Interactive exploration of acquired 3D data[*]

Lars Nyland, David K. McAllister, Voicu Popescu, Chris McCue and Anselmo Lastra[**]
University of North Carolina, Chapel Hill, NC 27599-3175

ABSTRACT

The goal of our image-based rendering group is to accurately render scenes acquired from the real world. To achieve this goal, we capture scene data by taking 3D panoramic photographs from multiple locations and merge the acquired data into a single model from which real-time 3D rendering can be performed. In this paper, we describe our acquisition hardware and rendering system that seeks to achieve this goal, with particular emphasis on the techniques used to support interactive exploration.

Keywords: Image-based rendering, range acquisition, depth maps, interactive 3D graphics.

1. INTRODUCTION

Image-based rendering is a new rendering technique [McMillan and Bishop 1995; Gortler, Grzeszczuk et al. 1996; Levoy and Hanrahan 1996; McMillan 1997; Shade, Gortler et al. 1998] that attempts to achieve two goals: first, the renderings are realistic since they are rendered from acquired photographs; and second, the rendering complexity is less than standard polygon rendering. In order to understand the advantages and pitfalls of this technique more fully, we have developed a hardware system that acquires images with depth and an interactive rendering system (using standard OpenGL hardware support for speed).

Our data-acquisition system, built from commercially available components, captures range and color data of scenes. The acquisition system uses a time-of-flight laser rangefinder and a high-resolution digital color camera. Once the data are captured, we perform significant analysis to fuse the color and range information into a single dataset. Samples are taken every milliradian, creating panoramas with several million samples.

Our visualization software (initially described in [McAllister, Nyland et al. 1999]) allows interactive exploration of the data, rendering it from the 3D image-based model using commercial OpenGL graphics accelerators or our custom graphics engine, PixelFlow [Molnar, Eyles et al. 1992; Eyles, Molnar et al. 1997]. The viewer is allowed to move about freely through the data, examining it from any possible point of view, achieving a very realistic sense of the captured environment. In order to render solid surfaces from point sample data, we have developed a special graphics primitive that properly renders solid surfaces represented as points.

We acquire panoramas from multiple locations in order to fill in occluded regions and improve poorly sampled surfaces. The panoramas are registered and fused into a single model with software-assisted and/or automated methods. Depending on the complexity of the environment, only a small number of viewpoints are needed to fill in the missing data, creating a model that includes most of the useful geometry in a scene.

The remainder of the paper first shows and describes some sample views from our renderer along with the input used to generate the images. The details of the acquisition system are described, followed by a description of the processing required to build a coherent model. A description of the details of the renderer is presented, explaining how to generate high-quality images from the data. We survey some potential uses of such data, followed by some conclusions and goals for future work.

2. EXPLORING A SCENE USING 3D PHOTOGRAPHS

Before delving into the technical details, we present some of the results of our work. Color Plate 1 shows a series of renderings captured while moving about in the reading-room dataset. The image series is rendered from two panoramas acquired at the scene. The real-time renderings give the freedom to move about a scene after the fact, for a better understanding of the visibility, spatial relationships, and distances therein. It also demonstrates the realistic quality of the rendering.

[*] For a full-color version of this paper, larger color images, quicktime movies and more, visit our website `http://www.cs.unc.edu/~ibr/`

[**] Email: {nyland, davemc, popescu, mccue, or lastra}@cs.unc.edu

Color Plate 2 shows range and color panoramas for a recently acquired outdoor mock car accident. This dataset is meant to illustrate an application in accident reconstruction. We set up a simulated accident scene as if the Alfa Romeo had struck the Mercedes Benz. Various car parts were also scattered about the site. The data were captured at night, using auxiliary lights, to avoid the difficult problem of lighting changes (we discuss this in Section 3.6).

We collected data from three places, both sides of the impact point as well as from the rear of the Alfa. Of course, the requirements of the application will guide the data collection process. As we discuss in Section 3.5, we foresee more automated collection. However, a complex scene such as this will likely require the operator to decide locations from which to capture the 3D photographs, as with standard photography.

