Zippered Polygon Meshes from Range Images

Greg Turk and Marc Levoy
Computer Science Department
Stanford University

Abstract

Range imaging offers an inexpensive and accurate means for digitizing the shape of three-dimensional objects. Because most objects self occlude, no single range image suffices to describe the entire object. We present a method for combining a collection of range images into a single polygonal mesh that completely describes an object to the extent that it is visible from the outside.

The steps in our method are: 1) align the meshes with each other using a modified iterated closest-point algorithm, 2) zipper together adjacent meshes to form a continuous surface that correctly captures the topology of the object, and 3) compute local weighted averages of surface positions on all meshes to form a consensus surface geometry.

Our system differs from previous approaches in that it is incremental; scans are acquired and combined one at a time. This approach allows us to acquire and combine large numbers of scans with minimal storage overhead. Our largest models contain up to 360,000 triangles. All the steps needed to digitize an object that requires up to 10 range scans can be performed using our system with five minutes of user interaction and a few hours of compute time. We show two models created using our method with range data from a commercial rangefinder that employs laser stripe technology.

CR Categories: I.3.5 [Computer Graphics]: Computational Geometry and Object Modelling.
Additional Key Words: Surface reconstruction, surface fitting, polygon mesh, range images, structured light range scanner.

1 Introduction

This paper presents a method of combining multiple views of an object, captured by a range scanner, and assembling these views into one unbroken polygonal surface. Applications for such a method include:

- Digitizing complex objects for animation and visual simulation.
- Digitizing the shape of a found object such as an archaeological artifact for measurement and for dissemination to the scientific community.
- Digitizing human external anatomy for surgical planning, remote consultation or the compilation of anatomical atlases.
- Digitizing the shape of a damaged machine part to help create a replacement.

There is currently no procedure that will allow a user to easily capture a digital description of a physical object. The dream tool would allow one to set an industrial part or a clay figure onto a platform, press a button, and have a complete digital description of that object returned in a few minutes. The reality is that much digitization is done by a user painstakingly touching a 3D sensing probe to hundreds or thousands of positions on the object, then manually specifying the connectivity of these points. Fortunately range scanners offer promise in replacing this tedious operation.

A *range scanner* is any device that senses 3D positions on an object's surface and returns an array of distance values. A *range image* is an $m \times n$ grid of distances (*range points*) that describe a surface either in Cartesian coordinates (a height field) or cylindrical coordinates, with two of the coordinates being implicitly defined by the indices of the grid. Quite a number of measurement techniques can be used to create a range image, including structured light, time-of-flight lasers, radar, sonar, and several methods from the computer vision literature such as depth from stereo, shading, texture, motion and focus. The range images used to create the models in this paper were captured using structured light (described later), but our techniques can be used with any range images where the uncertainties of the distance values are smaller than the spacing between the samples.

Range scanners seem like a natural solution to the problem of capturing a digital description of physical objects. Unfortunately, few objects are simple enough that they can be fully described by a single range image. For instance, a coffee cup handle will obscure a portion of the cup's surface even using a cylindrical scan. To capture the full geometry of a moderately complicated object (e.g. a clay model of a cat) may require as many as a dozen range images.

There are two main issues in creating a single model from multiple range images: *registration* and *integration*. Registration refers to computing a rigid transformation that brings the points of one range image into alignment with the portions of a surface that is shares with another range image. Integration is the process of creating a single surface representation from the sample points from two or more range images.

Our approach to registration uses an iterative process to minimize the distance between two triangle meshes that were created from the range images. We accelerate registration by performing the matching on a hierarchy of increasingly more detailed meshes. This method allows an object to be scanned from any orientation without the need for a six-degree-of-freedom motion device.

E-mail: turk@redclay.stanford.edu, levoy@cs.stanford.edu
Web site: www-graphics.stanford.edu

We separate the task of integration into two steps: 1) creating a mesh that reflects the topology of the object, and 2) refining the vertex positions of the mesh by averaging the geometric detail that is present in all scans. We capture the topology of an object by merging pairs of triangle meshes that are each created from a single range image. Merging begins by converting two meshes that may have considerable overlap into a pair of meshes that just barely overlap along portions of their boundaries. This is done by simultaneously eating back the boundaries of each mesh that lie directly on top of the other mesh. Next, the meshes are *zippered* together: the triangles of one mesh are clipped to the boundary of the other mesh and the vertices on the boundary are shared. Once all the meshes have been combined, we allow all of the scans to contribute to the surface detail by finding the *consensus geometry*. The final position of a vertex is found by taking an average of nearby positions from each of the original range images. The order in which we perform zippering and consensus geometry is important. We deliberately postpone the refinement of surface geometry until after the overall shape of the object has been determined. This eliminates discontinuities that may be introduced during zippering.

The remainder of this paper is organized as follows. Section 2 describes previous work on combining range images. Section 3 covers the basic principles of a structured light range scanner. Section 4 presents the automatic registration process. Section 5 describes zippering meshes into one continuous surface. Section 6 describes how surface detail is captured through consensus geometry. Section 7 shows examples of digitized models and compares our approach to other methods of combining range data. Section 8 concludes this paper by discussing future work.

2 Previous Work

There is a great deal of published work on registration and integration of depth information, particularly in the vision literature. Our literature review only covers work on registration or integration of dense range data captured by an active range scanner, and where the product of the integration is a polygon mesh.

2.1 Registration

Two themes dominate work in range image registration: matching of "created" features in the images to be matched, and minimization of distances between all points on the surface represented by the two images. In the first category, Wada and co-authors performed six degree of freedom registration by matching distinctive facets from the convex hulls of range images [Wada 93]. They computed a rotation matrix from corresponding facets using a least squares fit of the normal vectors of the facets.

