registration is more robust and efficient when the intrinsic camera parameters are determined beforehand. We used the [Tsai 1986] camera calibration algorithm to find the intrinsic parameters of our camera and lens, which is modeled as a simple planar pinhole with one coefficient of radial distortion.

The calibration routine takes in three-space points and their projections on the image plane. The fiducials are the corners of black and white checkerboard squares on a 1 x 2-meter planar grid. We mounted the camera on an optical rail perpendicular to the grid and took images at several positions along the 1-meter rail in order to change the depth of the fiducials, yielding a total of 1460 fiducials.

The fiducial image locations are detected automatically at sub-pixel precision by correlating a single checkerboard square template with the image. The template is divided into four parts by two lines that pass through its center. The slopes of the two lines and the center coordinates are all varied to optimize the correlation.

Since the camera has a wide-angle lens and a high resolution CCD, the radial distortion is significant. The corners of the image are thirty pixels closer to the image center than they would be for a pinhole camera. Before the images are used, they must be undistorted using the radial distortion coefficient determined by calibration.

The measured calibration error was quite small. The 3D locations of the fiducials were projected using the calibrated camera parameters and their distance to the actual image plane locations was computed. The mean distance was 0.66 pixels, with a max of 5.09 pixels. Then the 2D fiducials were projected to three-space using their original three-space depth and the calibration results, and their distance to the actual 3D locations was computed. The mean was 0.62 mm and the max was 2.94 mm. The entire calibration experiment was done twice with almost identical results. This and the small measured error lead us to believe that the camera calibration is quite accurate enough for our application, which is confirmed by the rendering results.

4 Processing

Once the color images and range scans have been acquired they must be converted into a form suitable for efficient storage and image-based rendering. This includes registering all scan positions into a common reference frame, finding each color image's rotation relative to the rangefinder, creating a depth image for each color image based on the range scan, marking surface discontinuities in each depth image and finally culling redundant image portions, yielding a final image-based model.

4.1 Registration of Multiple Positions

Since several scans of the room will be combined into a single model we must register them into a common coordinate system. The registration process will yield a transformation matrix that rigidly translates and rotates a scan to match the others.

We register a pair of scans by manually identifying corresponding planar surfaces in the range data, then use their plane equations to align them. First we map the points from spherical to Cartesian coordinates and then select in each scan three non-parallel

planar surfaces that exist in both. The best-fit plane (in the least squares sense) is calculated for each of the selected surfaces using code from [Eberly 1998]. Once these planes are known, a rotation makes the first plane from each scan parallel to each other, a second rotation makes the other two planes as nearly parallel as is possible, and finally a translation aligns the scans. Since each selected plane includes about 50,000 samples, the best-fit plane is robustly computed and, assuming the three are fairly linearly independent, the resulting transformation makes corresponding planes parallel to within 0.5º. Scene features that are far from the selected planes are sometimes misregistered by several centimeters because of imperfect calibration of the scanning rig. We wanted to evaluate this new approach on indoor scenes, but will probably replace it with a deformation-based registration method.

4.2 Registration of Color and Range Data

The laser rangefinder data is used to provide a depth value for each pixel of the color images, and the resulting color-depth images will be used in rendering. Since the color images were taken with a different device than the range scans, the two must be registered, matching each range sample with the right color pixel. We take care to place the camera's nodal point at the same physical location as the laser's center of rotation, and to rotate both devices about this point. This homographic relationship simplifies the registration to the camera's three rotations relative to the laser. Having no translation greatly increases the quality of our registration and of our final images because it bypasses a projective warp and the associated image resampling.

We use fairly standard image-processing techniques to register each color image with the spherical range image. Simply, we want to find the orientation of the camera image that best correlates with the rangefinder image. Since we have the infrared intensity image, it would seem that we could correlate this directly with the color image (or perhaps with a grayscale representation or just the red channel of the color image). But the illumination of the two images is so different that simple image correlation gives poor results. Specifically, the laser image is illuminated directly by the laser itself; it has no shadows, the entire scene is equally illuminated, and specularities occur in different places than in the normally lit scene.

Instead of correlating the images directly, we correlate edges of the color and range images. The color photograph has high edge strength nearly everywhere in the image because of the high spatial frequency of the natural scene and because of noise in the camera's sensor. In order to find just the salient edges we preprocess the color image using a variable conductance diffusion filter. See [Yoo 1993] for details, but briefly a VCD filter is similar to convolving the image with a gaussian (a standard low-pass filter), but VCD also attenuates each pixel's weight on the pixel at the center of the filter kernel by a gaussian evaluated at their distance in color space[2]. This causes pixels to blend with their neighbors (reducing high frequencies or edge strength), but pixels blend less across sharp boundaries (salient edges) because the boundary in color space is too great. Salient edges act as insulators to the blurring operation, hence the name variable conductance diffusion. Since this operation does not have a closed form solution it is performed iteratively[3]. The total process takes about one minute for a 2Mpixel image.