3. 3D SCENE ACQUISITION

3.1 Hardware

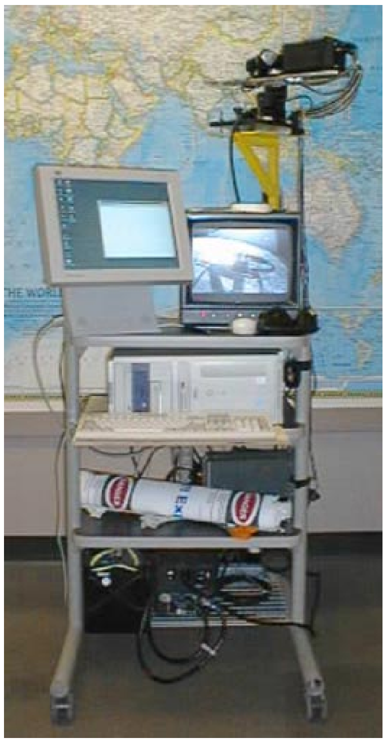
To acquire color and range scans of real environments, we have assembled a system consisting of a rangefinder, a high-resolution digital camera and a pan-tilt unit. All of these components are controlled by a PC and were put together on a cart, with the rangefinder placed approximately at eye-level. We manually place the scanning rig in locations that attempt to maximize the amount of visible detail of the scene. 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.

3.2 Scanning, Panning Laser Rangefinder

The scanning laser rangefinder is an Acuity Research LIR-4000. It is a time-of-flight rangefinder that modulates an 8mw infrared laser to set up an interference pattern, determining the distance from the frequency with the strongest 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 4096-position encoder to sweep out a plane.

The pan-tilt unit is a Directed Perception PTU-70-15. The positioning resolution is 14,000 steps in 180°, or about 78 positions per degree, but we typically take data at every third or fourth position (again, 20 to 25 samples per degree). The spinning mirror and panning motor combine to allow the laser to sweep out a longitude-latitude sphere.

Figure 1. Data acquisition cart. The rangefinder and pan-tilt unit (upper right), plus a PC, power supplies, control boxes, and batteries.



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

During acquisition, raw data are 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 most hardware systems, calibration has taken considerable effort. For instance, we have determined that the pan-tilt unit actually moves 14,039 steps in 180°, not 14,000. The 45° mirror actually has an angle of 44.8°. While these discrepancies may seem small, they lead to an error of several centimeters for objects only 3m away. Other values that require calibration are the shaft encoder position of the horizon and the distance to the spinning mirror. We have experiments to calibrate these factors and we apply corrections in software.

3.3 Digital Color 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 from 0.5m to ∞ is in focus.

We built a mounting bracket that makes the camera's nodal point coincide with that of the laser and both devices rotate about this point. This was essential to achieving high quality registration of color and range data. We mount the camera vertically and pan the camera to acquire color images covering the field scanned by the laser.

3.4 Camera Calibration

The camera parameters are 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 camera calibration algorithm [Tsai 1986] 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 maximum 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 error was 0.62 mm and the maximum 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 accurate enough for our application, which is confirmed by the rendering results.

3.5 Data Collection Process

Data collection consists of planning and acquisition. Planning is currently done using ad hoc methods where we examine the scene and attempt to place the device with several goals in mind. The first is to get good sampling in regions of high interest. This is done by placing the acquisition system close to those regions. The second goal is to reduce the occluded areas. We try to place the cart such that the concave regions are viewed in at least one of the panoramas acquired. The third goal is coverage. Our panoramas are typically between 180° and 270° (although recently we've become aware of the value of 360° panoramas with regard to parameterization of the camera [Szeliski and Shum 1997]).

Among our future goals is automated planning of scanner locations. As each panorama is acquired, we would like the system to hypothesize about occluded regions and full coverage in choosing the next location for a scan of the scene, probably taking into account the accessibility constraints of the size of the cart. We foresee the planning following one of two very different strategies: one where the total number of scans is known ahead of time, and a more greedy, but simpler scheme that tries to maximize the goals as much as possible in each scan.