In the second category, Champleboux and co-workers used a data structure called an octree-spline that is a sampled representation of distances to an object's surface [Champleboux 92]. This gave them a rapid way to determine distances from a surface (and the distance gradient) with a low overhead in storage. Chen and Medioni establish a correspondence between points on one surface and nearby tangent planes on the other surface [Chen 92]. They find a rigid motion that minimizes the point-to-tangent collection directly and then iterate. Besl and McKay use an approach they call the *iterated closest-point* algorithm [Besl 92]. This method finds the nearest positions on one surface to a collection of points on the other surface and then transforms one surface so as to minimize the collective distance. They iterate this procedure until convergence.

Our registration method falls into the general category of direct distance minimization algorithms, and is an adaptation of [Besl 92]. It differs in that we do not require that one surface be a strict subset of the other. It is described in Section 4.

2.2 Integration

Integration of multiple range scans can be classified into *structured* and *unstructured* methods. Unstructured integration presumes that one has a procedure that creates a polygonal surface from an arbitrary collection of points in 3-space. Integration in this case is performed by collecting together all the range points from multiple scans and presenting them to the polygonal reconstruction procedure. The Delaunay triangulation of a set of points in 3-space has been proposed as the basis of one such reconstruction method [Boissonnat 84]. Another candidate for surface reconstruction is a generalization of the convex hull of a point set known as the alpha shape [Edelsbrunner 92]. Hoppe and co-authors use graph traversal techniques to help construct a signed distance function from a collection of unorganized points [Hoppe 92]. An isosurface extraction technique produces a polygon mesh from this distance function.

Structured integration methods make use of information about how each point was obtained, such as using error bounds on a point's position or adjacency information between points within one range image. Soucy and Laurendeau use a structured integration technique to combine multiple range images [Soucy 92] that is similar in several respects to our algorithm. Given n range images of an object, they first partition the points into a number of sets that are called *common surface sets*. The range points in one set are then used to create a grid of triangles whose positions are guided by a weighted average of the points in the set. Subsets of these grids are stitched together by a constrained Delaunay triangulation in one of n projections onto a plane. We compare our method to Soucy's in Section 7.

3 Structured Light Range Scanners

In this section we describe the operating principles of range scanners based on structured light. We do this because it highlights issues common to many range scanners and also because the range images used in this article were created by such a scanner.

3.1 Triangulation

Structured light scanners operate on the principle of triangulation (see Figure 1, left). One portion of the scanner projects a specific pattern of light onto the object being scanned. This pattern of light is observed by the sensor of the scanner along a viewing direction that is off-axis from the source of light. The position of the illuminated part of the object is determined by finding the intersection of the light's projected direction and the viewing direction of the sensor. Positions can be accumulated across the length of the object while the object is moved across the path of the projected light. Some of the patterns that have been used in such scanners include a spot, a circle, a line, and several lines at once. Typically the sensor is a CCD array or a lateral effect photodiode.

The scanner used for the examples in this paper is a Cyberware Model 3030 MS. It projects a vertical sheet of He-Ne laser light onto the surface of an object. The laser sheet is created by spreading a laser beam using a cylindrical lens into a sheet roughly 2 mm wide and 30 cm high. The sensor of the Cyberware scanner is a 768×486 pixel CCD array. A typical CCD image shows a ribbon of laser light running from the top to the bottom (see Figure 2). A range point is created by looking across a scanline for the peak intensity of this ribbon. A range point's distance from the scanner (the "depth") is given by the horizontal position of this peak and the vertical position of the range point is given by the number of the scanline. Finding the peaks for each scanline in one frame gives an entire column of range points, and combining the columns from multiple frames as the object is moved through the laser sheet gives the full range image.

3.2 Sources of Error

Any approach to combining range scans should attempt to take into account the possible sources of error inherent in a given scanner. Two sources of error are particularly relevant to integration. One is a result of light falling on the object at a grazing angle. When the projected light falls on a portion of the object that is nearly parallel to the light's path, the sensor sees a dim and stretched-out version of the pattern. Finding the center of the laser sheet when it grazes the

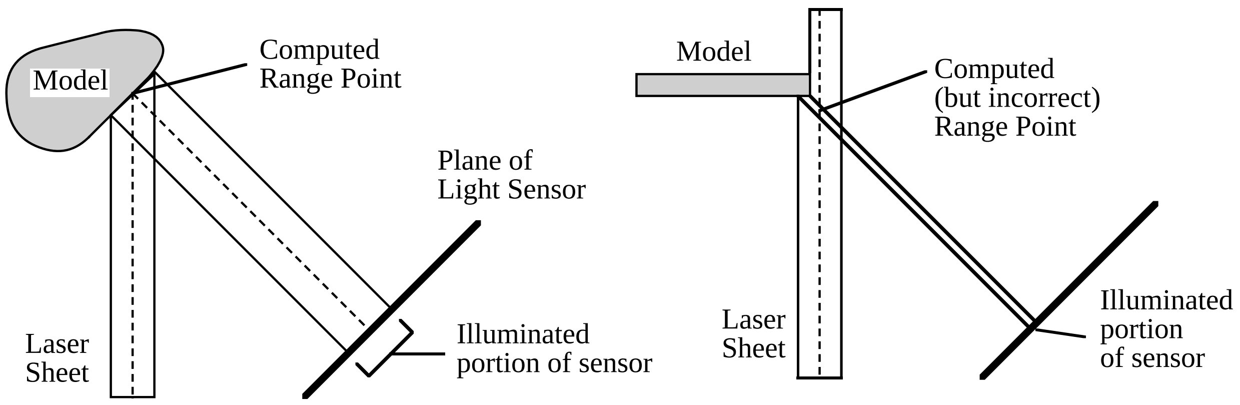
Model
Computed Range Point
Plane of Light Sensor
Laser Sheet
Illuminated portion of sensor
Model
Computed (but incorrect) Range Point
Laser Sheet
Illuminated portion of sensor

Figure 1: Structured light triangulation (left) and false edge extension in the presence of a partially illuminated edge (right).

object becomes difficult, and this adds uncertainty to the position of the range points. The degree of uncertainty at a given range point can be quantified, and we make use of such information at several stages in our approach to combining range images.

