Survey of Isosurface Extraction and Surface Reconstruction

Ivan Lee
CS 598 MJG

1. Introduction

There are numerous ways to scan in a model: Magnetic Resonance Imaging (MRI), Computed Tomography (CT), laser range scanners, and Positron Emission Tomography (PET) are just a few examples. With these scans come large amounts of data that represent the model of interest. Consequently, this data can be a mix of unorganized points with no explicit information on connectivity or a series of images with coloring to represent different tissue in the human body. With this data, the question at hand is how to connect this data together to reconstruct the original model.

The difference between isosurface extraction and surface reconstruction was hinted above. For MRI, CT, and PET scans, a series of slices are created, each of which are coded by varying color or grayscale values. Each of these values represent layers of a function, which, when combined, creates a grid of isosurface values that are essentially a discretization of a scalar field, $F(x,y,z)$. The goal of isosurface extraction is to extract the surface representing all points in this scalar field where $F(x,y,z) = c$, where c is the desired isosurface contour constant. In contrast, laser scanners measure the distance from the scanner to wherever the laser strikes the surface, resulting in a series of points that rest on the surface. Surface reconstruction deals with building faces from these unconnected vertices to recreate the scanned model.

2. Desired Algorithm Properties

As more and more research is conducted, algorithms have begun to develop a series of desirable properties. Curless and Levoy [2] outline one such list for mesh producing algorithms to possess. Though these were outlined in the context of surface reconstruction, many of them apply to isosurface extraction as well:

- *Time and space efficiency*: An algorithm lacking this quality makes it impractical to use in the field and only valid to discuss in theory.
- *Robustness*: Algorithms not handling special cases will also be unusable.
- *Limited assumptions*: Assumptions take away from the robustness of an algorithm; however, some may be unavoidable for the proposed algorithms. Assumptions about the geometry's topology can lead to significant shortcomings.

3. Definitions

Given a surface, there are a number of scalar functions that can represent the surface. The requirement of the function, f, is that $f(x,y,z) = 0$ for any point on the surface. For points off of the surface, there are no restrictions. One such function is a *scalar distance field* in which the value at a given point is the distance to the closest point on the surface. A *directed distance field*, as defined by Kobbelt et al. [9], stores the distance to the surface along each of the coordinate axes.

Given a set of vertices, V, the *Voronoi cell* of a vertex, v_i, is the region of space where all points in that region are closer to v_i than any of the other vertices in V. A *Voronoi diagram* (figure 1) is the partitioning of a space containing V into these cells. A *Voronoi vertex* in $\mathbf{R}^d$ is a

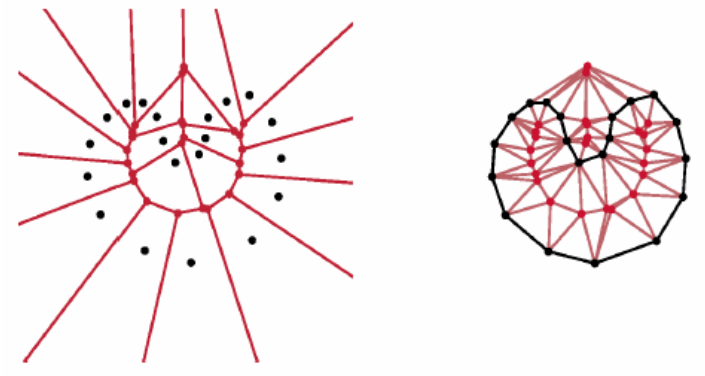

Figure 1: A Voronoi diagram (left) and its triangulation with the Voronoi vertices (right)
Amenta et al. [4]

point that is equidistant to d+1 sample points. These Voronoi vertices can be located in the Voronoi diagrams where two Voronoi cell boundaries intersect.

The dual of the Voronoi diagram is the *Delaunay triangulation*. A way of finding the Delaunay triangulation is triangulating the vertices such that the circumcircle of all triangles do not contain any other vertices. This accomplishes minimizing the maximum internal angle of all the triangles and results in a "nice" triangulation. Another nice property is that the boundary of the triangulation is the convex hull of the vertices. One way to find the triangulation in $\mathbf{R}^2$ is to project (x,y) to the paraboloid $x^2 + y^2$, find the convex hull of the projection, and project it back down to the plane.

