Real-Time 3D Model Acquisition

Szymon Rusinkiewicz   Olaf Hall-Holt   Marc Levoy
Princeton University*   Stanford University†

Abstract

The digitization of the 3D shape of real objects is a rapidly expanding field, with applications in entertainment, design, and archaeology. We propose a new 3D model acquisition system that permits the user to rotate an object by hand and see a continuously-updated model as the object is scanned. This tight feedback loop allows the user to find and fill holes in the model in real time, and determine when the object has been completely covered. Our system is based on a 60 Hz. structured-light rangefinder, a real-time variant of ICP (iterative closest points) for alignment, and point-based merging and rendering algorithms. We demonstrate the ability of our prototype to scan objects faster and with greater ease than conventional model acquisition pipelines.

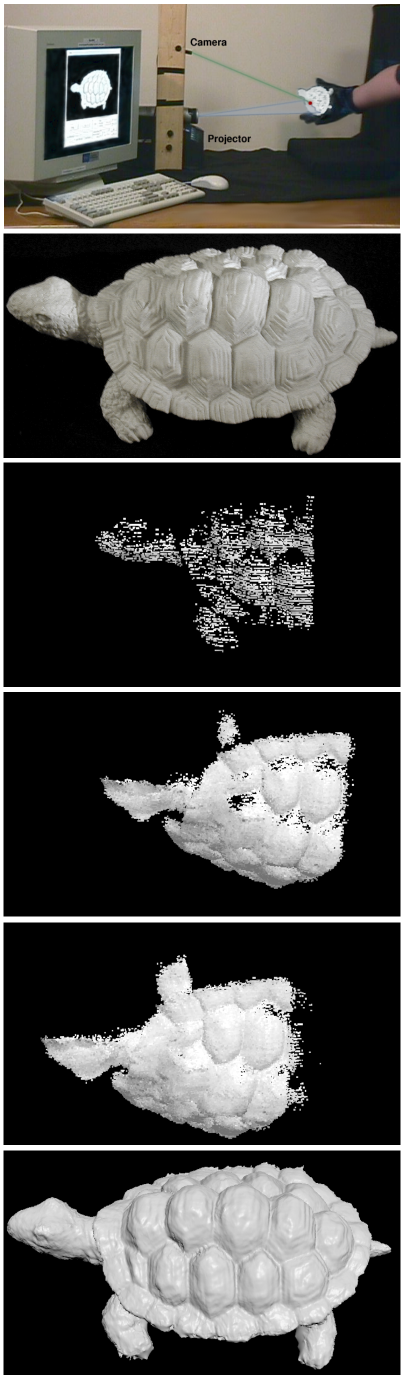
Categories and Subject Descriptors: I.3.1 [Computer Graphics]: Hardware Architecture – Input devices; I.4.8 [Image Processing and Computer Vision]: Digitization and Image Capture – Imaging geometry, scanning; B.4.2 [Input/Output and Data Communications]: Input/Output Devices.

Additional Keywords: 3D model acquisition, 3D scanning, range scanning, real-time modeling.

1 Introduction

The desire to reduce the dependence on human input in making realistic images of complex scenes has, over the past ten years, resulted in an increased role for measurements of the real world in the computer graphics pipeline. In this paper, we focus on using 3D scanning to build detailed models of complex objects for use in rendering.

Although for some uses of range scanning it is sufficient to obtain a single range image of the object, many applications require a complete 3D model. Since most 3D scanners only obtain data from one side of an object (or even a small region of one side) at a time, it is necessary to move the scanner (or, equivalently, to move the model relative to a stationary scanner) to cover the object from multiple views. These views must then be registered to each other and merged into a single, consistent model. Thus, for these *3D model acquisition* applications, the range scanner itself is



(a) Layout of our system. It consists of a DLP projector that displays structured light patterns and an NTSC video camera. The green and blue lines have been added in this visualization.

(b) Photograph of a turtle figurine, approximately 18 cm. long.

(c) Shortly after the start of scanning, data has been accumulated relatively sparsely. The individual point primitives used by our merging data structure are visible.

(d) After a few seconds of scanning, the front part of the turtle has been covered relatively well. However, the user sees a few remaining holes.

(e) The user turns the object to fill the holes. The user may try a number of different positions until the holes are filled – immediate feedback is available about the effectiveness of each orientation.

(f) Once all the data has been gathered, high-quality offline global registration and surface reconstruction algorithms are used to produce a final model.

Figure 1: Our real-time 3D model acquisition system was used to scan a small turtle figurine. The total scanning time was 4 minutes and the final model, at $^1/_2$ mm. resolution, contains approximately 200,000 polygons. (c) through (e) are rendered using splats (see Section 2.3), and (f) is rendered as a polygon mesh (see Section 2.4).

*Department of Computer Science
35 Olden St.
Princeton University
Princeton, NJ 08544
smr@cs.princeton.edu

†Department of Computer Science
Gates Building 3B
Stanford University
Stanford, CA 94305
{olaf,levoy}@graphics.stanford.edu

only one part of a pipeline that must also include deciding where to take scans (view planning), aligning the scans (registration), and reconstructing a surface (merging). Unless the pipeline is completely automatic it is also necessary to present intermediate results to the user, so a final piece of the pipeline is rendering.

Previous implementations of this pipeline have been proposed, with a variety of technologies at each of the stages [Soucy and Laurendeau 1992; Turk and Levoy 1994; Curless and Levoy 1996]. Although such systems have been used to create complex, high-resolution models, as in several recent sculpture digitization projects [Rushmeier et al. 1998; Levoy et al. 2000; Miyazaki et al. 2000], doing so is neither fast nor easy. One of the reasons for this difficulty is that, in most existing pipelines, human intervention is needed at several stages. First, most range image registration algorithms require the user to provide an initial guess by roughly aligning new scans to the existing 3D model. Next, all scans are rendered together and the operator uses this information to perform view planning (e.g., to find holes). Each iteration of this alignment / view planning cycle is time consuming, and feedback about the effectiveness of the scanner's position is not available until the next scan is performed and aligned. Automated next-best-view systems have been proposed to eliminate the user from this loop [Maver and Bajcsy 1993; Stamos and Allen 1998], but they are often computationally intensive and require computer control over the position of the scanner.

We propose a new system that returns range images in real time (60 Hz.) and lets the user perform hole detection and view planning interactively. Our system is based on a consumer video projector, used to generate sequences of structured light patterns, and a consumer video camera, used to observe those patterns. The user holds the object in his or her hands and moves it slowly. As range images are acquired, they are automatically aligned to the existing scans and merged together to create a 3D model. Our alignment and merging algorithms are optimized to work well given the small translations and rotations that occur when the object is moved slowly by hand. By displaying a rough merged model in real-time on a computer screen, the user can find holes (unscanned areas) visually, and move the object to fill them. Although our approach requires a human in the loop, it eliminates the need for expensive calibrated motion control stages, and it eliminates the time delay incurred by offline view planning algorithms. This tradeoff makes our system fast and inexpensive, although not automatic. An overview of the system is presented in Figure 1.