A second source of inaccuracy occurs when only a portion of the laser sheet hits an object, such as when the laser sheet falls off the edge of a book that is perpendicular to the laser sheet (see Figure 1, right). This results in a false position because the peak-detection and triangulation method assumes that the entire width of the sheet is visible. Such an assumption results in edges of objects that are both curled and extended beyond their correct position. This false extension of a surface at edges is an issue that needs to be specifically addressed when combining range images.

3.3 Creating Triangle Meshes from Range Images

We use a mesh of triangles to represent the range image data at all stages of our integration method. Each sample point in the $m \times n$ range image is a potential vertex in the triangle mesh. We take special care to avoid inadvertently joining portions of the surface together that are separated by depth discontinuities (see Figure 3).

To build a mesh, we create zero, one or two triangles from four points of a range image that are in adjacent rows and columns. We find the shortest of the two diagonals between the points and use this to identify the two triplets of points that may become triangles. Each of these point triples is made into a triangle if the edge lengths fall below a distance threshold. Let s be the maximum distance between adjacent range points when we flatten the range image, that is, when we don't include the depth information (see Figure 3). We take the distance threshold be a small multiple of this sampling distance, typically $4s$. Although having such a distance threshold may prevent joining some range points that should in fact be connected, we can rely on other range images (those with better views of the location in question) to give the correct adjacency information.

This willingness to discard questionable data is representative of a deliberate overall strategy: to acquire and process large amounts of data rather than draw hypotheses (possibly erroneous) from sparse data. This strategy appears in several places in our algorithm.

4 Registration of Range Images

Once a triangle mesh is created for each range image, we turn to the task of bringing corresponding portions of different range images into alignment with one another. If all range images are captured using a six-degree of freedom precision motion device then the information needed to register them is available from the motion control software. This is the case when the object or scanner is mounted on a robot arm or the motion platform of a precision milling machine. Inexpensive motion platforms are often limited to one or two degrees of freedom, typically translation in a single direction or rotation about an axis. One of our goals is to create an inexpensive system. Consequently, we employ a registration method that does not depend on measured position and orientation. With our scanner, which offers translation and rotation around one axis, we typically take one cylindrical and four translational scans by moving the object with the motion device. To capture the top or the underside of the object, we pick it up by hand and place it on its side. Now the orientation of subsequent scans cannot be matched with those taken earlier, and using a registration method becomes mandatory.

4.1 Iterated Closest-Point Algorithm

This section describes a modified iterated closest-point (ICP) algorithm for quickly registering a pair of meshes created from range images. This method allows a user to crudely align one range image with another on-screen and then invoke an algorithm that snaps the position of one range image into accurate alignment with the other.

The iterated closest-point of [Besl 92] cannot be used to register range images because it requires that *every* point on one surface have

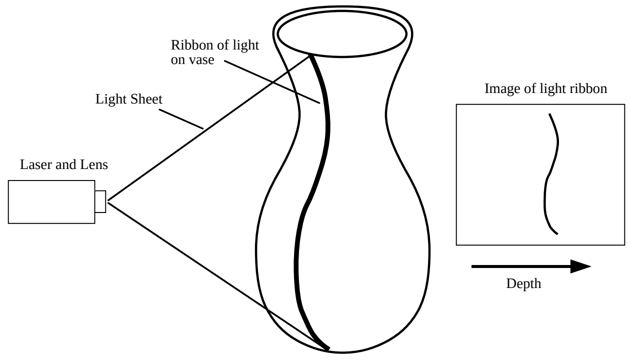

Figure 2: Light-stripe projected on vase (left) and corresponding CCD image (right).

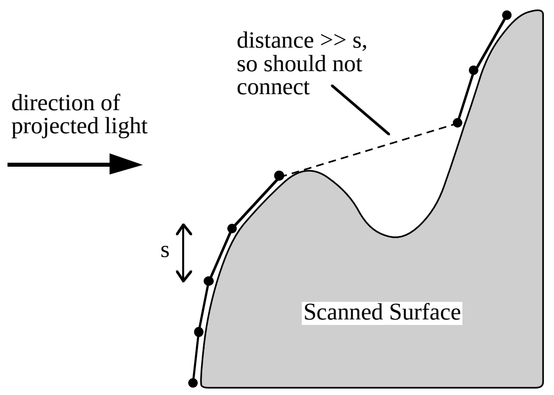
distance >> s, so should not connect
direction of projected light
s
Scanned Surface

Figure 3: Building triangle mesh from range points.

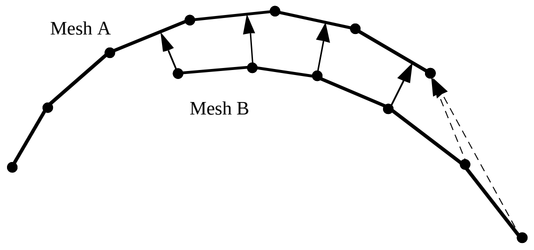

Figure 4: Finding corresponding points for mesh registration. Dotted arrows show matches that should be avoided because they will cause mesh *B* to be erroneously dragged up and left.

a corresponding point on the other surface. Since our scans are overlapping, we seldom produce data that satisfies this requirement. Thus we have developed our own variant of this algorithm. Its steps are:

1) Find the nearest position on mesh *A* to each vertex of mesh *B*.
2) Discard pairs of points that are too far apart.
3) Eliminate pairs in which either points is on a mesh boundary.
4) Find the rigid transformation that minimizes a weighted least-squared distance between the pairs of points.
5) Iterate until convergence.
6) Perform ICP on a more detailed mesh in the hierarchy.