The *medial axis* is the set of points that is the closest point to more than two points on the surface. In two dimensions, the medial axis can be approximated by the Voronoi vertices, however this extension cannot be made so easily to three dimensions. For this, we introduce the idea of *poles*, which are the two boundary vertices of a Voronoi cell that are farthest from the point the cell is built around. These poles are usually on opposite sides of the surface, though some special cases of the crust algorithm may not enforce this requirement. These poles approximated the medial axis in three dimensions.

4. Isosurface Extraction

Given a scalar field function, $f(x,y,z)$, the goal of isosurface extraction is to create a mesh representing a contour surface $f(x,y,z) = c$. The Marching Cubes algorithm (Lorensen and Cline [11]) provides one solution to this problem that offers a few new features over previous algorithms.

4.1. Previous Work

Keppel [8] started on the contour surface and progressively connected triangles on the surface between adjacent slices. However, an isosurface with more than one component would cause ambiguities for connecting contours. Farrel [5] and Hohne and Bernstein [7] used ray casting to find the surface and shading it with hue gradients or grayscales. A disadvantage of these ray casting algorithms is that they result in approximate shading with an unnormalized gradient.

4.2. Marching Cubes

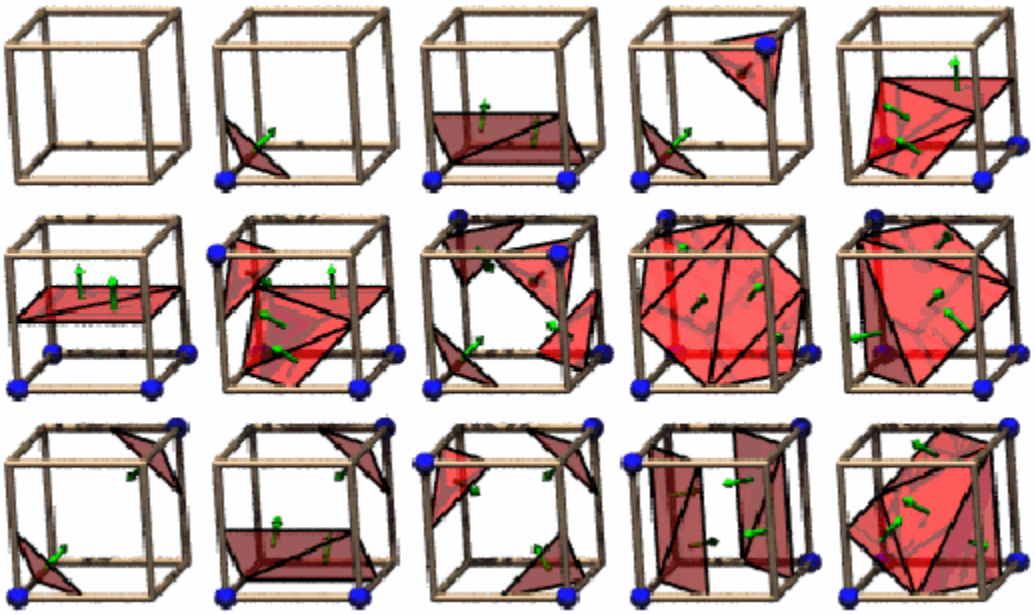

Figure 2: The 15 possible combinations. http://www.exaflop.org/docs/marchcubes/ind.html

The Marching Cubes algorithm partitions the space containing the scalar field into cubes. For a given scalar value, *c*, each vertex in the voxel grid is assigned a value of 1 if it is greater than or equal to the value and 0 otherwise. This creates 2^8 possible polygonizations of the cube, which can be reduced using rotational symmetry and swapping inside and outside vertices to reduce it down to 15 cases, which can be seen in Figure 2. A lookup table is created, which can be indexed by creating an 8 bit index from each vertex of the cube whose entry in the table points to the polygonization.

The normal at a surface point can be calculated by estimating the gradient of the scalar field function at that point. To estimate the gradient of a point within a cube, the gradient at each vertex of the cube is estimated by taking central differences of the scalar field:

$$G_x(i,j,k) = \frac{F(i+1,j,k) - F(i-1,j,k)}{\Delta x}$$

$$G_y(i,j,k) = \frac{F(i+1,j,k) - F(i-1,j,k)}{\Delta y}$$

$$G_z(i,j,k) = \frac{F(i+1,j,k) - F(i-1,j,k)}{\Delta z}$$

These central differences create a unit normal vector at each of the cube's vertices, which can then be interpolated to the surface point of concern within the cube.