---

[2] We use RGB, but another color space such as LUV may give better results.

[3] We performed six iterations, with an image space gaussian of st. dev. 2.5 pixels and a color space gaussian of st. dev. 3.5 intensity values (on 0 to 255).

We then apply an edge detector to the VCD-blurred color image and list the edge pixels that had a partial derivative in either axis for any color channel greater than a threshold. We match these edges against the spherical laser image after correcting them for the barrel distortion in the color image (see Section 3.3).

We prepare the spherical range image for correlation by running an edge detector on the laser intensity image and on the range image and storing the combined result as an edge image[4]. To provide a smoother gradient for the image correlation optimization process we convolve the edge image with a kernel that has very broad support (we chose 21 pixels) and a steeply increasing gradient near the center of the kernel. This makes potential solutions have a better error value when they are near the solution (so the optimizer knows it is close), but the best solution has a significantly smaller error than any approximate solution.

The registration proceeds by optimizing the correlation of the edge images. We optimize for the three angles orienting the planar camera image relative to the spherical laser image. Imagine rotating the planar image, looking through each of its edge pixels to sample the corresponding location on the spherical edge image. The objective function is the sum of the squares of the sampled edge strengths. We use a downhill simplex optimization algorithm, but an abundance of local minima force us to restart often, which is done with a variant of simulated annealing in which the algorithm evaluates many solutions and then refines on several of the better solutions. The entire registration process takes about ten minutes for one color image. This optimization algorithm converges given a nearby approximation that can be obtained from the pan-tilt unit's position, or a manual guess. See Appendix for a result.

4.3 Planar Projection

We now create a planar, pinhole image with per-pixel range. This involves two stages. We first undistort the color image to correct for barrel distortion (see Section 3.3), resampling using a bilinear basis function.

Then, given the rotation resulting from each color image's registration process, we are prepared to create the range map for this color image. Note that the data returned from the rangefinder must be filtered before being used because some surfaces do not return accurate values, and noise causes outliers. Also, the rangefinder integrates its sample over a finite aperture, rather than point sampling, due to the continuous mirror rotation during the sampling period. Thus, when sampling a silhouette edge, the returned range value will generally be a point floating in space between the two surfaces. Also, the range samples are not regularly located in longitude and latitude – each has an arbitrary theta and phi. In order to filter the range samples to address all these issues, we first regularize the data on a planar grid. We project the raw, scattered rangefinder samples onto the image plane using a bilinear basis for the support of each splat, and store for each pixel a list of all range samples that touch the pixel and their associated weights. Changing the size of the splat controls the size of dropouts that are filled. We must then properly filter this layered range image, which is a difficult problem. Convolving range images with standard filters creates additional unwanted outlying range values [Shade 1998]. Our approach is to cluster each pixel's list of samples, using a threshold to direct the clustering. The weighted mean of the largest-weight cluster becomes the pixel's range value. This method allows proper

[4] Since the IR image is perfectly registered with the range image, the two can be summed without confusing the resulting edge image.

filtering of range samples in the interior of a surface, but avoids creation of floating samples at silhouette edges because the two sides will fall into different clusters. This method works well on standard, solid surfaces of the type that our scenes include, but is no improvement over current methods for areas of very high range frequency.

We now have the finished range image. We chose to represent the range values as generalized disparity [McMillan 1995], which is the distance from the center-of-projection (COP) to the pixel on the image plane divided by the distance from the COP to the surface sampled. This is a simple conversion from the range value. Range values are inappropriate for rendering because range cannot be projected onto a view plane using a projective transformation, but generalized disparity can.

We also store a flag byte for each pixel with information needed when processing or rendering this pixel. One flag bit marks bad pixels (invalid color value, invalid range value, etc.). We store the color, range, and flag components and the image's camera matrix (described below) in a TIFF file variant. A full-resolution image is approximately 13 MB and takes about three minutes to create.

4.4 Finding Silhouette Edges

One method we use to reconstruct an image from the projected pixels of a source image is to treat the source image as a regular mesh and rasterize the mesh, similar to [Mark 1997]. As they note, when this mesh is projected to other viewpoints it can tear at silhouette edges. It is inappropriate to interpolate across silhouette edges because the interpolated surface does not really exist in the environment. We must detect silhouette edges and mark them using flag bits to avoid rasterizing these false surfaces or "skins". We present two methods for detecting silhouette edges in range images.

Our first approach is similar to a common solution from the range scan literature – reject all polygons that are at grazing angles (cf. [Pulli 1997]). This heuristic works well for solid surfaces. We use a similar, more efficient heuristic, based on range images being locally planar except at silhouettes. To quickly detect planarity we note that generalized disparities of equidistant, collinear pixels increase linearly only when they sample points on a line (or a plane) in 3D. A formal proof is straightforward, but the key is that generalized disparity is inversely proportional to eye-space Z. Simply convolving directional second derivative operators with the generalized disparity map yields a surface planarity map. We mark all pixels with planarity below a threshold as silhouette edges. Since depth and range do not vary linearly in perspective images they are less appropriate for planarity detection than generalized disparity.