In Section 2, we describe the design of our system. We review the structured-light rangefinder [Hall-Holt and Rusinkiewicz 2001] and real-time alignment algorithm [Rusinkiewicz and Levoy 2001] used by our system, then describe our real-time merging and display algorithms. In Section 3 we investigate how to choose which scans to align to each other, and describe what happens when the system loses track of the correct alignment (e.g., because the object is moved out of the field of view of the scanner). In Section 4 we present a few notes on implementation, and in Section 5 we present sample results from our prototype. In Section 6 we analyze the performance of our system, and in Section 7 we present some ideas for future research.

2 Real-Time Range Scanning Pipeline

As stated above, our model acquisition system consists of a real-time pipeline integrating a rangefinder and algorithms for alignment, merging, and rendering. In order to implement a prototype of this pipeline that operates at interactive rates on current hardware, it was necessary to make some tradeoffs in the technologies and algorithms that were used. In the following sections we present the structured-light rangefinder, real-time alignment algorithm, voxel-based merging algorithm, and splat-based renderer used by our system.

2.1 Structured-Light Range Scanning

Our 3D model acquisition pipeline begins with a real-time (60 Hz.) rangefinder. Previous real-time 3D scanning systems have been proposed based on defocus [Pentland et al. 1989; Nayar et al. 1996], stereo [Kanade et al. 1996], silhouettes [Matsumoto et al. 1997; Matusik et al. 2000], structured light [Gruss et al. 1992; Proesmans et al. 1998], and time of flight (commercial systems by 3DV and Perceptron). We chose to use a system based on projected structured-light triangulation, since it uses off-the-shelf hardware, permits (slowly) moving objects, and returns a full range image, as opposed to a single point or line of 3D data, every 60th of a second. Here we present a brief summary of the design of this system; a full description appears in [Hall-Holt and Rusinkiewicz 2001].

Simple Structured-Light Systems: Our rangefinder is based on the principle of structured-light triangulation. As shown in Figure 2, the simplest such system consists of a projector that emits a stripe (plane) of light and a camera placed at an angle with respect to the projector. At each point in time, the camera obtains 3D positions for points along a 2D contour traced out on the object by the plane of light.

In order to obtain a full range image, it is necessary either to sweep the stripe along the surface (as is done by many commercial single-stripe laser range scanners) or to project multiple stripes. Although projecting multiple stripes leads to faster data acquisition, such a system must have some method of determining which stripe is which. There are three major ways of doing this: assuming surface continuity so that adjacent projected stripes are adjacent in the camera image [Proesmans et al. 1998], differentiating the stripes based on color [Boyer and Kak 1987], and coding the stripes by varying their illumination over time. The first approach (assuming continuity) allows depth to be determined from a single frame but fails if the surface contains discontinuities. Using color allows more complicated surfaces but fails if the surface is textured. Temporal stripe coding is robust to moderate surface texture but takes several frames to compute depth and, depending on the design, may fail if the object moves.

In the simplest systems using temporal coherence, the entire acquisition process takes some number of frames n, and in each frame any given stripe is either turned on or off. The on-off pattern of each stripe is chosen to uniquely identify that stripe. For example, suppose that we wish to use 2^n different stripes. We assign each stripe a code from 0 to $2^n - 1$ and look at each of these codes as an n-bit binary number. We project n different patterns, where in pattern k each stripe is on if and only if the k-th bit of its code is a 1. Thus, after any given pixel has been observed for n frames, the identity of the stripe that illuminated that pixel may be determined.

Searching for Stripe Boundaries: The type of time-coded structured-light system described above has one significant disad-

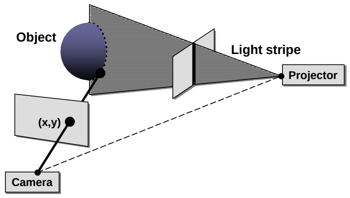


Figure 2: Schematic layout of a single-camera, single-stripe-source triangulation system. The 3D positions of points on the object are determined from the intersection between the camera ray and the plane of light produced by the illumination source.

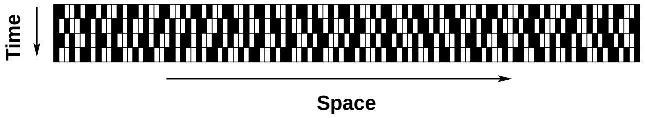


Figure 3: A four-frame sequence of projected patterns. Each pattern has 110 stripe boundaries (111 stripes), with the property that each stripe boundary has a unique code (consisting of the black/white illumination history on either side of the boundary over the sequence of four frames).

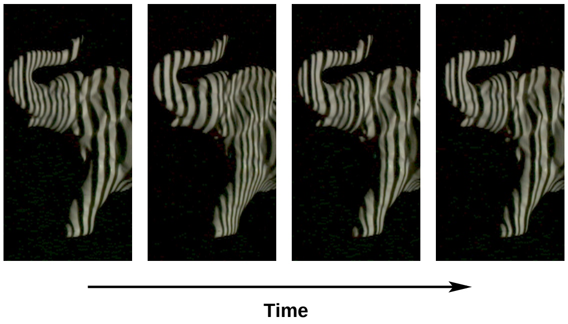


Figure 4: A sequence of video frames of an elephant figurine illuminated with a stripe boundary code. The projector cycles among these patterns at 60 Hz., and the illuminated object is photographed by an NTSC camera synchronized to the projector.

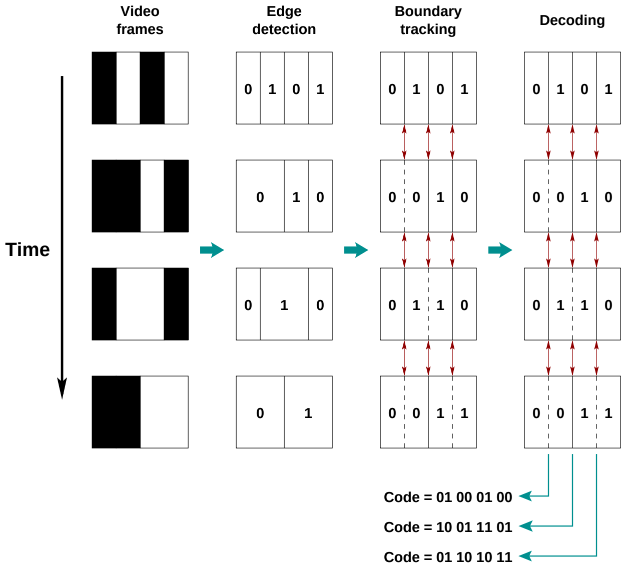


Figure 5: Tracking stripe boundaries. Our system identifies edges in the video frames, tracks the stripe boundaries over time, and propagates the illumination history of each boundary over four frames to determine the boundary's code. Note that the tracking stage has to infer the presence of "ghost" boundaries (shown as dashed lines) in some of the double-wide black and white regions. Our projected patterns are designed to minimize the presence of such invisible "boundaries," and have the property that there are never two ghosts adjacent in space or in time.

vantage: the object being scanned must not move while the *n* coded frames are being obtained. If the object does move, the system is likely to obtain an incorrect stripe code, and hence incorrect 3D data.