Some initial enhancements were proposed in Lorensen and Kline [11] to speed up the original algorithm. One of these is to take advantage of the spatial coherence by reusing the values from adjacent cubes. By doing this, for all cubes other than those on the boundary, only three of the edges need to calculate the position along the edge where the surface intersects it. This provides a speedup factor of 3. Also, the ability to create CSG objects with Marching

Cubes was added by allowing Boolean operations and creating a truth table to see if a triangle should appear in the cube being evaluated.

The Marching Cubes algorithm produces some artifacts when generating a mesh. Some aliasing occurs at sharp edges on the model. This occurs because the algorithm encloses a cube around the sharp edge and no matter how fine the cube size is, the edge or corner is still sharp in the cube. The implication of this is that the normal on the surface in the cube does not converge to the surface's normal, which does happen for smoother portions of the surface, unless the grid happens to line up directly with one of the sharp edge's faces. Also, with the original algorithm, a uniform sampling is done so that there is a uniform distribution across the surface despite varying levels of detail on the surface. Additionally, this causes a uniform aliasing that vaguely represents the grid's cubic structure. These have all been topics of research in other papers, such as in the extended Marching Cubes algorithm explained in section 4.3.

4.3. Extended Marching Cubes

In the original Marching Cubes algorithm, sharp edges create undesirable artifacts. In Kobbelt et al [9], additional vertices are added to account for these sharp features.

To accomplish this, a couple modifications were made to the original algorithm. Instead of using the scalar distance field, the directed distance field, as defined in section 3, obtains more information than if the algorithm were to just interpolate the scalar values. As seen in figure 3, using the scalar distance field does not give information of the direction to the surface, and a linear interpolation of the values results in an errant estimation of the intersection point of the surface with the edge. Instead, the directed distance field actually provides the exact intersection point with the gird axis. Another modification is a feature detection algorithm that adds vertices to the mesh to bring out these features.

For the algorithm itself, the addition occurs when using the local information of the directed distance field. A feature detection algorithm is used that compares the normals of neighboring surfaces of a cell, and if the angle of the cone spanned by these normals is greater than some threshold, then we tag the cube to have a feature. Once this detection has been made, the normals at these points are used to find an intersection between the tangent planes along the surface in the given cube instead of directly connecting the edge intersection points. This intersection creates a new sampling point which results in an increased convergence rate compared to just a normal linear interpolation between edge intersections. To further differentiate features, the maximum deviation of the normals from each other is calculated, and a threshold is used to differentiate between edges and corners. A final step flips edges between adjacent cells with features present. An assumption made in this algorithm is that there is at most one sharp feature in each cell. With this assumption, these sharp features can be assumed to be connected between adjacent feature cells, and thus this edge flipping creates the desired edge or corner.

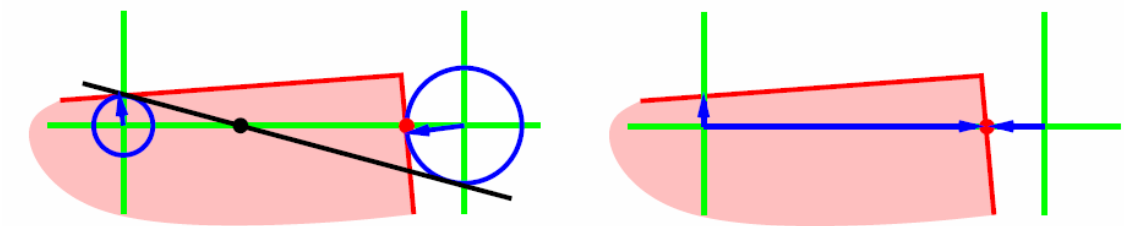

Figure 3: Computing the intersection location of the surface with the cell edges. The left figure shows uses scalar distance fields, and the right uses directed distance fields.
(Kobbelt et al. [2]