In step 1, it is important to note that we are looking for the 3-space position A_i on the surface of mesh *A* that is closest to a given vertex B_i of mesh *B* (see Figure 4). The nearest point A_i may be a vertex of *A*, may be a point within a triangle, or may lie on a triangle's edge. Allowing these points A_i to be anywhere on a C^0 continuous surface means that the registration between surfaces can have greater accuracy than the spacing *s* between range points.

4.2 Constraints on ICP

Our ICP algorithm differs from Besl's in several ways. First, we have added a distance threshold to the basic iterated closest-point method to avoid matching any vertex B_i of one mesh to a remote part of another mesh that is likely to not correspond to B_i. Such a vertex B_i from mesh *B* might be from a portion of the scanned object that was not captured in the mesh *A*, and thus no pairing should be made to any point on *A*. We have found that excellent registration will result when this distance threshold is set to twice the spacing *s* between range points. Limiting the distance between pairs of corresponding points allows us to perform step 2 (eliminating remote pairs) during the nearest points search in step 1.

The nearest points search can be accelerated considerably by placing the mesh vertices in a uniform subdivision of space based on the distance threshold. Because the triangle size is limited in the mesh creation step, we can search over all triangles within a fixed distance and guarantee that we miss no nearby portion of any triangle. Because we will use this constrained nearest-point search again later, it is worth giving a name to this query. Let nearest_on_mesh(*P*,*d*,*M*) be a routine that returns the nearest position on a mesh *M* to a given point *P*, or that returns nothing if there is no such point within the distance *d*.

Second, we have added the restriction that we never allow boundary points to be part of a match between surfaces. Boundary points are those points that lie on the edge of a triangle and where that edge is not shared by another triangle. Figure 4 illustrates how such matches can drag a mesh in a contrary direction to the majority of the point correspondences.

4.3 Best Rigid Motion

The heart of the iterated closest-point approach is in finding a rigid transformation that minimizes the least-squared distance between the point pairs. Berthold Horn describes a closed-form solution to this problem [Horn 87] that is linear in time with respect to the number of point pairs. Horn's method finds the translation vector *T* and the rotation **R** such that:

$$E = \sum_{i=1}^{n} |A_i - \mathbf{R}(B_i - B_c) - T|^2$$

is minimized, where A_i and B_i are given pairs of positions in 3-space and B_c is the centroid of the B_i. Horn showed that *T* is just the difference between the centroid of the points A_i and the centroid of the points B_i. **R** is found by constructing a cross-covariance matrix between centroid-adjusted pairs of points. The final rotation is given by a unit quaternion that is the eigenvector corresponding to the largest eigenvalue of a matrix constructed from the elements of this cross-covariance matrix. Details can be found in both [Horn 87] and [Besl 92].

As we discussed earlier, not all range points have the same error bounds on their position. We can take advantage of an optional weighting term in Horn's minimization to incorporate the positional uncertainties into the registration process. Let a value in the range from 0 to 1 called *confidence* be a measure of how certain we are of a given range point's position. For the case of structured light scanners, we take the *confidence* of a point *P* on a mesh to be the dot product of the mesh normal *N* at *P* and the vector *L* that points from *P* to the light source of the scanner. (We take the normal at *P* to be the average of the normals of the triangles that meet at *P*.) Additionally, we lower the confidence of vertices near the mesh boundaries to take into account possible error due to false edge extension and curl. We take the confidence of a pair of corresponding points A_i and B_i from two meshes to be the product of their confidences, and we will use w_i to represent this value. The problem is now to find a *weighted* least-squares minimum:

$$E = \sum_{i=1}^{n} w_i |A_i - \mathbf{R}(B_i - B_c) - T|^2$$

The weighted minimization problem is solved in much the same way as before. The translation factor *T* is just the difference between the weighted centroids of the corresponding points. The solution for **R** is described by Horn.

4.4 Alignment in Practice

The above registration method can be made faster by matching increasingly more detailed meshes from a hierarchy. We typically use a mesh hierarchy in which each mesh uses one-forth the number of range points that are used in the next higher level. The less-detailed meshes in this hierarchy are constructed by sub-sampling the range images. Registration begins by running constrained ICP on the lowest-level mesh and then using the resulting transformation as the initial position for the next level up in the hierarchy. The matching distance threshold *d* is halved with each move up the hierarchy.

Besl and McKay describe how to use linear and quadratic extrapolation of the registration parameters to accelerate the alignment process. We use this technique for our alignment at each level in the hierarchy, and find it works well in practice. Details of this method can be found in their paper.

The constrained ICP algorithm registers only two meshes at a time, and there is no obvious extension that will register three or more meshes simultaneously. This is the case with all the registration algorithms we know. If we have meshes *A*, *B*, *C* and *D*, should we register *A* with *B*, then *B* with *C* and finally *C* with *D*, perhaps compounding registration errors? We can minimize this problem by registering all meshes to a single mesh that is created from a cylindrical range image. In this way the cylindrical range image acts as a common anchor for all of the other meshes. Note that if a cylindrical scan covers an object from top to bottom, it captures all the surfaces that lie on the convex hull of the object. This means that, for almost all objects, there will be some common portions between the cylindrical scan and all linear scans, although the degree of this overlap depends on the extent of the concavities of the object. We used such a cylindrical scan for alignment when constructing the models shown in this paper.

5 Integration: Mesh Zippering

The central step in combining range images is the integration of multiple views into a single model. The goal of integration is to arrive at a description of the overall topology of the object being scanned. In this section we examine how two triangle meshes can be combined into a single surface. The full topology of a surface is realized by zippering new range scans one by one into the final triangle mesh.

Zippering two triangle meshes consists of three steps, each of which we will consider in detail below:

1) Remove overlapping portions of the meshes.
2) Clip one mesh against another.
3) Remove the small triangles introduced during clipping.