Our second method detects only those silhouette edges that are truly skins (false surfaces assumed to exist because we have no data indicating otherwise), versus those surfaces that happen to be viewed very obliquely. A false surface may occlude some real surface seen in another view. Thus, we can detect a mesh triangle as a skin by viewing it from the viewpoint of some other source image and comparing Z buffers to detect occlusions. For example, in Figure 1, to detect the skins in source image **A** we warp it to the viewpoint of source image **B**, computing the projected Z as part of the warp. We reconstruct the projected samples using a regular triangle mesh. We refer to the image resulting from warping **A** to **B** as **A'**. Now, any pixel whose Z value in **B** is greater than in **A'** indicates a skin in image **A** since the Z value of the pixel in **B** is in effect making the statement that space is empty up to this given Z value. But the pixel from **A'** claims to exist in that empty space, so we mark as a skin the mesh triangle in **A** that projected to this pixel. This useful notion of range images making a statement about empty space is used for other purposes in [Curless 1996] and elsewhere.

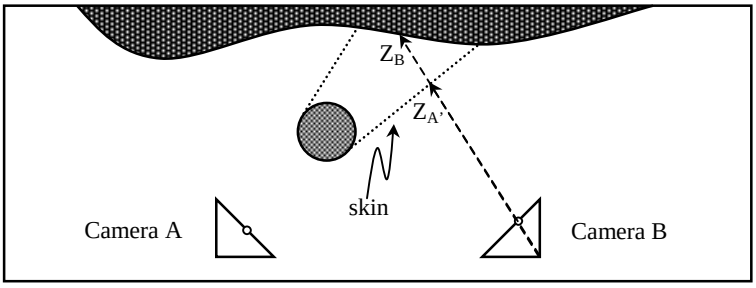


Fig. 1. Finding silhouette edges in image A by projecting it to the viewpoint of image B. Pixels with $Z_B > Z_{A'}$ indicate skins in image A.

The algorithm warps each source image **A** in the model to the point of view of all other images **B**, comparing Z-buffer **B** with the warped Z-buffer **A'**. For the pixels failing the Z test[5], we flag a skin in the source image. This is easily done by rasterizing an item buffer for **A'** indicating which mesh triangle of **A** is visible (see Appendix). Since one skin may occlude another skin, we must repeatedly warp **A** to **B** until a warp can be done with no skins found. We then warp **A** to the next image **B**. The running time is quadratic in the number of images. For our data sets of 40 to 100 images, the whole process takes less than an hour with high-end graphics hardware. This algorithm is well-suited to graphics hardware because the core of the algorithm involves rasterizing triangle meshes and comparing Z buffers.

This method worked well and gave results similar to the planarity heuristic. No skins were marked incorrectly. Some skins were not found because no source image could "see through" that skin. In practice, this works well. If the source images are assumed to represent the region from which a user will see the environment, keeping these skins will prevent the user from ever seeing disocclusion errors that could have been filled using a skin. Another practical value of this reprojection-based silhouette edge finder is that it gives rise to a whole class of range image processing algorithms based on projecting one range image to the point of view of another. This lets model processing and other typically-3D operations be performed in image space where pixel connectivity and occlusion properties are known. [Seitz 1998] includes similar goals, but computes correspondence between pixels in different source images using a cleverly-created explicit voxel representation instead of a reprojection warp.

4.5 Image Tile Primitives

In order to process and render from portions of source images instead of from entire images, we use an image tile primitive. A tile is a higher-order primitive than a pixel sample, but smaller than a whole image. We use tiles for culling redundant imagery from the data set, view-frustum culling, and distributing the work over parallel rendering hardware. Tiles can be any size, and do not all need to be the same size, though our standard size is 32x32 pixels. Tiles need to be large enough that overhead is negligible, but small enough that culling on a per-tile basis is efficient. [Grossman 1998] used image tiles with goals similar to ours, but stored different per-pixel data such as a surface normal.

A tile primitive contains the color, disparity, and flag bits of each pixel, plus the minimum and maximum disparity values and the 4x4 matrix **M** that projects the tile's pixels to their 3D locations. This matrix is the concatenation of three matrices:

[5] We perform a soft Z comparison using glPolygonOffset, to allow for error and discreteness in the images. We then draw the image again using wide lines to account for error in X and Y.

$$\mathbf{M} = \mathbf{R}\,\mathbf{E}\,\mathbf{T} \quad (1)$$

T is the offset of this tile within the full image, allowing pixels to be indexed relative to the tile. **E** is the camera matrix determined in Section 3.3, describing the projection of a pixel to a point in space. **R** is the transformation of this camera pose in world space from the registration process of Section 4.2.