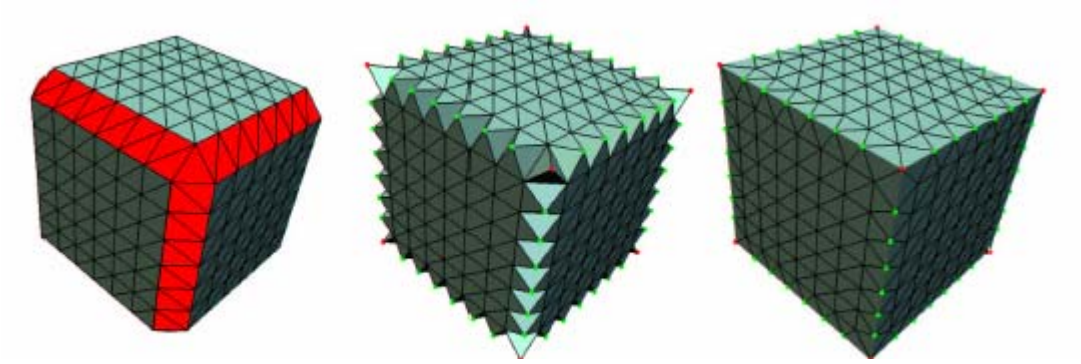

Figure 4: The additional steps of the extended Marching Cubes. (Kobbelt et al. [2])

The advantages of this algorithm over the original one are straightforward. It was designed for feature detection, thus sharp edges and corners can finally be handled adequately. Also, the use of directed distance fields provides the algorithm with much more information without much additional computational cost. However, the feature detection step adds computation time to the original algorithm, as was expected. Also, since sample points are added, the resulting meshes are larger than those produced by the original Marching Cubes.

5. Surface Reconstruction

In contrast, when not given a function, but rather a series of unorganized points, the problem to solve is how to connect these points. Here, the methods of Amenta et al. [1] and Curless and Levoy [2] are explained. The difference distinguishing the two is that the crust algorithm of Amenta et al. just assumes a set of points whereas the Curless and Levoy paper assumes a series of range images taken from different angles of the model.

5.1. Crust Algorithm

Prior to this paper, a few algorithms concentrated on constructing a model from unorganized points (Edelsbrunner et al. [3,4], Curless and Levoy [2]). Edelsbrunner et al. used alpha shapes in one such algorithm. The idea of alpha shapes is a generalization of the convex hull for a series of points. In $\mathbf{R}^2$, given an alpha parameter, a circle is drawn around each vertex and all vertices whose circles intersect are connected. An intuitive way of thinking of this is to imagine each vertex to be a rock and the rest of the space to be filled with Styrofoam. Then, with a Styrofoam eraser of radius alpha, erase as much of the Styrofoam as possible without intersecting any of the rocks. Vertices adjacent to cavities made by the eraser should be connected to create the alpha shape. A disadvantage of this method is that the experimental parameter alpha must be determined. This alpha is constant over the entire surface, thus a model of varying point density over the surface will not be appropriately shaved down by the algorithm.

The method of zero sets takes a set of input points and generates an implicit signed distance function for the space. This distance function passes through the Marching Cubes algorithm to output a model. An example of this is the Curless and Levoy paper described in section 5.2. This algorithm, however, produces an approximation of the original set of points, rather than an interpolation like the proposed crust algorithm.

Finally, Delaunay sculpting computes the Delaunay triangulation of the set of points and progressively removes tetrahedra from the polygonization by using an application-defined metric function to rank the tetrahedra. The crust algorithm and alpha shapes are essentially enhancements of Delaunay sculpting.

The algorithm itself starts off by computing the Voronoi diagram of the set of points. For each sample point, the poles are calculated for each sample point. These poles are then added to the original set of points and are used to recalculate the Delaunay triangulation. All triangles in which all three vertices are from the original set of sample points are kept, throwing out all triangles whose vertices includes a pole. The poles approximate the medial axis, and this secondary triangulation allows for the medial axis to block triangles from being created across the inside of the surface.

Once this triangulation is formed, a normal filtering step deletes all triangles whose normals differ by a certain threshold from the direction vector between the triangle vertices and the poles. This case occurs when the model actually contains to components and the previous steps connect the two components with a thin sheet of triangles. The final step is to assure the model is a manifold by orienting triangles so they are consistent with their neighbors and to remove sharp dihedral edges. To estimate the normal at a vertex, a neighborhood of the point is chosen and the covariance matrix is constructed. The eigenvector of the covariance matrix with the smallest eigenvalue defines the least-squares optimal plane, which effectively estimates the normal.