5.1 Removing Redundant Surfaces

Before attempting to join a pair of meshes, we eat away at the boundaries of both meshes until they just meet. We remove those triangles in each mesh that are in some sense "redundant," in that the other mesh includes an unbroken surface at that same position in space. Although this step removes triangles from the meshes, we are not discarding data since all range points eventually will be used to find the consensus geometry (Section 6). Given two triangle meshes *A* and *B*, here is the process that removes their redundant portions:

```
Repeat until both meshes remain unchanged:
    Remove redundant triangles on the boundary of mesh A
    Remove redundant triangles on the boundary of mesh B
```

Before we can remove a given triangle *T* from mesh *A*, we need to determine whether the triangle is redundant. We accomplish this by querying mesh *B* using the `nearest_on_mesh()` routine that was introduced earlier. In particular, we ask for the nearest positions on mesh *B* to the vertices V_1, V_2 and V_3 of *T*. We will declare *T* to be redundant if the three queries return positions on *B* that are within a tolerance distance *d* and if none of these positions are on the boundary of *B*. Figure 7 shows two overlapping surfaces before and after removing their redundant triangles. In some cases this particular decision procedure for removing triangles will leave tiny gaps where the meshes meet. The resulting holes are no larger than the maximum triangle size and we currently fill them in an automatic post-processing step to zippering. Using the fast triangle redundancy check was an implementation decision for the sake of efficiency, not a necessary characteristic of our zippering approach, and it could easily be replaced by a more cautious redundancy check that leaves no gaps. We have not found this necessary in practice.

If we have a measure of confidence of the vertex positions (as we do for structured light scanners), then the above method can be altered to preserve the more confident vertices. When checking to see if the vertices V_1, V_2 and V_3 of *T* lie within the distance tolerance of mesh *B*, we also determine whether at least two of these vertices have a lower confidence measure than the nearby points on *B*. If this is the case, we allow the triangle to be removed. When no more triangles can be removed from the boundaries of either mesh, we drop this confidence value restriction and continue the process until no more changes can be made. This procedure results in a pair of meshes that meet along boundaries of nearly equal confidences.

5.2 Mesh Clipping

We now describe how triangle clipping can be used to smoothly join two meshes that slightly overlap. The left portion of Figure 5 shows two overlapping meshes and the right portion shows the result of clipping. Let us examine the clipping process in greater detail, and for the time being make the assumption that we are operating on two meshes that lie in a common plane.

To clip mesh *A* against the boundary of mesh *B* we first need to add new vertices to the boundary of *B*. Specifically, we place a new vertex wherever an edge of a triangle from mesh *A* intersects the boundary of mesh *B*. Let *Q* be the set of all such new vertices. Together, the new vertices in *Q* and the old boundary vertices of mesh *B* will form a common boundary that the triangles from both meshes will share. Once this new boundary is formed we need to incorporate the vertices *Q* into the triangles that share this boundary. Triangles from mesh *B* need only to be split once for each new vertex to be incorporated (shown in Figure 5, right). Then we need to divide each border triangle from *A* into two parts, one part that lies inside the boundary of *B* that should be discarded and the other part that lies outside of this boundary and should be retained (See Figure 5, middle). The vertices of the retained portions of the triangle are passed to a constrained triangulation routine that returns a set of triangles that incorporates all the necessary vertices (Figure 5, right).

The only modification needed to extend this clipping step to 3-space is to determine precisely how to find the points of intersection *Q*. In 3-space the edges of mesh *A* might very well pass above or below the boundary of *B* instead of exactly intersecting the boundary. To correct for this we "thicken" the boundary of mesh *B*. In essence we create a wall that runs around the boundary of *B* and that is roughly perpendicular to *B* at any given location along the boundary. The portion of the wall at any given edge *E* is a collection of four triangles, as shown in Figure 6. To find the intersection points with the edges of *A*, we only need to note where these edges pass through the wall of triangles. We then move this intersection point down to the nearest position on the edge *E* to which the intersected portion of the wall belongs. The rest of the clipping can proceed as described above.

5.3 Removing Small Triangles

The clipping process can introduce arbitrarily small or thin triangles into a mesh. For many applications this does matter, but in situations where such triangles are undesirable they can easily be removed. We use vertex deletion to remove small triangles: if any of a triangle's altitudes fall below a user-specified threshold we delete one of the triangle's vertices and all the triangles that shared this vertex. We then use constrained triangulation to fill the hole that is left by deleting these triangles (see [Bern 92]). We preferentially delete vertices that were introduced as new vertices during the clipping process. If all of a triangle's vertices are original range points then the vertex opposite the longest side is deleted.

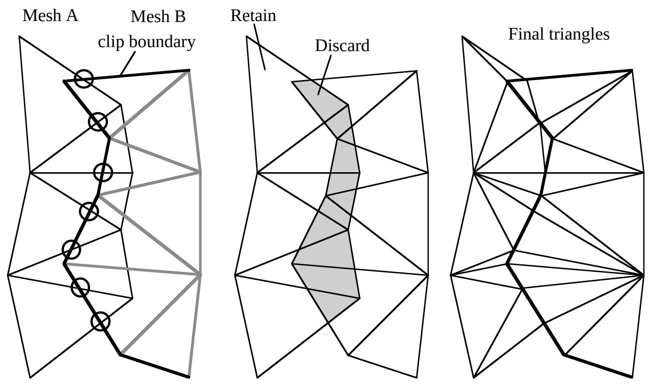

Figure 5: Mesh *A* is clipped against the boundary of mesh *B*. Circles (left) show intersection between edges of *A* and *B*'s boundary. Portions of triangles from *A* are discarded (middle) and then both meshes incorporate the points of intersection (right).

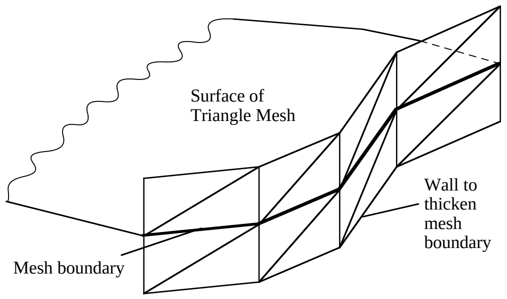

Figure 6: Thickened boundary for clipping in 3-space.