In order to allow object motion during scanning, it is necessary to add frame-to-frame tracking into the general framework of time-coded structured light. In particular, if the object moves slowly enough that stripes may be tracked over time, it is possible to accumulate the bits of the code over time, despite the fact that they do not come from the same camera pixel. In practice, one can only track a region by tracking its boundaries. Therefore, we design a system around the idea of tracking projected stripe boundaries, and using these boundaries to convey codes over time.

Given the framework of tracking stripe boundaries as they move from frame to frame, we search for a way of coding stripes that is optimized for this application. First, we observe that it is desirable for stripes to be as narrow as possible, so that the set of boundaries we are tracking is as dense as possible. Since the bit-plane technique of simple binary coding results in very wide stripes in some frames, this is not a desirable code for our application. Second, our shift of focus from stripes to boundaries permits us to convey information more efficiently: instead of obtaining one bit of information per frame, we may obtain two bits of information by looking at the on/off status on both sides of the boundary. Thus, we wish to design a code such that every *pair* of adjacent stripes has a unique code over time.

The final criterion in designing our codes is the observation that, since we are looking at both sides of a stripe boundary, that "boundary" will not be visible at every frame, since the stripes on either side of that boundary might both be either on or off. Thus, we design a code that minimizes the number of these invisible "ghosts," to maximize the chances of tracking these boundaries even if they disappear. Note that this is just a special case of the general strategy of making stripes as narrow as possible, as described above. As described in [Hall-Holt and Rusinkiewicz 2001], it is possible to design a code that never contains two ghosts adjacent in space or in time.

Figure 3 shows the sequence of illumination patterns used by our system. There are four different patterns, which can uniquely code 110 stripe boundaries. The system cycles among these frames at 60 Hz., captures video with a synchronized NTSC camera (Figure 4), performs the boundary tracking and decoding (Figure 5), and returns a range image at each point in time. Since we may use *any* sequence of four consecutive frames to obtain depth, there is a latency of 4 frames in identifying a new stripe boundary but depths are obtained at every frame thereafter.

2.2 Fast 3D Registration

The previous section described the design of a range scanner capable of returning the 3D shape of a moving object as seen from a single viewpoint in real time. As mentioned in the introduction, however, the goal of our system is to produce complete models of rigid objects. Therefore, we must align the range images from different viewpoints that are produced as a rigid object is moved relative to the range scanner.

There are three classes of methods that have been considered for this application. The first relies on known motion: the object and scanner are moved relative to each other by a calibrated rotational or translational stage. As mentioned earlier, however, a key usability improvement in our design comes from lifting the restriction to calibrated motion and allowing the object and scanner to be moved freely with respect to each other. In addition, avoiding calibrated motion helps reduce the cost of building and calibrating the scanner [Davis and Chen 2001].

A second way of obtaining the alignment between range images is to place a tracker on either the object or the scanner (whichever is moved relative to the other). Although we have chosen not to use this option because of accuracy and cost considerations, we believe that in many circumstances it would provide substantial benefits in the context of our proposed pipeline. As discussed in Section 6, a separate tracker would be especially useful for preventing the global drift that results from our use of scan-to-scan alignment.

The option we use for alignment in our pipeline is based on registering individual scans to each other based on the geometry in overlapping areas. The algorithm we have chosen is a variant of

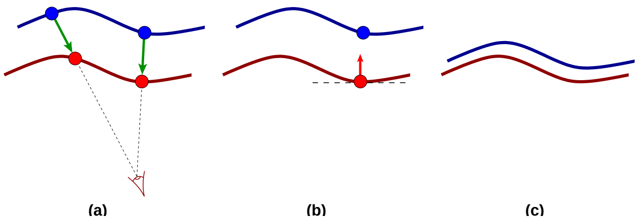


Figure 6: Schematic view of projection-based ICP. (a) A random subset of points on one scan (shown at top in blue) is selected. For each point, a ray is found through the point of projection of the other scan (shown below in red). The point along this ray that intersects the red range image is taken as the matching point. (b) For each such pair, the distance is minimized between one of the points and the plane through the other point and perpendicular to its normal. (c) One scan is moved to minimize the sum of squared point-to-plane distances, bringing the two scans into closer alignment. The process may now be repeated until the scans are aligned, according to some convergence criterion.

ICP (Iterative Closest Points), which is widely used for geometric alignment of three-dimensional models when an initial estimate of the relative pose is known. In the real-time range scanner application, the relative motion between two consecutive range images is small, so in the simplest case we may simply align each range image to the previous one.

The ICP algorithm [Besl and McKay 1992] has become the dominant method for aligning three-dimensional models based purely on the geometry, and sometimes color, of the meshes. ICP starts with two meshes and an initial guess for their relative rigid-body transform, and iteratively refines the transform by repeatedly generating pairs of corresponding points on the meshes and minimizing an error metric. As described in [Rusinkiewicz and Levoy 2001], it is possible to decompose the ICP algorithm into six stages:

1. **Selection** of some set of points in one or both meshes. This is typically uniform or random subsampling.
2. **Matching** these points to samples in the other mesh. In the original ICP algorithm, this was done by matching to the closest point in the other mesh, though other approaches are possible.
3. **Weighting** the corresponding pairs appropriately, for example based on the distance between the points or compatibility of normals.
4. **Rejecting** certain pairs based on looking at each pair individually or considering the entire set of pairs. This is done to eliminate outliers.
5. Assigning an **error metric** to the current alignment, based on the point pairs. This is usually point-to-point distance, but Chen and Medioni have observed that it is more effective to minimize the distance from one point to the plane containing the other point and perpendicular to its normal [Chen and Medioni 1991].
6. **Minimizing** the error metric (e.g. via least squares).

By choosing a different variant at each stage, we obtain a variety of algorithms with different stability, robustness, and efficiency characteristics.

For our current needs, we are interested in an algorithm that offers the highest possible performance. After some experimentation, we have chosen the following variant, described in more detail in [Rusinkiewicz and Levoy 2001]:

1. Random sampling of points on one mesh.
2. Projection-based matching [Blais and Levine 1995] – see Figure 6.
3. Uniform weighting of all point pairs.
4. Rejection of pairs with point-to-point distance greater than a threshold.
5. Point-to-plane error metric [Chen and Medioni 1991].
6. The “standard” iterative minimization.

The result is an algorithm that, in our application, aligns scans in a few milliseconds, fast enough to be incorporated into our real-time pipeline. The main gain in speed over most other ICP-like algorithms comes from the use of projection-based matching. This relies on the fact that a range image is naturally organized as a 2-D array of samples. Finding a matching point by projecting into the range image is therefore just indexing into the array, and does not require a comparatively slow 3D closest-points search. An overview of the algorithm is presented in Figure 6.