Such an automated process may not be the best to use in a problem domain where (a) there are many complex occlusions, and (b) where a human may best decide what is of interest. An example is our mock accident scene. There an investigator is likely to know what features are of interest for reconstruction of the events, and so would choose sampling locations to capture those features.

3.6 Outdoor Scenes

Collection of datasets outdoors presents some special and difficult problems because we can't control the outdoor scene as well as we can an indoor scene. First, the geometry of the scene may change because of external forces, such as wind and rain. Second, the lighting changes as the day progresses, and the sun moves across the sky.

The changes in geometry are especially severe in plants. For some scenes, such as our mock car crash, changes in the positions of leaves are unlikely to cause problems. However, for visual simulation in jungles or forests, blurry plants will not be acceptable. It will require further research to find a solution for this.

Since it may take one or two hours to collect data, the lighting will chance substantially. This is also a problem for photogrammetric data collection. It’s conceivable that one could correct for lighting changes over the course of the day, especially since the rangefinder is collecting the geometry of the environment. However, it’s likely to be an extremely difficult problem because of inter-reflections, unknown surface properties, etc. Two interim ways to avoid the problem are to collect data at night, like we did, or on a cloudy day. Again, if the objective is not to render a realistic view of the scene, but rather to solve a scientific problem, such as accident analysis, lighting changes are less of a problem.

4. DATA PROCESSING

Once the color images and range scans have been acquired, they must be converted into an efficient representation for 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 translates and rotates a scan to match the others.

We currently register scans with a software-assisted method, although we are exploring automatic techniques. To register a pair of scans, we identify corresponding planar surfaces in the range data, then use their plane equations to align them. First, we select three non-parallel planar surfaces in each scan that exist in both. The best-fit plane 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 5,000 - 50,000 samples, the best-fit plane is robustly computed and, assuming the three are 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.

4.2 Registration of Color and Range Data

The laser rangefinder data are used to provide a depth value for each pixel of the color images, and the resulting color-depth images, or *3D photographs*, 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 standard image-processing techniques to register each planar 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). However, 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 photographs have surprisingly 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 [Yoo and Coggins 1993]. Briefly, a VCD filter is similar to convolving the image with a gaussian kernel (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[1]. 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 difference 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[2]. The total process takes about one minute for a 2Mpixel image.

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

2 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 (of 256).

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 radial distortion in the color image.

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[3]. 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 two 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.

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 radial distortion, resampling using a bilinear basis function.

Then, given the rotation resulting from each color image's registration process, we prepare a range map for each undistorted 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. In addition, 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 data 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, Gortler et al. 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 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. We now have the finished range image.

We choose to represent the range values as generalized disparity [McMillan and Bishop 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 in a TIFF file variant. A full-resolution image is approximately 13 MB and takes about two 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 and Bishop 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, Cohen et al. 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

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

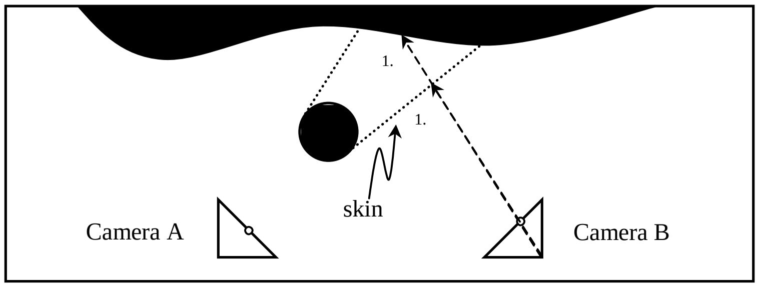


Figure 2. Finding silhouette edges in image A by projecting it to the viewpoint of image B. Pixels with $Z_B > Z_{A'}$ indicate skins in image A.

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 determine if a mesh triangle is a skin by viewing it from the viewpoint of some other source image and comparing Z buffers to detect occlusions. For example, in figure 2, 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'**. 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. A very useful notion of range images is that they not only tell the distance to objects in the scene, but they also indicate that there is nothing else in between the sensor and the object. This is used in [Curless and Levoy 1996] and elsewhere.