5.4 False Edge Extension

As described in Section 3.2, range points from a structured light scanner that are near an object's silhouette are extended and curled away from the true geometry. These extended edges typically occur at corners. If there is at least one scan that spans both sides of the corner, then our method will correctly reconstruct the surface at the corner. Since we lower the confidence of a surface near the mesh boundaries, triangles at the false edge extensions will be eliminated during redundant surface removal because there are nearby triangles with higher confidence in the scan that spans the corner. For correct integration at a corner, it is the user's responsibility to provide a scan that spans both sides of the corner. Figure 7 illustrates correct integration at a corner in the presence of false edge extension. Unfortunately, no disambiguating scan can be found when an object is highly curved such as a thin cylinder.

Although the problem of false edge extension is discussed in the structured light literature [Businski 92], we know of no paper on surface integration from such range images that addresses or even mentions this issue. We are also unaware of any other integration methods that will correctly determine the geometry of a surface at locations where there are false extensions. Our group has developed a method of reducing false edge extensions when creating the range images (to appear in a forthcoming paper) and we are exploring algorithms that will lessen the effect of such errors during integration. It is our hope that by emphasizing this issue we will encourage others to address this topic in future research on range image integration.

6 Consensus Geometry

When we have zippered the meshes of all the range images together, the resulting triangle mesh captures the *topology* of the scanned object. This mesh may be sufficient for some applications. If surface detail is important, however, we need to fine-tune the *geometry* of the mesh.

The final model of an object should incorporate all the information available about surface detail from each range image of the object. Some of this information may have been discarded when we removed redundant triangles during mesh zippering. We re-introduce the information about surface detail by moving each vertex of our zippered mesh to a consensus position given by a weighted average of positions from the original range images. Vertices are moved only in the direction of the surface normal so that features are not blurred by lateral motion. This is in contrast to unstructured techniques which tend to blur small features isotropically. Our preference for averaging only in the direction of the surface normal is based on the observation that most points in range scans are generally accurately placed with respect to other points in the same scan, but may differ between scans due to alignment errors such as uncorrected optical distortion in the camera. Let $M_1, M_2,..., M_n$ refer to the original triangle meshes created from the range images. Then the three steps for finding the consensus surface are:

1) Find a local approximation to the surface normal.
2) Intersect a line oriented along this normal with each original range image.
3) Form a weighted average of the points of intersection.

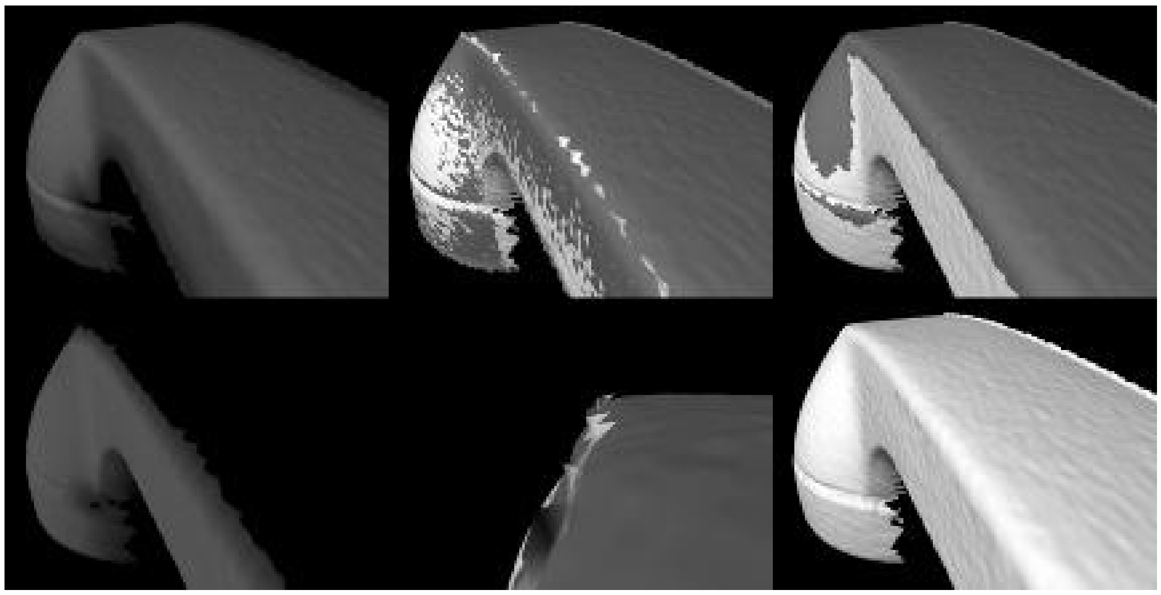

Figure 7: Left (top and bottom): Meshes created from two range images of a telephone. Red denotes locations of high confidence and blue shows low confidence. Note the low confidence at the edges to account for false edge extensions. Top middle: The two meshes (colored red and white) after alignment. Bottom middle: Close-up of aligned meshes that shows a jagged ridge of triangles that is the false edge extension of the white mesh at a corner. Top right: The meshes after redundant surface removal. Bottom right: The meshes after zippering.