2.3 Merging and Rendering

The goal of a live preview of the scanned model is to give the user feedback about which areas of the model have been scanned, and whether any holes are left. Since the model accumulates incrementally in our system, some way must be found to display this constantly increasing set of range samples. The traditional way to display multiple scans is to merge them using a surface reconstruction algorithm, thereby forming a 3D model represented by a polygon mesh. Although there are many such algorithms that produce high-quality results, running times are typically on the order of several seconds or minutes for models of moderate complexity. To avoid these long running times, we have chosen to avoid reconstructing a surface in our merging step, instead directly displaying the raw range samples.

The difficulty with this approach is that in a real-time scanning system, data is accumulating rapidly – hundreds of thousands of points per second in our case. Thus, to maintain an acceptable interactive frame rate, it is necessary to perform some sort of merging or discarding of redundant 3D data in order to reduce the number of primitives that must be rendered. Our approach relies on collapsing points in cells of a voxel grid.

Our merging algorithm first triangulates each new range image and computes per-vertex surface normals. Next, after aligning the range image, we discard the connectivity information and perform merging based only on the points themselves. This consists of quantizing all points to a 3D grid and combining all points that map to the same location in this grid (the idea of collapsing all points in a voxel grid cell is similar to the one used in the mesh simplification method of Rossignac and Borrel [1993]). A running-average normal is maintained at each grid cell for use in rendering.

The rendering is done using a method called “splatting,” which has recently seen increased interest for rendering large, detailed 3D models [Rusinkiewicz and Levoy 2000; Pfister et al. 2000]. In this technique, a screen-aligned splat (e.g. an alpha-blended Gaussian or an opaque circle or square) is drawn for each point, scaled such that the splats for neighboring points overlap without leaving a gap. When the points are regularly arranged on a grid, such as in this application, their spacing is known *a priori* and it is simple to select a splat size for a given viewpoint that guarantees that the splats for adjacent samples overlap without leaving holes. Thus, a rendering is produced that gives the appearance of a merged surface without the need to triangulate the points or to reconstruct a consistent polygon mesh. The merging and rendering algorithms are illustrated in Figure 7.

Although this merging algorithm runs quickly, it does not produce the highest-quality results. If scans are misaligned by more than the grid spacing, samples from those scans will not be merged, creating two or more layers of occupied voxels. After many range images have been incorporated, the result is a region of occupied voxels in the vicinity of the true surface (see Figure 8). Since these voxels are rendered with Z-buffering, the visible voxels will be those that are outermost, causing our surface to appear thickened

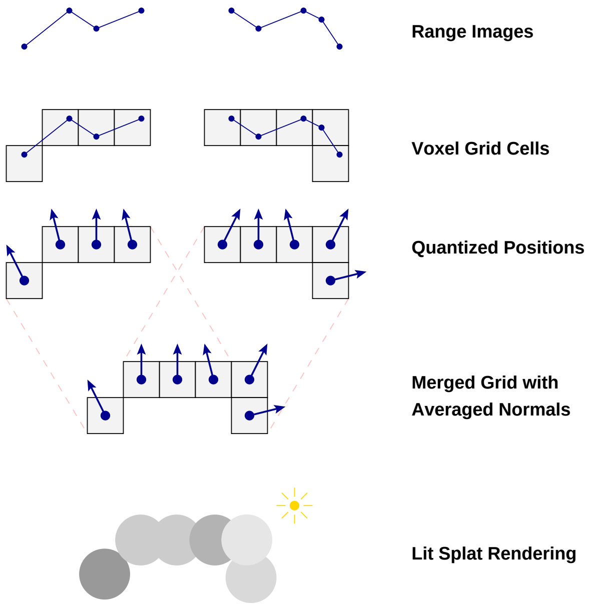


Figure 7: Grid-based merging and rendering. Samples from each range image are quantized in a voxel grid and are merged with samples from other scans. An average normal is accumulated at each grid cell, and splatting is used to render the merged samples. All of these illustrations represent plan views, but the splats in the bottom row have been turned to face the reader so that their shape and color may be seen.

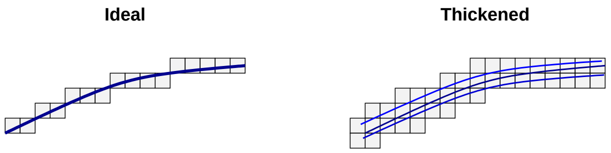


Figure 8: If there were no noise in the scans and alignment were perfect, only a thin later of voxels would be accumulated in our grid-based merging data structure (left). In practice, we see a wider layer of voxels (right). This results in thickened silhouettes and a noisier rendering. The overall visual effect is still acceptable, however, since splats are shaded using average normals.

and possibly noisy. However, the quality of the real-time reconstruction only needs to be good enough to guide the user in positioning the object and determining the presence of holes in the model; a high-quality reconstruction may be performed offline at the conclusion of the scanning process. Moreover, the visual quality of the thickened surface is better than one might expect: although the surface geometry is noisy, voxels are rendered with averaged normals so the shading appears relatively smooth.

Outlier Elimination: As mentioned above, moderate misalignment and noise does not degrade the appearance of the merged grid unacceptably. Large outliers, however, *are* visible, and it is necessary to eliminate them before merging. Fortunately, since we acquire data at such a high rate, we can be aggressive in eliminating outliers; any correct data that is mistakenly discarded is highly likely to be acquired again, because the user will see a hole and acquire replacement data on the spot. Thus, the rapid feedback given to the user, in addition to helping with view planning, actually simplifies the problem of dealing with noisy data.

Our outlier rejection algorithm operates after a range image has been triangulated. We eliminate all long and thin triangles, as well as triangles that are backfacing with respect to the camera or projector. If this results in points that are not part of any triangle, those points are eliminated.

Depending on the parameter in the above algorithm (that is, the “skinniest” permitted triangles), this method may eliminate a certain amount of data that was seen at a large tilt with respect to the camera. This is not necessarily a drawback, since that data is likely to be of lower quality anyway. Discarding this data encourages the user to turn those regions of the object towards the range scanner (so that higher-quality data may be obtained). For creases in some objects, however, it may be impossible to orient the object such that a given portion of the surface is both visible from and normal to the camera. Therefore, the outlier elimination should not be set to be too aggressive. In practice, we currently discard triangles in which the smallest angle is less than 10 degrees.

Grid Size: In order to perform this voxel-based merging and rendering, it is necessary to choose an appropriate grid size. Smaller grid cells result in greater detail, but at the expense of greater memory usage, lower frame rates, and greater sensitivity to noise. Larger grids give higher frame rates, but the individual splats become more visible. In addition, when using larger grids the user may not be able to see small holes in the data.

In practice, we use a grid size on the order of the spacing of samples given by the range scanner; higher grid resolution results in substantially lower frame rates without corresponding increases in perceived quality. For our prototype, we usually use a grid size of $^1/_2$ mm., which is roughly equal to the range sample spacing near the front of our working volume.