4.6 Culling Redundant Scene Data

Since each surface may appear in multiple source images we select one image to represent each piece of surface, and cull the others from our representation. Our tile culling preprocess loads all source images and evaluates each tile, deciding whether to include it in the final scene description, or cull it. First, the tiles are ranked by their intrinsic quality: We count how many pixels are flagged as skins or as having low-confidence range values. We compute the tile's average range value as a metric of sampling density. And we compute the range variance, as a metric of viewing angle – the lower the variance the less oblique the viewing angle.

We step through the tiles from worst to best, attempting to reject each tile by projecting it to the point of view of each other source image and comparing the two in that image space. As in finding silhouette edges (see Section 4.4), we compare Z values in the space of the second image. For pixels whose projected Z is similar to that stored in the image (which we sample bilinearly), the two pixels sample the same surface. For these pixels we compare their quality (their range and viewing angle), and keep the higher quality pixel. By adding a threshold when comparing, we vary how aggressively the tiles are culled. After projecting a tile to all source views, if all its pixels have been rejected, we can cull the entire tile. We believe that this approach to combining multiple source images into a scene description has great potential that we have not fully explored. One of its advantages is that it is non-redundant, similar to a Layered Depth Image [Shade 1998], and yet is constructed without resampling the pinhole camera source images, unlike an LDI. The rendering efficiency is basically equivalent to that of an LDI.

5 Rendering

A simple interactive application allows the user to explore the image-based environment. The application is implemented using OpenGL and runs on a Silicon Graphics Onyx2 with InfiniteReality2 graphics, or on PixelFlow [Eyles 1997]. With smaller data sets, the application also runs on a PC.

The application's rendering primitive is an image tile. To render a tile we first concatenate the tile's projection matrix **M** (see Section 4.5) with the OpenGL model-view matrix **V** and projection matrix **P**.

$$\mathbf{C} = \mathbf{P}\,\mathbf{V}\,\mathbf{M} \quad (2)$$

The matrix **C** maps points in the image tile of the form $\langle x, y, 1, d(x,y)\rangle$ to points in homogeneous space. $d(x,y)$ is the generalized disparity at point x,y.

Using this transformation matrix, the program attempts to trivially reject tiles that project entirely off-screen. The extrema of the x, y, and disparity dimensions of the tile form a bounding volume, which is projected and clipped against the screen. View frustum culling simply rejects the entirely off-screen tiles. From typical viewpoints, approximately 65% of all tiles are rejected at this stage. Using a standard-sized tile, the cost of the culling test is less than 1% of the cost of projecting all the tile's pixels.

On PixelFlow, we wrote a custom primitive [Olano 1998] to incrementally transform the *x*, *y*, and *z* columns of the matrix, so each pixel requires only the multiplication with the disparity coordinate, which varies arbitrarily from pixel to pixel [McMillan 1995]. This incremental transformation reduces the cost to one fourth of the multiplication operations of a standard vertex. Converting from NDC space to screen space using the OpenGL viewport and depth range transformations [Neider 1993] finishes the transformation. Note that this transformation yields a screen-space Z value, which will be used in compositing.

One problem in McMillan-style image-based rendering is correctly resolving visibility when projecting multiple source images into a single output image. The occlusion-compatible image traversal order allows any individual image to resolve self-occlusions but provides no means of compositing multiple source images. An inverse warping renderer [McMillan 1997] addresses this problem by searching all source images to find the front-most surface at each pixel, as in ray tracing, but incurs a cost penalty. The Layered Depth Image [Shade 1998] resolves visibility *a priori* by resampling onto an intermediate layered image [Max 1996] and then using the occlusion-compatible order. Our approach is to use standard Z-buffer composition to resolve visibility. This allows compositing of standard geometric primitives with image-based primitives. Z-buffering hardware is ubiquitous and the additional cost of computing Z at each sample is minimal.

Once the source image samples have been projected to the screen, the last remaining issue is reconstructing an image from this scattered data. For this interactive system, we have explored several reconstruction strategies.

5.1 Point-Based Reconstruction

Our simplest reconstruction method is to render each sample using GL_POINTS. On the SGI IR2 this is fairly fast if the primitives are placed in display lists. However, the reconstruction quality is worse than rendering as triangles because we have poor control over each sample's footprint. Also, most graphics hardware does not properly accelerate point primitives although it is becoming increasingly important to do so.

5.2 Triangle Mesh Reconstruction

An image tile can be treated as a regular mesh, with the tile pixel positions and range values dictating the mesh vertex positions, and the pixel colors being used as the vertex colors [Mark 1997]. Each square of the mesh can be interpolated using a bilinear patch or simply using two triangles. Many people have used this approach to render height fields on graphics hardware [Taylor 1993].

Our mesh never exists explicitly as 3D geometric primitives, but is projected from the source-image tile directly to the screen. The pixel colors are linearly interpolated across the mesh. When the tile pixels being interpolated are samples of a single continuous surface, this method works quite well and provides excellent reconstruction quality. Reconstruction algorithms for image-based rendering often exhibit gaps when the source image pixels project sparsely onto the screen because the algorithms use a splat size with too small a support for the given projection. Reconstructing by linearly interpolating adjacent samples as we do here avoids these gaps. But in order to avoid interpolating across disocclusions, we check the skin bits of each sample (see Section 4.4) and cull mesh triangles that cross silhouette edges.