The skin-detection 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[4], 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. 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 on our Oynx2. This algorithm is well-suited to graphics hardware because the core of the algorithm involves rasterizing triangle meshes and comparing Z buffers.

This method gives 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. 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. Model processing (and other typical 3D operations) can be performed in image space where pixel connectivity and occlusion properties are known. In [Seitz and Kutulakos 1998], similar goals are outlined, but correspondence between pixels in different source images is used to cleverly create explicit voxel representation of space 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 less complex 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. In [Grossman and Dally 1998], image tiles are used with goals similar to ours, but different per-pixel data are stored (e.g., surface normals).

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:

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

4 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.

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 4.2, 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 4.1.

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. The tile culling preprocess loads all source images and evaluates each tile, deciding whether to include it in the final scene description. The tiles are ranked by their intrinsic quality. This is done by counting how many pixels are flagged as skins or as having low-confidence range values, by examining the tile's average range value as a metric of sampling density, and by computing the range variance as a metric of viewing angle – the lower the variance the less oblique the viewing angle.

The tiles are examined 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, 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, Gortler et al. 1998], and yet is constructed without resampling the pinhole camera source images, unlike an LDI. The rendering efficiency is 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, Molnar et al. 1997]. With smaller data sets, the application also runs on OpenGL-accelerated PC's.

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

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

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 and Lastra 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 and Bishop 1995]. This incremental transformation reduces the cost to one fourth of the multiplication operations of a standard vertex. The transformation is finished by converting from NDC space to screen space using the OpenGL viewport and depth range transformations [Neider, Davis et al. 1993]. 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, Gortler et al. 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 these 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. In addition, 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 and Bishop 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, Robinett et al. 1993].

Our mesh never exists explicitly as 3D geometric primitives, as it 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. We avoid interpolating across disocclusions by checking the skin bits of each sample (see Section 4.4) and culling 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, Cohen et al. 1997]), but the results are often blurry.

On our PixelFlow hardware, we implemented a quality-based compositing operation similar to the confidence compositing in [Mark and Bishop 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 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. The work per pixel is no larger than depth compositing and requires 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 and Whitted 1985; Grossman and Dally 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 was used in [Larson 1998] 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 be done with a single Z composite.

Figure 3 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.

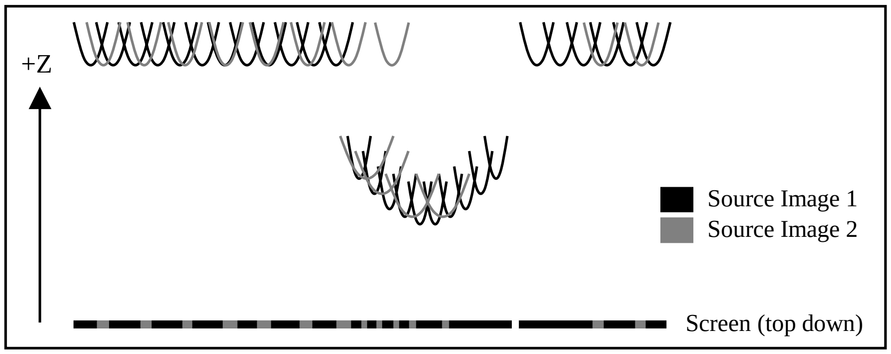


Figure 3. The results of compositing two images reconstructed using quadratic splat primitives, shown as a top-down view of screen space.

The primitive's quadratic shape approximates the central lobe of a gaussian splat's alpha component, as in volume rendering [Westover 1990] but the primitives can be rendered in any order, instead of 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 and Baum 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, Gortler et al. 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. APPLICATIONS FOR VIEWING 3D PHOTOGRAPHS

While the usefulness of 3D photographs may seem obvious, this section describes both end-user applications and research issues enabled by the availability of such data.