2.4 Offline Registration and Surface Reconstruction

As mentioned above, the quality achievable with the real-time pipeline is not as high as that of state-of-the-art offline registration and surface reconstruction algorithms. Thus, once the user has acquired data for the entire object, an offline post-process may be run to reconstruct a final, high-quality surface. Because the user is confident that all necessary data has been acquired and because the results of the real-time registration are available as an initial guess, this process may run completely automatically, without further user intervention.

We use the following post-processing pipeline:

- The grid data structure used for real-time merging is discarded, and all further processing is performed on the original range images.
- Starting from scan positions and transforms computed by the real-time algorithm, ICPs are performed on consecutive scans, as well as additional pairs of overlapping scans. The algorithm used at this stage is not the high-speed ICP variant used in the real-time pipeline but a slower, higher-quality algorithm [Pulli 1999].
- A globally-optimal alignment is computed by simultaneously considering the results of all the scan-to-scan alignments and performing a global relaxation [Pulli 1999].
- The range images are triangulated, and merged using the VRIP algorithm [Curless and Levoy 1996].
- A final, merged triangular mesh is extracted using marching cubes [Cline et al. 1988].

Additional processing (such as decimation, smoothing, or filling of inaccessible holes) may now be performed on the model.

3 Anchor Scans and Restarting Alignment

So far, we have suggested that the alignment stage of the pipeline always involves performing an ICP between a range image and the previous one. Under ideal conditions, this usually works, but in practice an ICP will occasionally fail. This might happen if the object is moved too fast, such that the stripe boundary tracking fails, and it will certainly happen if the object is moved out of the field of view of the scanner. Once such a situation has been corrected (i.e., the object slows down to a reasonable velocity and/or comes back within the field of view), it is necessary to regain the alignment of the new scans to the previously-acquired model.

A reasonable approach to restarting after a failed alignment might be to treat it as a general problem of aligning two 3D models given an unknown initial pose. There exist algorithms to solve this problem [Johnson and Hebert 1997; Chen et al. 1999], but they are typically slow and not robust. Instead, we may take advantage of the human operator's strengths in performing pattern matching by simply displaying one or more range images, asking the user to position the object so it roughly lines up with one of them, and attempting to perform ICP to those range images until one of the ICPs succeeds.

If we adopt this strategy, we must choose which range images to present to the user. We have observed that just using the last range image before the ICP failure is not a good choice. If the object is moving out of the field of view, the last few range images see smaller and smaller pieces of the object. A user is likely to have difficulty in aligning the object to one of these. More seriously, having smaller and smaller amounts of range data decreases the constraints on ICP, thus increasing the likelihood of obtaining an incorrect alignment.

Anchor Scans: In order to make it possible to display good range images for restarting ICP, we maintain some number of *anchor scans* to which we attempt to align. We would like these to have the following characteristics:

- The anchor scans should be large enough to be recognizable to the user and for ICP to operate reliably.
- We would like the anchors to have relatively low overlap among themselves, so that they cover as much of the object as possible.

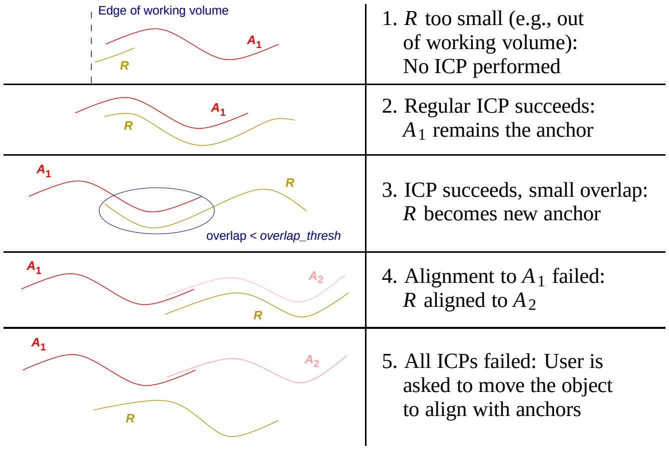
In order to maintain the above properties, we define a set of anchors $A_1 \ldots A_n$, initially consisting of only the first scan taken. As each additional range image R is acquired, the alignment and recovery algorithms are as follows:

1. If R is empty or has fewer than *align_min* points we do not attempt alignment, since such an alignment would be more likely to be incorrect. We indicate to the user that the object is out of the working volume.
2. Otherwise, we attempt to align R to A_1. Our initial guess for the transform is the result of the last successful ICP.
3. If the ICP succeeds, we evaluate whether R should become a new anchor. This is done if:
   - R is large (more than *anchor_min* points), and
   - The overlap between R and the anchor for which ICP succeeded is less than *overlap_thresh*.

   If both of these conditions hold, R becomes A_1, the old anchors $A_1 \ldots A_{n-1}$ become $A_2 \ldots A_n$, and the old A_n is deleted.
4. If the initial ICP fails, we retry it a number of times using more iterations, larger thresholds, and several perturbations to the starting position of R. If all of these attempts fail, we try to align R to each of $A_2 \ldots A_n$. If any of these succeeds, we move that anchor to the front of the list (so it becomes A_1).
5. If ICP to all of the anchors has failed, we draw R and $A_1 \ldots A_n$ in different colors and ask the user to move the object to line up R with one of the anchors.

The effects of this algorithm are shown in Figure 9.

The above rules for deciding when a scan becomes an anchor were chosen so that anchors tend to be large and reasonably spaced out (in practice, we observe that 5–10% of scans become anchors). This makes it easy for the user to restart the "normal" ICP after a failed alignment. In addition, it has the benefit of preventing the accumulation of alignment errors when the object is not moving. This is because, when the object is not moving, the overlap between each new scan and the anchor will be large, so the anchor will not change. Thus, all of the scans will be aligned to the same anchor. If instead we performed simple scan-to-scan alignment, the small errors introduced by each ICP would have the potential of accumulating, leading to more global drift in the alignment.



1. R too small (e.g., out of working volume): No ICP performed
2. Regular ICP succeeds: A_1 remains the anchor
3. ICP succeeds, small overlap: R becomes new anchor
4. Alignment to A_1 failed: R aligned to A_2
5. All ICPs failed: User is asked to move the object to align with anchors

Figure 9: As described at left, our system maintains a number of "anchor scans" $A_1 \ldots A_n$ to which each new scan R is aligned. The different cases of the algorithm ensure that the anchors are relatively large and spread out over the surface, leading to easier (manual) error recovery and less accumulated drift in alignment. Except in case 5, the different colors used in this visualization are not visible to the user.

4 Implementation

Hardware: The system uses a Compaq MP1800 DLP projector, with a maximum resolution of 1024x768. Because of the need to synchronize it with the video camera, we currently send an S-Video signal to the projector, limiting us to a resolution of 640x240 interlaced. The camera we use is a Sony DXC-LS1 NTSC camera, with a 1/500 sec. shutter speed. The video is digitized by a Pinnacle Studio DC10+ capture card, yielding interlaced 640x240 video fields at 60 Hz.