5.3 Compositing by Confidence

Since many different source images may view a given surface in the scene, we cull the redundant image area as a preprocess, but some overlaps remain to prevent seams. These overlap regions cannot be satisfactorily resolved by the Z composite because, being images of the same surface, they have nearly identical Z values. We want to composite in the way that gives the best results in regions of overlap. An alpha composite is a common choice here (cf. [Pulli 1997]), but the results are often blurry.

We implemented on PixelFlow a quality-based compositing operation similar to the confidence compositing in [Mark 1997]. Our method primarily composites by Z, but when two instances of a surface exist, the better-sampled surface remains visible. Our sampling quality metric is the screen-space density of the projected samples – the denser the sampling, the higher the resolution of the rendered frame. We measure sampling density for each triangle to be rasterized as the triangle's area, which is already computed as part of triangle rasterization. To implement this compositing efficiently we scale the triangle's area and add it to the Z value of each vertex, yielding Z'. When rasterizing the triangle we interpolate Z' instead of Z and perform the depth composite on Z' values. This takes no more work per pixel than depth compositing and required no change to our graphics hardware. This method is also order-independent, unlike alpha compositing. The result of this composite is that larger triangles (worse-sampled surfaces for this projection) will be pushed back more in Z than smaller triangles and so will lose to denser samplings of the same surface.

5.4 Quadratic Splat Primitive

When samples project sparsely onto the screen (less than one source sample per screen pixel) the continuous interpolation provided by triangle mesh reconstruction is a significant quality advantage over other reconstruction methods. However, in many systems like ours the samples often project more densely than one source sample per screen pixel. Reconstructing as triangles is inefficient since their area is often smaller than half a pixel. Both the transformation and rasterization work are overkill. A geometrically simpler primitive such as a point would alleviate this inefficiency [Levoy 1985; Grossman 1998], and would be applicable whenever the screen density of primitives approximates or exceeds the screen's pixel density (as in high-resolution image-based or volume rendering).

We have developed and implemented on PixelFlow a primitive that reconstructs the image from the projected samples by rasterizing each sample as a solid-colored circle that has a quadratic falloff in depth. One of these quadratic splats can be rasterized in less than one third the clock cycles needed to rasterize two triangles per sample. The falloff in depth normally causes each pixel to receive the color of the closest sample. This is similar to the technique of rendering Voronoi regions by rasterizing a cone for each sample [Haeberli 1990], which [Larson 1998] used for reconstruction in image-based rendering. Unlike cones for Voronoi regions, our splat centers do not all have the same Z value. Rather, they have the sample's Z value at the center and increase quadratically toward the circle's edge. This allows compositing for visibility and reconstruction to both be done with a single Z composite.

Figure 2 shows an example of reconstructing samples from two images using the quadratic splat primitive. Note that both source images contribute to the final image, with source image 1 having a greater contribution because it is better sampled.

The primitive's quadratic shape approximates the central lobe of a gaussian splat's alpha component, as used in volume rendering [Westover 1990] but the

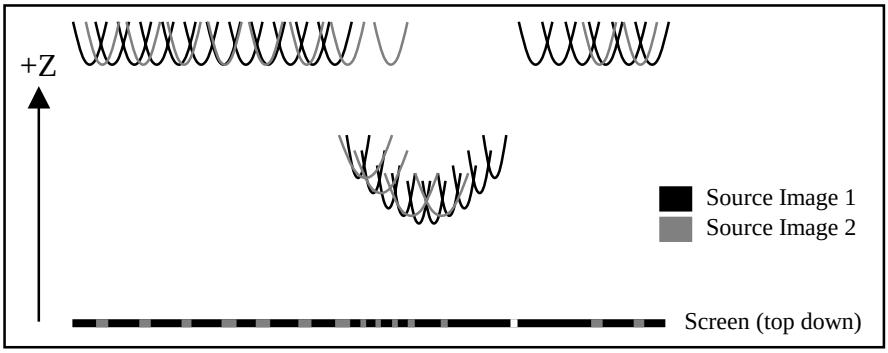


Fig. 2. The results of compositing two images reconstructed using quadratic splat primitives, shown as a top-down view of screen space.

primitives can be rendered in any order, instead of sorting them back to front or vice-versa. The reconstruction of the scattered samples is piecewise-constant, but since the samples will usually project more densely than the screen pixels and since we supersample the screen [Molnar 1991], the flat-color regions are properly filtered to prevent artifacts of piecewise-constant reconstruction like aliasing and Mach banding.

Our quadratic splat primitive, implemented on an edge equation-based rasterizing system, is actually a much simpler primitive than even a single triangle. Other edge equation-based systems like InfiniteReality [Montrym 1997] could probably implement circle or quadratic splat primitives with very little added hardware.