An obvious application for 3D photographs is to fully capture the scene of an automobile accident, a crime, an airplane crash or other tragedy. With such photographs, investigators would be able to "re-visit" the scene as needed. Some characteristics of these scenes are that important information is unique (location of objects, marks indicating trajectories, and deformations), temporary, and of legal importance in understanding what occurred. Runways have to be cleaned up and intersections must be cleared of accidents to restore them to their normal use. Currently, photographs and measurements are taken, but these provide only limited information. Often, the site is revisited to acquire additional data with more elaborate equipment (such as surveying equipment), but object removal certainly hampers full understanding.

Any application where the viewer cannot easily visit the scene will also benefit greatly from 3D photographs. Examples are visiting distant historic sites and museums. The Digital Michelangelo project [Levoy 1999] has put forth tremendous effort to accurately capture range data of many of Michelangelo's sculptures. When interactively displayed, the viewer will not only be able to view the sculptures from all the locations where they could stand, but will also be able to view them from high above the floor or other arbitrary locations. In addition, they will be able to get much closer to the sculptures than is currently allowed, and even make annotations on the virtual sculpture.

Related to remote walk-through activities is historic (or forensic) preservation. With the rapid change of our world, many structures or environments will be lost forever. This includes buildings, archeological sites, monuments, classic automobiles, and natural environments. Multiple panoramic 3D photographs of these objects or environments will preserve them for later viewing, or even for reconstruction.

Renovation and retrofit operations can also benefit from 3D photographs, especially if the site is remote. A good example is renovation or modifications to an oil-drilling platform in the North Sea, since multiple engineers and designers could use the data, annotating and changing it with CAD software without having to make difficult travel to harsh locations.

Remanufacturing requires accurate shape data of existing systems in order to recreate them. In this application, the color data are not as important as the range, but in complicated structures, multiple 3D photographs will be required to adequately describe the object.

Since our work (and that of others) relies on relatively long acquisition times, all of the above applications are based on relatively static scenes. With the advent of real-time acquisition of range images at multiple frames per second (with a device such as that from 3dvSystems [3DV Systems 1999]), many new capabilities arise. Certainly one of the largest will be in entertainment. Instead of watching a movie from the camera's point of view, it may be possible to view the scene from other (probably limited) viewpoints. Sporting events could offer the viewers the opportunity to view the event from any location they like, including positions where, in reality, they would interfere with the game.

Just as sports viewers might like to view a game from positions where the camera cannot possibly be, so would the reviewers of surveillance data. For instance, with real-time 3D cameras placed in a low-flying drone, views could be created as if the viewer were standing on the ground. A similar situation exists for standard surveillance cameras in office buildings, stores, banks and warehouses, since the cameras are typically placed in out-of-the-way locations.

Real-time range acquisition will also impact the world of business, allowing immersive telecollaboration to replace standard teleconferencing. Research is underway at UNC that studies the problem of providing an immersive telecollaboration experience [Raskar, Welch et al. 1998].

Creating 3D photographs raises many research issues. Among the most pressing is an efficient scene representation that supports interactive viewing, given thousands of source images (every room in a building, for instance). The LDI Tree [Chang, Bishop et al. 1999] is one method of removing redundant samples, but does not address the important issue of occlusion culling [Greene and Kass 1993; Zhang, Manocha et al. 1997].

Automatic registration of panoramas with one another is another open research problem, and differs from standard point cloud registration since the space between the sensor and the surface is known to be empty. Once the range scan data is registered it can be treated as a polygonal model and simplified, possibly using standard geometry simplification methods. However, this real data has discontinuities, edges and holes that the algorithms must handle. As these results improve, these real models can be rendered using standard methods, with texture maps coming from the color images. This has been done for mostly-planar indoor scenes in [El-Hakim, Boulanger et al. 1997].

7. CONCLUSIONS AND FUTURE WORK

We have developed a prototype system for acquiring and rendering 3D scenes of the real world. This includes the process of color-to-range registration and panorama-to-panorama registration, as well as distortion removal. The quality of the data we acquire has exceeded our expectations, and when supplied to our exploration software, we generate very realistic renderings of the captured scenes. In fact, the quick acquisition of this kind of data opens many new research areas, which we have merely begun to explore.