CPU Usage: Our prototype uses a dual-CPU system, with Intel Pentium III Xeon processors running at 1 GHz. One CPU is used for the first few stages of the range scanning pipeline, namely grabbing video frames, finding stripe boundaries, matching the boundaries across time, and identifying the boundaries from the accumulated illumination history. The second CPU performs triangulation to find 3D points, aligns the scans using the fast ICP algorithm, integrates range images into the 3D grid, and renders the updated grid. The first piece of this pipeline operates at full speed (60 Hz.), while the second operates slower, approximately 10 Hz.

The reason for choosing this unequal division of stages among CPUs is to ensure that the matching stage does not drop frames; this permits the highest-possible speeds for object motion. It is not as critical for the rest of the pipeline to run at the full 60 Hz. camera rate, since this only results in a lower frame rate for the display.

Layout: The layout of the system determines its working volume and resolution. For the scans presented here, we have positioned the camera and projector 20 cm. apart, with a triangulation angle of 21 degrees. This configuration produces a working volume approximately 10 cm. across. Near the front of the working volume, samples are spaced roughly every 0.5 mm. in Y (parallel to the stripe direction) and every 0.75 mm. in X (perpendicular to the stripes).

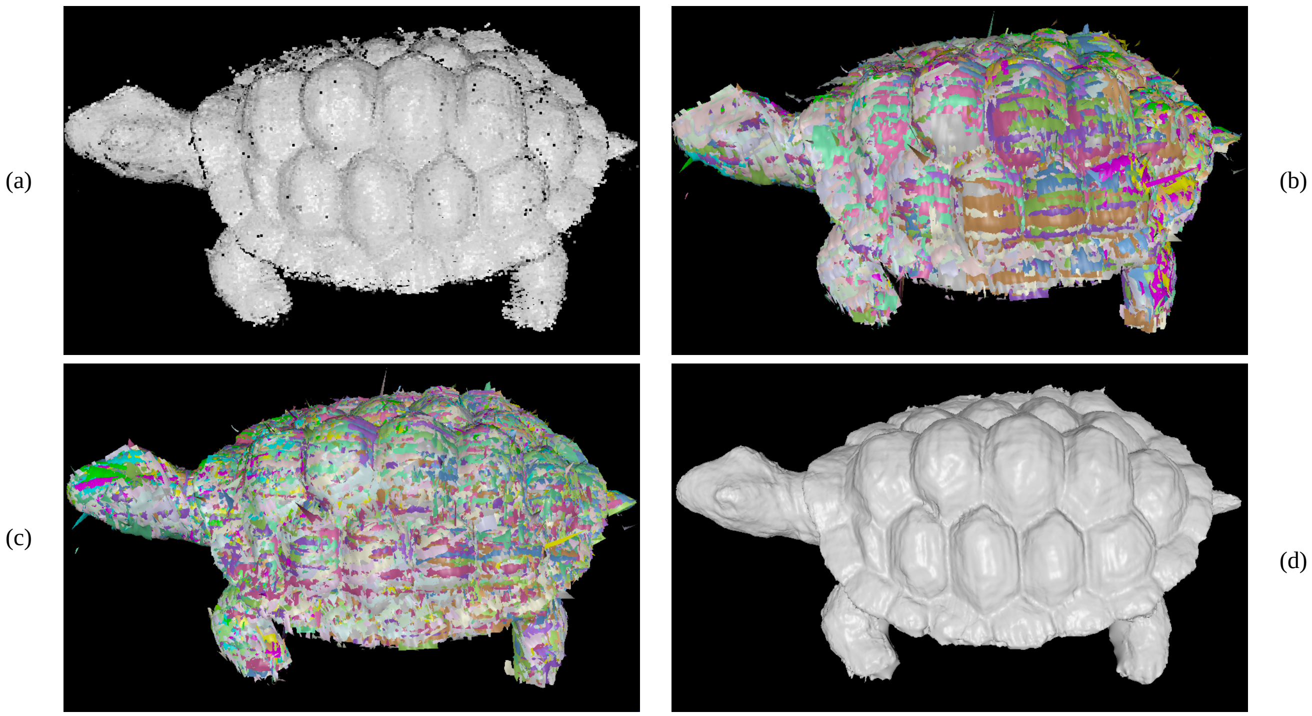
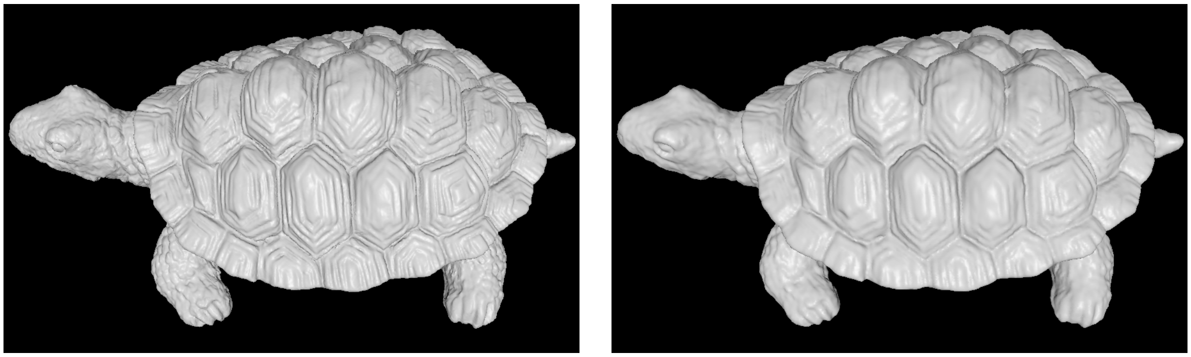


Figure 10: Post-processing stages for the turtle figurine shown in Figure 1. (a) The grid data structure at the completion of scanning (similar to Figure 1c–e). The black speckles are due to undetected outlier points with incorrect normals, not holes. (b) The data from the grid is displayed as triangulated range images, each in a different color. For clarity, we show only every tenth range image. (c) The range images have been aligned with a global registration algorithm. Note that fewer regions of constant color are visible (i.e., more interpenetration of scans), showing that a better alignment has been achieved. (d) The VRIP surface reconstruction algorithm has been used to generate a single, merged surface (same as Figure 1f).



Cyberware Model 15

Cyberware Model 15, subsampled 2x

	Our scanner	**Cyberware Model 15**	**Cyberware, subsampled**
Range images	1,830	22	22
Avg. samples / scan	3,140	75,380	18,845
Sample spacing (x)	0.75 mm.	0.25 mm.	0.5 mm.
Sample spacing (y)	0.5 mm.	0.33 mm.	0.66 mm.
Alignment	automatic	manual	manual
Scanning time	4 min.	30 min.	30 min.

Figure 11: Scans of the turtle figurine performed with a Cyberware Model 15 single-stripe laser triangulation scanner (compare Figure 10d). The scans have been aligned using the algorithm of [Pulli 1999] starting from a manual initial alignment, and have been merged using VRIP [Curless and Levoy 1996]. Since our scanner has a lower resolution than the Cyberware, we also show a model generated by subsampling the original Cyberware scans by a factor of 2 in each dimension.