Samples in the interior of a surface tend to have Voronoi region-shaped footprints, but to avoid artifacts at silhouette edges we must bound the radius of the circle. If the radius is too large, silhouette edges will look scalloped. If the radius is too small, gaps will appear in the surfaces. The correct radius is a function of the projected area of a source image pixel. Since the projection of a source image pixel will be an ellipse, but our splats are only circles, the radius must be approximate.

Our approximation is based on the average scaling of the composite projection matrix **C** from Equation 2. We compute this as the cube root of the determinant of the 3x3 principal submatrix of **C**, times the average scaling of the viewport transformation (computed similarly), divided by the *w* coordinate of the projected sample. Since only the *w* coordinate varies per sample, we can compute each sample's projected area with just one multiply. [McMillan 1997] suggests a similar method that is less efficient but yields elliptical splats. [Shade 1998] uses one multiply and a lookup table that is recomputed each frame. Their method has the advantage of taking the sample's normal into account, but it is unsuitable for rendering from tiles because it would take much longer to compute the lookup table than to render the entire tile.

6 Results

Rendering performance is acceptable, but not quite what we are hoping for. We timed an 800-frame sequence with each of the four rendering methods. It averaged 549 visible tiles (1024 pixels per tile) per frame. We always render at 1920x1080 resolution, with four samples per pixel on PixelFlow and one on InfiniteReality2. On the IR2 the average frame rate was 12.14 frames per second for points and 6.21 for triangle meshes. On a sixteen-node PixelFlow, the average frame rate was 8.47 frames per second for confidence-composited triangles and 12.43 for quadratic splats. For the

triangle mesh rasterizers, this is more than 1.1M triangles per frame – after view frustum culling. PixelFlow works best as a retained-mode machine, since the high-speed interface has not been completed. This limits the size of the data sets we can view on that machine. We plan to double the size of PixelFlow, but would prefer to be able to transfer data on demand as the viewpoint changes.

The rendering quality is quite good, with the quadratic splat yielding the best results (see Appendix). We plan to improve this reconstruction method by including compositing by confidence. Acquired color and geometry represented at full resolution is significantly more detailed than an equivalent synthetic model, and we believe that the rendering results are more realistic. Details such as crooked or sagging ceiling tiles, clutter on desks, spills on the floor, and cracks in leather chairs are intractable to model by hand, but can be had automatically using an acquisition system.

The data from the laser are excellent, except for problems with specular or very dark surfaces. The spatial resolution of the range data is as high as we need at this time, but some small calibration errors remain. Registration of multiple range scans using planes works well on indoor scenes, but barring perfect rangefinder calibration, we will have to deform the scans to make them align. We intend to implement a deformation based on the calibration parameters of the rangefinder, rather than a generic three-space deformation like [Curless 1996].

Registration of color and range usually works but occasionally the optimization does not converge without manually adjusting the starting parameters. Our plans for the future include building a device that captures color and range from the same location, at the same time, thus eliminating the need for color/range registration.

7 Conclusions

Through the creation of this system we learned and observed several things that are worth sharing. Assembling the laser rangefinder and writing the software was much more difficult than we expected. It has taken a man-year to get good results, and we are still fighting subtle calibration errors. We compensate in software for some of the hardware's weaknesses such as missing range data, but more work is required.

The area of image-based model representations for large environments requires additional research. Most systems represent an environment as a simplified polygonal mesh with textures. This is adequate for scenes without much detail in the geometry, but is less appropriate for scenes with geometric detail like plants, clutter, and people. Our approach preserves the sampled geometric detail by using an image-based model, but the model is represented in terms of the images that acquired the model. The only benefits of this are that it is easy to implement, does not require resampling of the images, and is more compact and efficient to render than an equivalent polygon model. We would prefer a representation that unifies the model and fully implies the connectivity while still being compact to store and efficient to render by using an incremental transformation. Our results also indicate that capturing directional exitant radiance and representing it in the model will yield more satisfying images, although this is not the system's primary weakness right now.

One of our major goals is to acquire outdoor scenes. Some obstacles can be overcome, but some are difficult to solve, regardless of technology. It is difficult to keep outdoor scenes static when wind blows and people walk through the scene. The motion of the sun, even over just 15 minutes, leads to difficulties in acquiring consistent color. Color acquisition must become quicker, or perhaps scenes could be

acquired on overcast days or at night. But the acquisition of multiple panoramas of a single scene will always require that they be taken one at a time to avoid seeing the acquisition devices, unless the devices become so small as to be unobtrusive.

The most significant performance bottleneck we encountered in this research was the rendering. We would like to see faster and more flexible point primitives in graphics hardware, such as points whose size is defined in model space instead of screen space and points with an application-defined alpha or depth falloff. Much experimentation remains to be done in this area. We are investigating special-purpose hardware for image-based rendering, but hopefully this work shows the advantage of certain small changes to existing graphics hardware.