Several commercial rangefinders provide data similar in density to ours. The companies K^2T and Cyra sell rangefinders that acquire range images, over larger ranges than ours. However, their goal is usually to convert the data as quickly as possible to triangle-based CAD models, for manipulation with conventional modeling tools, for applications such as renovation and retrofitting. We keep the samples in their original, unsimplified form as long as possible, yielding detailed, accurate image-based renderings.

In the immediate future, we plan to study both the problem of registering panoramas with one another, and the acquisition of more accurately registered color and range information. Both of these issues are bottlenecks in our current process, thus they will receive immediate attention. In addition, our plane-based panorama registration method prevents us from registering outdoor scenes (for example an automobile accident with the car in a ravine). We will experiment with standard range scan registration algorithms and extend them as needed to handle scans of environments, instead of traditional scans of objects.

To support our wide-area tracking environment, we are attacking the problem of video-rate rendering (30 fps or faster) to support stereo viewing in a head-mounted display. This will allow viewers to physically walk around in a scene gaining much more comprehension about the interrelationships of the objects. The fast rendering will be achieved with a combination of methods, including level-of-detail simplification, rapid tile choice, and parallel rendering hardware (at least one GL accelerator for each eye's view, perhaps more).

Since the model of the scene is often incomplete, we are exploring rendering techniques that will show this to the viewer. What typically happens is that an object is occluded by another, so the first object does not appear in any of the acquired images (we call this an "occluded occluder"). If, in the rendering, the volumes that no ray pierced are shown differently, then no assumptions of incorrect visibility will occur. This seems to be a necessary component of using data such as ours for legal decisions.

8. REFERENCES

3DV Systems (1999). The ZCAM Depth Camera.

Chang, C.-F., G. Bishop and A. Lastra (1999). LDI Tree: A Hierarchical Representation for Image-Based Rendering. SIGGRAPH 99.

Curless, B. and M. Levoy (1996). A volumetric method for building complex models from range images. SIGGRAPH '96.

Eberly, D. (1998). MAGIC : My Alternate Graphics and Image Code.

El-Hakim, S. F., P. Boulanger, F. Blais and J.-A. Beraldin (1997). Sensor Based Creation of Indoor Virtual Environment Models. Intl. Conf. on Virtual Systems and MultiMedia - VSMM'97, Geneva, Switzerland.

Eyles, J., S. Molnar, J. Poulton, T. Greer, A. Lastra, N. England and L. Westover (1997). PixelFlow: The Realization. SIGGRAPH/Eurographics Workshop on Graphics Hardware, Los Angeles, CA.

Gortler, S. J., R. Grzeszczuk, Richard Szeliski and M. F. Cohen (1996). The Lumigraph. SIGGRAPH 96.

Greene, N. and M. Kass (1993). Hierarchical Z-Buffer Visibility. SIGGRAPH 93, Anaheim, California.

Grossman, J. P. and W. Dally (1998). Point Sample Rendering. Eurographics Workshop on Rendering Techniques '98, Vienna, Austria.

Haeberli, P. (1990). Paint By Numbers: Abstract Image Representations. SIGGRAPH '90, Dallas, TX.

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

Levoy, M. (1999). The Digital Michelangelo Project.

Levoy, M. and P. Hanrahan (1996). Light Field Rendering. SIGGRAPH'96.

Levoy, M. and T. Whitted (1985). The Use of Points as a Display Primitive. UNC-Chapel Hill Dept. of Computer Science**:** Technical Report 85-022 Chapel Hill.

Mark, W. R. and G. Bishop (1997). Memory Access Patterns of Occlusion-Compatible 3D Image Warping. Proceedings of SIGGRAPH/Eurographics Workshop on Graphics Hardware, Los Angeles, California.

Max, N. (1996). Hierarchical Rendering of Trees from Precomputed Multi-Layer Z-Buffers. 7th Eurographics Workshop on Rendering, Porto, Portugal.

McAllister, D. K., L. Nyland, V. Popescu, A. Lastra and C. McCue (1999). Real-Time Rendering of Real-World Environments. Rendering Techniques '99, Proceedings of the Eurographics Workshop on Rendering, Granada, Spain.