5 Results

In Figure 10, we focus on the turtle figurine of Figure 1, and compare the results of the real-time pipeline to the post-processed version (after global registration and VRIP surface reconstruction). Figure 11 compares our scanner to a commercially-available single-stripe laser triangulation scanner. The results are similar, though the model we generate is not as high-quality as that produced by the Cyberware scanner, chiefly because of digitizer noise and miscalibration (see the following section). Finally, some other objects scanned using our prototype system are shown in Figure 12.

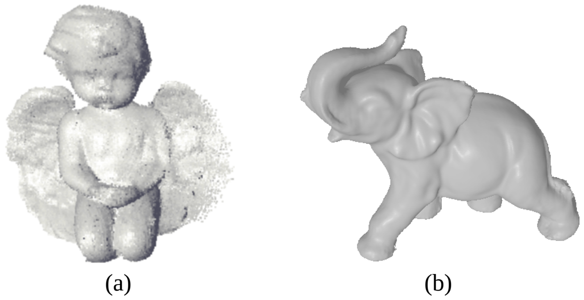


Figure 12: Additional examples of objects scanned using our system. (a) An angel statuette is shown at an intermediate point during scanning (see video). (b) The final model for the elephant figurine of Figure 4.

6 Analysis

Accuracy and Precision: There are several ways in which the accuracy of our range scanner may be characterized. We may look at the spacing between samples on the surface, the noise in the location of each sample, and the distortion in each range image due to miscalibration. All of these depend on the physical arrangement of camera and projector, and so must be compared to the size of the working volume. The figures below all apply to the front 10 cm. section of the working volume discussed above.

The sample spacing in our prototype, as we have stated, is 0.5-0.75 mm. The noise in each of these samples is primarily due to the error in locating a stripe boundary, which may be due to noise in the camera and digitizer or due to object texture. For surfaces without high-frequency texture, we may find the locations of the stripe boundaries with subpixel precision, and we estimate the noise in each sample to be under 0.1 mm. (ignoring, for the moment, outliers and distortion). This is due almost entirely to noise in the camera and capture card: using a higher-quality camera and digitizer (Toshiba IK-TU40A 3-CCD camera and DPS-465 digitizer) we obtain lower per-sample noise – under 0.03 mm.

Ultimately, using a high-quality camera and digitizer, the limit on the minimum achievable local noise depends on two factors. The first is the focus of the camera and projector. Using a smaller aperture (especially on the projector – most commercially-available models use large apertures) would permit better localization of stripe boundaries. The second major limit on the accuracy of this system is scene texture. When the reflectance of the surface varies rapidly (on the order of the camera pixel spacing), we are not able to perform accurate subpixel estimation, and so the per-sample error is substantially larger, on the order of 0.2 mm.

Calibration and Warping: In addition to noise in the samples, the accuracy of our scans is degraded by the distortion in our scanner due to miscalibration. We have adopted a calibration procedure in which known 3D points in the scene are measured using a Faro arm touch probe, and their (u, v) camera locations as well as projector p coordinate are found. The optimal intrinsic and extrinsic calibration parameters are found by minimizing the error in all the $(u, v, p) \rightarrow (x, y, z)$ mappings simultaneously.

Although we may estimate the error in the calibration directly from the convergence of the minimization algorithm, a more meaningful estimate arises from considering the maximal misalignment between range images of the same object taken at a variety of different positions and orientations. For the turtle data set of Figure 10, we observe a misalignment of approximately 0.5 mm. (after high-quality ICP and global registration), leading us to conclude that the distortion is of this order of magnitude.

Registration Drift and Effects on Scanning Strategy: Even though the distortion in each individual scan due to miscalibration is fairly small, the effects of this distortion can easily add up from frame to frame, contributing to the “registration drift” problem mentioned in Section 3. Despite the use of anchor scans (which reduce the occurrence of the problem by a large factor) in some cases this drift is sufficiently severe that it becomes impractical to use the system to scan completely around an object. In this case, the system may be used to scan the object in a small number of pieces, and the pieces joined together at the end of scanning (an initial guess for the relative alignment must be obtained from the user). Alternatively, a small number of scans may be taken around the object, a global registration algorithm [Pulli 1999] may be run to align them, then the real-time pipeline may be used to fill holes in this “skeleton,” by always using the pre-aligned scans as the anchors. Regardless of the strategy chosen, the pipeline retains its main advantage of simplifying the process of hole-filling.

An alternative approach to solving the registration drift problem would be to integrate our system with conventional 6-DOF trackers, placed either on the object or the scanner depending on which is moved relative to the other. Using such trackers (either jointed digitizing arms or untethered systems) would permit the scanner to be applied to larger objects or environments by eliminating the registration error due to noise or miscalibration. In addition, it would make the system more stable in the presence of degenerate geometry (such as planes or cylinders) for which ICP is not able to determine all six degrees of freedom of the alignment.

7 Conclusions and Future Work

This paper has described a new 3D model acquisition system designed to be inexpensive, fast, and easy to use, and demonstrated results from a prototype implementation. In contrast with previous systems, our design permits the user to rotate an object by hand and see a continuously-updated model as the object is scanned, thus providing instant feedback about the presence of holes and the amount of surface that has been covered. The system uses off-the-shelf components and runs on today’s CPUs.

The ability to provide real-time feedback to the user has yielded benefits throughout the model acquisition pipeline. In addition to simplifying the view planning problem, it has proven useful in restarting after a failed alignment and prompted an aggressive strategy for outlier rejection. We anticipate future work in exploring the ways that a high data rate and instant user feedback affect the 3D scanning pipeline. Here we suggest only a few ideas for future work; a more complete list may be found in [Rusinkiewicz 2001].

There are several ways in which the system described here could be improved. Some, such as obtaining texture, would be relatively easy to incorporate into our pipeline. Others might involve changes in hardware, such as higher-resolution cameras and projectors (for higher-quality data or larger working volumes), high-speed cameras and projectors (for faster allowable motion), or multiple cameras or projectors (for faster scanning and better coverage).

In addition to improvements in hardware, one possible algorithmic improvement, possible with increasing CPU speeds or custom hardware, would be to improve the quality of the real-time merging and rendering algorithms. For example, with an order of magnitude faster CPU it would be possible to implement an implicit-surface reconstruction algorithm, such as VRIP, and either extract a polygon mesh or use volume rendering hardware to display it interactively. Another potential improvement would be to allow the pipeline to use available data from later stages to help the earlier stages. For example, ambiguities in edge detection or tracking might be resolved by looking at other range images, or even at the accumulated model. By casting the entire pipeline in a probabilistic framework, one could maintain multiple hypotheses, with confidence estimates, and delay making irrevocable decisions as long as possible.

For certain classes of objects, one might consider solving the model acquisition problem in the presence of nonrigid deformation. Although the first stage of our current pipeline (the 3D scanner) can handle deformation, the alignment and merging stages would require considerable changes. There has been recent work on tracking non-rigid objects [Costeira and Kanade 1998; Bregler et al. 2000], though much of it assumes either that an initial model is available or that the deformation is heavily constrained. Acquisition of deformable models would be especially attractive for capturing human animation [Allen et al. 2002].