We would like to thank Kurtis Keller and John Thomas for their help building the scanning rig, Chun Fa Chang and Chris Wynn for early software work on the PixelFlow application, and the UNC IBR group for assistance and ideas. We are grateful for funding from Integrated Device Technologies Corp., DARPA under order #E278 and NSF under grant #MIP-961. We are grateful for equipment donations from Intel Corp. and Hewlett-Packard Corp.

8 References

Besl, P. J. and N. D. McKay. A Method for Registration of 3-D Shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **14**(2): 239-256, 1992.

Chen, E. QuickTime VR - An Image-Based Approach to Virtual Environment Navigation. *Proc. of SIGGRAPH '95*, Los Angeles, CA, August, 1995.

Chen, E. and L. Williams. View Interpolation for Image Synthesis. *Proc. of SIGGRAPH '93*, Anaheim, CA, August, 1993.

Curless, B. and M. Levoy. A volumetric method for building complex models from range images. *Proc. of SIGGRAPH '96*, New Orleans, LA, August, 1996.

Debevec, P. E., C. J. Taylor, et al. Modeling and Rendering Architecture from Photographs. *Proc. of SIGGRAPH '96*, New Orleans, LA, August, 1996.

Eberly, D. MAGIC: My Alternate Graphics and Image Code. *http://www.magic-software.com*, 1998.

El-Hakim, S. F., P. Boulanger, et al. Sensor Based Creation of Indoor Virtual Environment Models. *Intl. Conf. on Virtual Systems and MultiMedia - VSMM'97*, Geneva, Switzerland, September, 1997.

Eyles, J., S. Molnar, et al. PixelFlow: The Realization. *SIGGRAPH/Eurographics Workshop on Graphics Hardware*, Los Angeles, CA, August, 1997.

Gortler, S. J., R. Grzeszczuk, et al. The Lumigraph. *Proc. of SIGGRAPH '96*, New Orleans, LA, August, 1996.

Grossman, J. P. and W. Dally. Point Sample Rendering. *Proc. of Eurographics Workshop on Rendering*, Vienna, Austria, June, 1998.

Haeberli, P. Paint By Numbers: Abstract Image Representations. *Proc. of SIGGRAPH '90*, Dallas, TX, August, 1990.

Larson, G. W. The Holodeck: A Parallel Ray-caching Rendering System. *Proc. of Eurographics Workshop on Parallel Graphics and Visualisation*, September, 1998.

Laveau, S. and O. Faugeras. 3-D Scene Representation as a Collection of Images and Fundamental Matrices. *Proc. of Intl. Conf. on Pattern Recognition*, Jerusalem, Israel, 1994.

Levoy, M. and P. Hanrahan. Light field rendering. *Proc. of SIGGRAPH '96*, New Orleans, LA, August, 1996.

Levoy, M. and T. Whitted. The Use of Points as a Display Primitive. Univ. of North Carolina, Computer Science, Technical Report 85-022, 1985.

Mark, B., L. McMillan, et al. Post-Rendering 3D Warping. *Proc. of Symposium on Interactive 3D Graphics*, Providence, RI, April, 1997.

Max, N. Hierarchical Rendering of Trees from Precomputed Multi-Layer Z-Buffers. *Proc. of Eurographics Workshop on Rendering*, Porto, Portugal, June, 1996.

McMillan, L. An Image-based Approach to Three-Dimensional Computer Graphics. Univ. of North Carolina, Computer Science, Ph.D. Dissertation, 1997.

McMillan, L. and G. Bishop. Plenoptic Modeling: An Image-Based Rendering System. *Proc. of SIGGRAPH '95*, Los Angeles, CA, August, 1995.

Molnar, S. Efficient Supersampling Antialisasing for High-Performance Architectures. Univ. of North Carolina, Computer Science, Technical Report 91-023, 1991.

Montrym, J. S., D. R. Baum, et al. InifiniteReality: A Real-Time Graphics System. *Proc. of SIGGRAPH '97*, Los Angeles, CA, August, 1997.

Neider, J., T. Davis, et al. OpenGL Programming Guide, Addison Wesley, 1993.

Olano, M. and A. Lastra. A Shading Language on Graphics Hardware: The PixelFlow Shading System. *Proc. of SIGGRAPH '98*, Orlando, FL, July, 1998.

Pulli, K., M. Cohen, et al. View-based Rendering: Visualizing Real Objects from Scanned Range and Color Data. *Proc. of Eurographics Workshop on Rendering*, St. Etienne, France, June, 1997.

Pulli, K., T. Duchamp, et al. Robust Meshes From Multiple Range Maps. *Intl. Conf. on Recent Advances in 3-D Digital Imaging and Modeling*, Ottowa, ON, 1997.

Rademacher, P. and G. Bishop. Multiple-Center-of-Projection Images. *Proc. of SIGGRAPH '98*, Orlando, FL, July, 1998.

Rander, P., P. J. Narayanan, et al. Virtualized Reality: Constructing Time-Varying Virtual Worlds from Real World Events. *Proc. of IEEE Visualization `97*, Phoenix, AZ, October, 1997.