McMillan, L. (1997). An Image-based Approach to Three-Dimensional Computer Graphics. University of North Carolina at Chapel Hill**:** Ph.D. Dissertation , also available as *UNC Technical Report TR97-013*.

McMillan, L. and G. Bishop (1995). Plenoptic Modeling: An Image-Based Rendering System. SIGGRAPH 95, Los Angeles, CA.

Molnar, S. (1991). Efficient Supersampling Antialisasing for High-Performance Architectures. UNC Computer Science**:** Technical Report TR91-023 .

Molnar, S., J. Eyles and J. Poulton (1992). PixelFlow: High-Speed Rendering Using Image Composition. SIGGRAPH '92.

Montrym, J. S. and D. R. Baum (1997). InfiniteReality: A Real-Time Graphics System. SIGGRAPH '97, Los Angeles, CA.

Neider, J., T. Davis and M. Woo (1993). OpenGL Programming Guide, Addison Wesley.

Olano, M. and A. Lastra (1998). A Shading Language on Graphics Hardware: The PixelFlow Shading System. SIGGRAPH 98, Orlando, Florida.

Pulli, K., M. Cohen, T. Duchamp, H. Hoppe, L. Shapiro and W. Stuetzle (1997). View-based Rendering: Visualizing Real Object from Scanned Range and Color Data. Eurographics Rendering Workshop 97, St. Etienne, France.

Raskar, R., G. Welch, M. Cutts, A. Lake, L. Stesin and H. Fuchs (1998). The Office of the Future: A Unified Approach to Image-Based Modeling and Spatially Immersive Displays. SIGGRAPH 98, Orlando, FL.

Seitz, S. M. and K. N. Kutulakos (1998). Plenoptic Image Editing. Proc. ICCV 98.

Shade, J., S. Gortler, L.-w. Hey and R. Szeliski (1998). Layered Depth Images. SIGGRAPH 98, Orlando, FL.

Szeliski, R. and H.-Y. Shum (1997). Creating Full View Panoramic Image Mosaics and Environment Maps. SIGGRAPH 97, Los Angeles, CA.

Taylor, R. M., W. Robinett, V. L. Chi, J. Frederick P. Brooks, W. V. Wright, R. S. Williams and E. J. Snyder (1993). "The Nanomanipulator: A Virtual-Reality Interface for a Scanning Tunneling Microscope." SIGGRAPH 93.

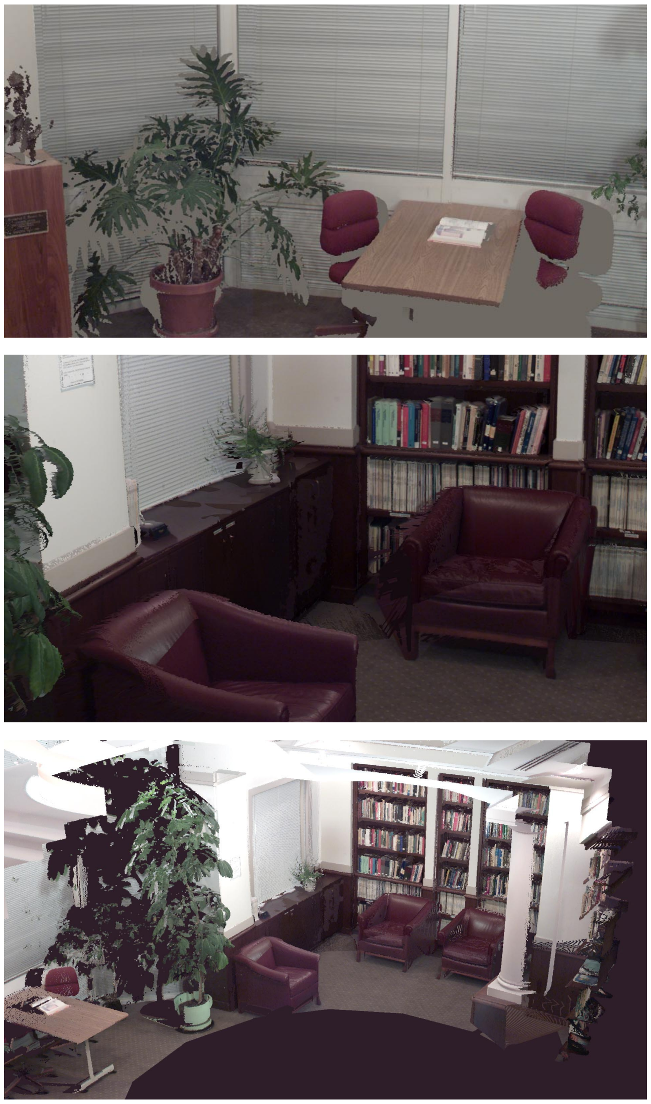
Tsai, R. Y. (1986). An Efficient and Accurate Camera Calibration Technique for 3D Machine Vision. Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, Miami Beach, FL.