Finally, we note that one major benefit of using a triangulation-based system is that it potentially scales to many different working volumes. One could imagine scaling the system up for scanning building interiors or movie sets, or scaling it down for applications in industrial inspection or medicine. The major problems when scaling up would be the physical size of the baseline, emitting enough light by the projector (relative to ambient light), and depth of field of the camera and projector. When scaling down, the major challenges would involve focus and the diffraction limit of light.

Acknowledgments

This research grew out of a system built together with Li-Wei He, and benefitted from conversations with Lucas Pereira, Sean Anderson, James Davis, and many others at the Stanford Graphics Lab. We would also like to thank Intel, Sony, and Interval for their financial support.

References

ALLEN, B., CURLESS, B., AND POPOVIC, Z. 2002. "Human Body Deformation From Range Scans," *Proc. ACM SIGGRAPH 2002*.

BESL, P. AND MCKAY, N. 1992. "A Method for Registration of 3-D Shapes," *Trans. PAMI*, Vol. 14, No. 2.

BLAIS, G. AND LEVINE, M. 1995. "Registering Multiview Range Data to Create 3D Computer Objects," *Trans. PAMI*, Vol. 17, No. 8.

BOYER, K. L. AND KAK, A. C. 1987. "Color-Encoded Structured Light for Rapid Active Ranging," *Trans. PAMI*, Vol. 9, No. 1.

BREGLER, C., HERTZMANN, A., AND BIERMANN, H. 2000. "Recovering Non-Rigid 3D Shape from Image Streams," *Proc. CVPR 2000*.

CHEN, C., HUNG, Y., AND CHENG, J. 1999. "RANSAC-Based DARCES: A New Approach to Fast Automatic Registration of Partially Overlapping Range Images," *Trans. PAMI*, Vol. 21, No. 11.

CHEN, Y. AND MEDIONI, G. 1991. "Object Modeling by Registration of Multiple Range Images," *Proc. IEEE Conf. on Robotics and Automation 1991*.

CLINE, H. E., LORENSEN, W. E., LUDKE, S., CRAWFORD, C. R., AND TEETER, B. C. 1998. "Two Algorithms for the Three-Dimensional Reconstruction of Tomograms," *Medical Physics*, Vol. 15, No. 3.

COSTEIRA, J. AND KANADE, T. 1998. "A Multi-Body Factorization Method for Motion Analysis," *IJCV*, Vol. 29, No. 3.

CURLESS, B. AND LEVOY, M. 1996. "A Volumetric Method for Building Complex Models from Range Images," *Proc. ACM SIGGRAPH 96*.

DAVIS, J. AND CHEN, X. 2001. "A Laser Range Scanner Designed for Minimum Calibration Complexity," *Proc. 3DIM 2001*.

GRUSS, A., TADA, S., AND KANADE, T. 1992. "A VLSI Smart Sensor for Fast Range Imaging," *Proc. IEEE Int. Conf. on Intelligent Robots and Systems 1992*.

HALL-HOLT, O. AND RUSINKIEWICZ, S. 2001. "Stripe Boundary Codes for Real-Time Structured-Light Range Scanning of Moving Objects," *Proc. ICCV 2001*.

JOHNSON, A. AND HEBERT, M. 1997. "Surface Registration by Matching Oriented Points," *Proc. 3DIM 1997*.

KANADE, T., YOSHIDA, A., ODA, K., KANO, H., AND TANAKA, M. 1996. "A Stereo Machine for Video-rate Dense Depth Mapping and Its New Applications," *Proc. CVPR 1996*.

LEVOY, M., PULLI, K., CURLESS, B., RUSINKIEWICZ, S., KOLLER, D., PEREIRA, L., GINZTON, M., ANDERSON, S., DAVIS, J., GINSBERG, J., SHADE, J., AND FULK, D. 2000. "The Digital Michelangelo Project: 3D Scanning of Large Statues," *Proc. ACM SIGGRAPH 2000*.

MATSUMOTO, Y., TERASAKI, H., SUGIMOTO, K., AND ARAKAWA, T. 1997. "A Portable Three-Dimensional Digitizer," *Proc. 3DIM 1997*.

MATUSIK, W., BUEHLER, C., RASKAR, R., GORTLER, S., AND MCMILLAN, L. 2000. "Image-Based Visual Hulls," *Proc. ACM SIGGRAPH 2000*.

MAVER, J. AND BAJCSY, R. 1993. "Occlusions as a Guide for Planning the Next View," *Trans. PAMI*, Vol. 15, No. 5.

MIYAZAKI, D., OOISHI, T., NISHIKAWA, T., SAGAWA, R., NISHINO, K. TOMOMATSU, T., TAKASE, Y., AND IKEUCHI, K. 2000. *Proc. VSMM 2000*.

NAYAR, S. K., WATANABE, M., AND NOGUCHI, M. 1996. "Real-Time Focus Range Sensor," *Trans. PAMI*, Vol. 18, No. 12.

PENTLAND, A., DARRELL, T., TURK, M. AND HUANG, W. 1989. "A Simple, Real-Tie Range Camera," *Proc. CVPR 1989*.

PFISTER, H., ZWICKER, M., VAN BAAR, J., AND GROSS, M. 2000. "Surfels: Surface Elements as Rendering Primitives," *Proc. ACM SIGGRAPH 2000*.

PROESMANS, M. VAN GOOL, L., AND DEFOORT, F. 1998. "Reading Between the Lines – A Method for Extracting Dynamic 3D with Texture," *Proc. ICCV 1998*.

PULLI, K. 1999. "Multiview Registration for Large Data Sets," *Proc. 3DIM 1999*.

ROSSIGNAC, J. AND BORREL, P. 1993. "Multi-Resolution 3D Approximations for Rendering Complex Scenes," *Geometric Modeling in Computer Graphics*.

RUSHMEIER, H., BERNARDINI, F., MITTLEMAN, J. AND TAUBIN, G. 1998. "Acquiring Input for Rendering at Appropriate Levels of Detail: Digitizing a Pietà," *Proc. Eurographics Rendering Workshop 1998*.

RUSINKIEWICZ, S. AND LEVOY, M. 2000. "QSplat: A Multiresolution Point Rendering System for Large Meshes," *Proc. ACM SIGGRAPH 2000*.

RUSINKIEWICZ, S. 2001. "Real-Time Acquisition and Rendering of Large 3D Models," Ph. D. Dissertation, Stanford University.

RUSINKIEWICZ, S. AND LEVOY, M. 2001. "Efficient Variants of the ICP Algorithm," *Proc. 3DIM 2001*.

SOUCY, M. AND LAURENDEAU, D. 1992. "Multi-Resolution Surface Modeling from Multiple Range Views," *Proc. CVPR 1992*.

STAMOS, I. AND ALLEN, P. 1998. "Interactive Sensor Planning," *Proc. CVPR 1998*.

TURK, G. AND LEVOY, M. 1994. "Zippered Polygon Meshes from Range Images," *Proc. ACM SIGGRAPH 94*.