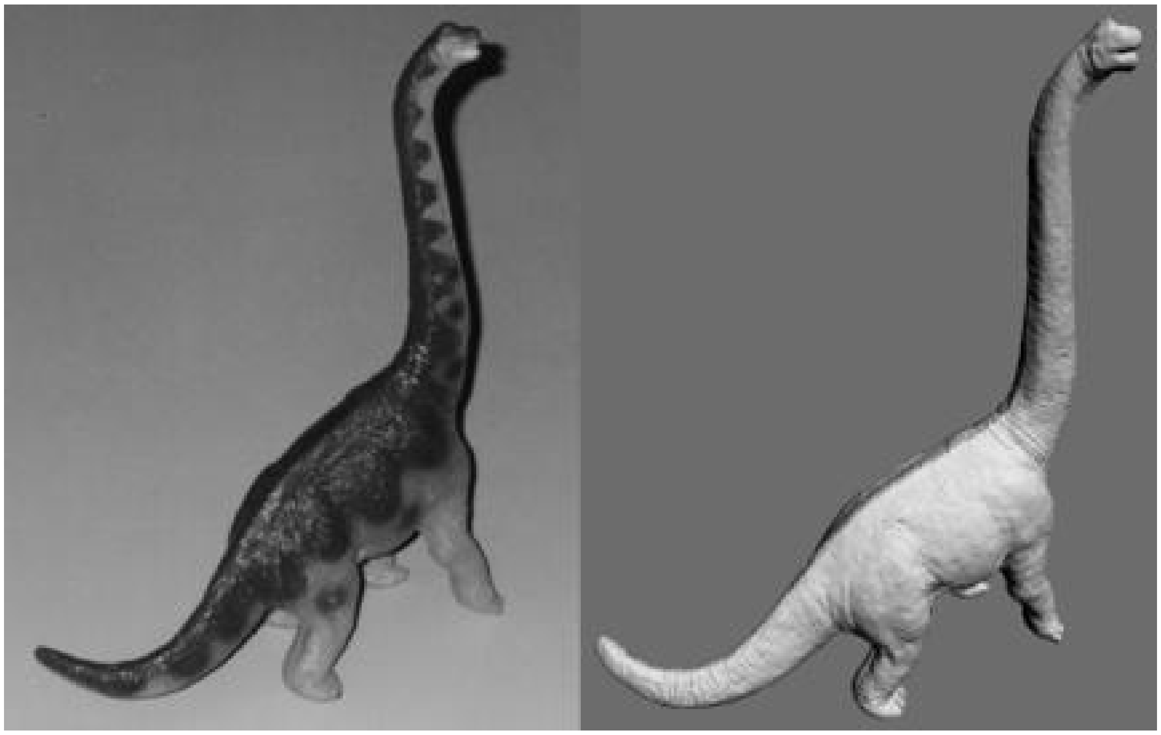

Figure 8: Photograph of a plastic dinosaur model (left) and a polygon mesh created by registering and zippering together 14 range images that were taken of the model (right). The mesh consists of more than 360,000 polygons.

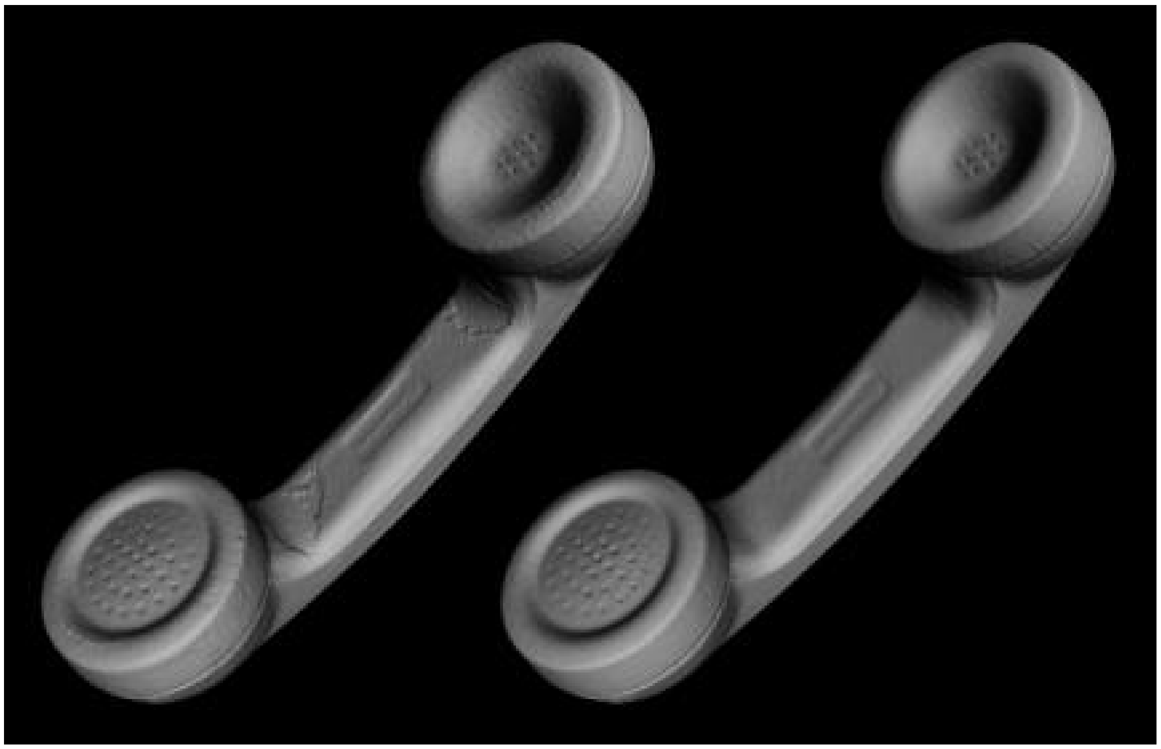

Figure 9: Left: This model of a telephone handset was created by zippering together meshes from ten range images. The mesh consists of more than 160,000 triangles. Right: The final positions of the vertices in the mesh have been moved to an average of nearby positions in the original range images. We call this the consensus geometry.

We approximate the surface normal N at a given vertex V by taking an average over all vertex normals from the vertices in all the meshes M_i that fall within a small sphere centered at V. We then intersect each of the meshes M_i with the line passing through V along the direction N. Let P be the set of all intersections that are near V. We take the consensus position of V to be the average of all the points in P. If we have a measure of confidence for positions on a mesh we use this to weight the average.