Westover, L. (1990). "Footprint Evaluation for Volume Rendering." SIGGRAPH 90 **24**: 367--376.

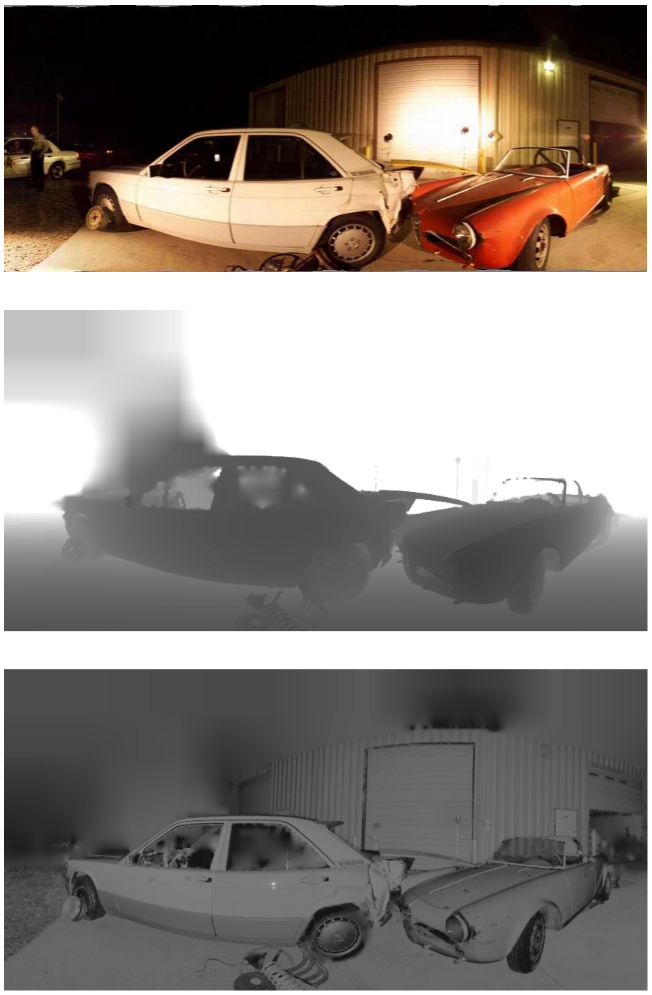
Yoo, T. S. and J. M. Coggins (1993). Using Statistical Pattern Recognition Techniques to Control Variable Conductance Diffusion. Information Processing in Medical Imaging (Lecture Notes in Computer Science 687). H. H. Barrett and A. F. Gmitro. Berlin, Springer-Verlag**:** 495-471.

Zhang, H., D. Manocha, T. Hudson and K. E. H. III (1997). Visibility Culling Using Hierarchical Occlusion Maps. SIGGRAPH 97.

Color Plate 1. A series of images rendered from two panoramic 3D photographs. This demonstrates the ability to explore data acquired at a scene, achieving a better understanding of the spatial relationships of the objects,



Color Plate 2. A single panoramic 3D photograph consists of the color data, shown at the top, the range data (a gray-scale rendering is shown in the center), and the reflected light of the range image, shown at the bottom.



Color Plate 3. Combining several scans into a single view.

Color Plate 4. A scan of Henry Fuchs's office.