Rushmeier, H., F. Bernardini, et al. Acquiring Input for Rendering at Appropriate Levels of Detail: Dgitizing a Pieta. *Proc. of Eurographics Workshop on Rendering*, June, 1998.

Sara, R., R. Bajcsy, et al. 3-D Data Acquisition and Interpretation for Virtual Reality and Telepresence. *Proc. of IEEE Workshop on Computer Vision for Virtual Reality Based Human Communications*, Bombay, India, January, 1998.

Sato, Y., M. D. Wheeler, et al. Object Shape and Reflectance Modeling From Observation. *Proc. of SIGGRAPH '97*, Los Angeles, FL, July, 1997.

Seitz, S. M. and K. N. Kutulakos. Plenoptic Image Editing. *Proc. of ICCV'98*, 1998.

Shade, J., S. Gortler, et al. Layered Depth Images. *Proc. of SIGGRAPH'98*, Orlando, FL, August, 1998.

Taylor, R. M., W. Robinett, et al. The Nanomanipulator: A Virtual-Reality Interface for a Scanning Tunneling Microscope. *Proc. of SIGGRAPH '93*, Anaheim, CA, August, 1993.

Teller, S. MIT City Scanning Project. *http://graphics.lcs.mit.edu/city/city.html*, 1998.

Tsai, R. Y. An Efficient and Accurate Camera Calibration Technique for 3D Machine Vision. *Proc. of IEEE Conf. on Computer Vision and Pattern Recognition*, Miami Beach, FL, 1986.

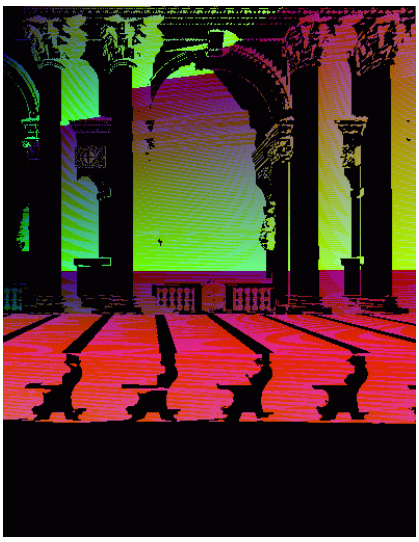
Turk, G. and M. Levoy. Zippered Polygon Meshes from Range Images. *Proc. of SIGGRAPH '94*, Orlando, FL, August, 1994.

Westover, L. Footprint Evaluation for Volume Rendering. *Proc. of SIGGRAPH '90*, Dallas, TX, August, 1990.

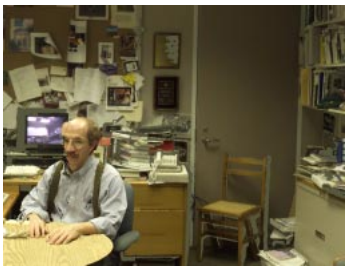
Whitaker, R. T. A Level-Set Approach to 3D Reconstruction from Range Data. *Intl. Journal of Computer Vision* **29**(3): 203-231, 1998.

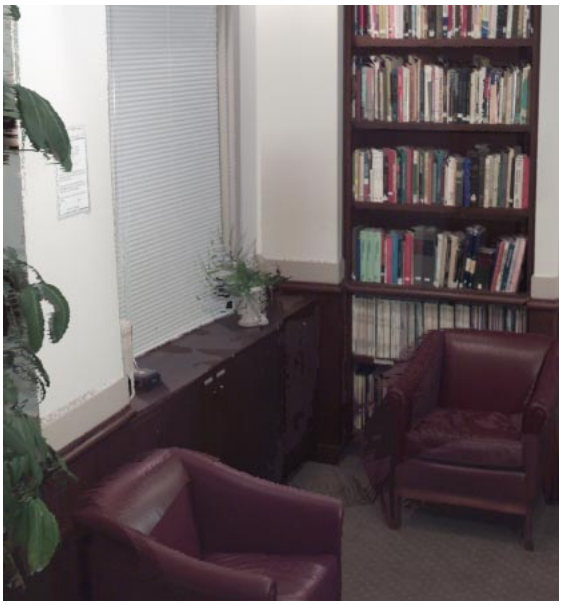
Yoo, T. S. and J. M. Coggins. Using Statistical Pattern Recognition Techniques to Control Variable Conductance Diffusion. Information Processing in Medical Imaging (Lecture Notes in Computer Science 687). H. H. Barrett and A. F. Gmitro. Berlin, Springer-Verlag**:** 459-471, 1993.





Left: The scanning rig used to acquire environments. **Middle: A r**ange scan registered with a color image. White lines are silhouette edges. **Right:** An item buffer resulting from warping a source image to the viewpoint of another source image with Z comparison. (McAllister, et al.)





Upper Left: An environment made from two scans – one with Henry, one without.
Lower Left: A plant scanned from four positions, composited using quadratic splats.
Right: Library with plants and leather chairs, composited from two scans. (McAllister, et al.)