7 Results and Discussion

The dinosaur model shown in Figure 8 was created from 14 range images and contains more than 360,000 triangles. Our integration method correctly joined together the meshes at all locations except on the head where some holes due to false edge extensions were filled manually. Such holes should not occur once we eliminate the false extensions in the range images. The dinosaur model was assembled from a larger quantity of range data (measured either in number of scans or number of range points) than any published model known to us. Naturally, we plan to explore the use of automatic simplification methods with our models [Schroeder 92] [Turk 92] [Hoppe 93]. Figure 9 shows a model of a phone that was created from ten range images and contains over 160,000 triangles. The mesh on the right demonstrates that the consensus geometry both reduces noise from the range images without blurring the model's features and also that it eliminates discontinuities at zippered regions.

A key factor that distinguishes our approach from those using unstructured integration ([Hoppe 92] and others) is that our method attempts to retain as much of the triangle connectivity as is possible from the meshes created from the original range images. Our integration process concentrates on a one-dimensional portion of the mesh (the boundary) instead of across an entire two-dimensional surface, and this makes for rapid integration.

Our algorithm shares several characteristics with the approach of Soucy and Laurendeau, which is also a structured integration method [Soucy 92]. The most important difference is the order in which the two methods perform integration and geometry averaging. Soucy's method first creates the final vertex positions by averaging between range images and then stitches together the common surface sets. By determining geometry before connectivity, their approach may be sensitive to artifacts of the stitching process. This is particularly undesirable because their method can create seams between as many as 2^n common surface sets from n range images. Such artifacts are minimized in our approach by performing geometry averaging after zippering.

In summary, we use zippering of triangle meshes followed by refinement of surface geometry to build detailed models from range scans. We expect that in the near future range image technology will replace manual digitization of models in several application areas.

8 Future Work

There are several open problems related to integration of multiple range images. One issue is how an algorithm might automatically determine the next best view to capture more of an object's surface. Another important issue is merging reflectance information (including color) with the geometry of an object. Maybe the biggest outstanding issue is how to create higher-order surface descriptions such as Bezier patches or NURBS from range data, perhaps guided by a polygon model.

Acknowledgments

We thank David Addleman, George Dabrowski and all the other people at Cyberware for the use of a scanner and for educating us about the issues involved in the technology. We thank all the members of our scanner group for numerous helpful discussions. In particular, Brian Curless provided some key insights for interpreting the range data and also wrote code to help this work. Thanks to Phil Lacroute for help with the color figures. This work was supported by an IBM Faculty Development Award, The Powell Foundation, and the National Science Foundation under contract CCR-9157767.

References

[Bern 92] Bern, Marshall and David Eppstein, "Mesh Generation and Optimal Triangulation," Technical Report P92-00047, Xerox Palo Alto Research Center, March 1992.

[Besl 92] Besl, Paul J. and Neil D. McKay, "A Method of Registration of 3-D Shapes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 14, No. 2 (February 1992), pp. 239–256.

[Boissonnat 84] Boissonnat, Jean-Daniel, "Geometric Structures for Three-Dimensional Shape Representation," *ACM Transactions on Graphics*, Vol. 3, No. 4 (October 1984), pp. 266–286.

[Businski 92] Businski, M., A. Levine and W. H. Stevenson, "Performance Characteristics of Range Sensors Utilizing Optical Triangulation," *IEEE National Aerospace and Electronics Conference*, Vol. 3 (1992), pp. 1230–1236.

[Champleboux 92] Champleboux, Guillaume, Stephane Lavallee, Richard Szeliski and Lionel Brunie, "From Accurate Range Imaging Sensor Calibration to Accurate Model-Based 3-D Object Localization," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, Champaign, Illinois, June 15-20, 1992, pp. 83–89.

[Chen 92] Chen, Yang and Gerard Medioni, "Object Modelling by Registration of Multiple Range Images," *Image and Vision Computing*, Vol. 10, No. 3 (April 1992), pp. 145–155.

[Edelsbrunner 92] Edelsbrunner, Herbert and Ernst P. Mücke, "Three-dimensional Alpha Shapes," *Proceedings of the 1992 Workshop on Volume Visualization*, Boston, October 19-20, 1992, pp. 75–82.

[Hoppe 92] Hoppe, Hugues, Tony DeRose, Tom Duchamp, John McDonald and Werner Stuetzle, "Surface Reconstruction from Unorganized Points," *Computer Graphics*, Vol. 26, No. 2 (SIGGRAPH '92), pp. 71–78.

[Hoppe 93] Hoppe, Hugues, Tony DeRose, Tom Duchamp, John McDonald and Werner Stuetzle, "Mesh Optimization," *Computer Graphics Proceedings*, Annual Conference Series (SIGGRAPH '93), pp. 19–26.

[Horn 87] Horn, Berthold K. P., "Closed-Form Solution of Absolute Orientation Using Unit Quaternions," *Journal of the Optical Society of America. A*, Vol. 4, No. 4 (April 1987), pp. 629–642.

[Schroeder 92] Schroeder, William J., Jonathan A. Zarge and William E. Lorensen, "Decimation of Triangle Meshes," *Computer Graphics*, Vol. 26, No. 2 (SIGGRAPH '92), pp. 65–70.

[Soucy 92] Soucy, Marc and Denis Laurendeau, "Multi-Resolution Surface Modeling from Multiple Range Views," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, Champaign, Illinois, June 15-20, 1992, pp. 348–353.

[Turk 92] Turk, Greg, "Re-Tiling Polygonal Surfaces," *Computer Graphics*, Vol. 26, No. 2 (SIGGRAPH '92), pp. 55–64.

[Wada 93] Wada, Nobuhiko, Hiroshi Toriyama, Hiromi T. Tanaka and Fumio Kishino, "Reconstruction of an Object Shape from Multiple Incomplete Range Data Sets Using Convex Hulls," *Computer Graphics International '93*, Lausanne, Switzerland, June 21-25, 1993, pp. 193–203.