As mentioned before, the advantages of this algorithm over the previous work include no need for experimental parameters since the triangulations are created adaptively to the curvature of the surface. Therefore, the crust algorithm does not require a uniform sampling of the mesh, and the only assumption taken is that the sampling is dense.

When this algorithm was first introduced, a major assumption that was made was that the sampling of points was dense. If this assumption is violated, undersampling can result in a hole. One instance of this happening is by imagining two sheets near each other. If the pole of one of the vertices of one sheet falls on the other sheet and the undersampling occurs around this pole, the algorithm assumes the pole means the surface should not be constructed in that region. This results in a hole. Similarly, at sharp edges, holes can occur. In the case that two sheets are perpendicular to each other, one of the poles on one of the sheets may be far off the surface, but when taking the pole on the other side of the surface, it may land on the surface, puncturing a hole. Lastly, boundaries may cause a few problems, but often if there are no other sample points around the boundary, these holes will be closed off by the algorithm, which is acceptable.

5.2. Complex Models from Range Images

A significant difference between the Curless and Levoy [2] and Amenta [1] is that the Curless and Levoy paper exploits the overall structure of the measured data. In the realm of range images, work has been done using Venn diagrams that identify overlapping regions and exploring methods of merging the regions together to form the model (Soucy and Laurendeau [12]). Another method shaves away repeated geometry regions and stitching up the non-overlapping regions (Levoy and Turk [10]). Other work includes creating depth maps and using different weights to create a zero set which is extracted (Grosso et al. [6], Succi et al. [13]).

The main idea of this algorithm is to generate an implicit function using a weighted average of the range images. These weights are computed to reflect the uncertainty of the range image measurement and can often be related to the angle between the surface's normal and the direction from the sensor to the surface point. Now that an implicit function has been created, an isosurface extraction algorithm such as Marching Cubes can be run to output a model. To enhance the speed of the algorithm, it would logical to move along the volume in the same area as the range image. To do so, a mapping is created between each voxel scanline and the range scanline.

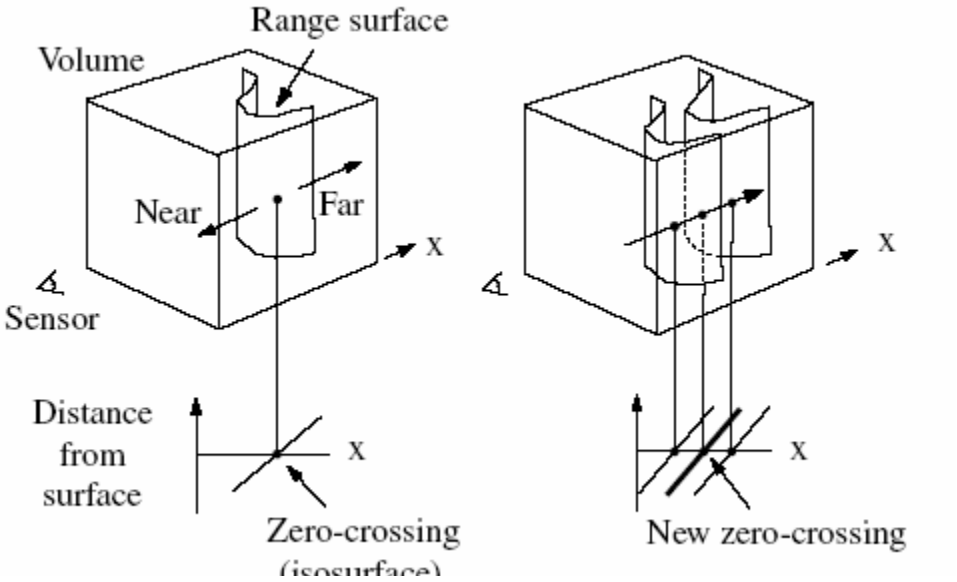

Figure 5: Using multiple range images to create an average of the surface. **Error! Bookmark not defined.**

Holes are handled to an extent with this algorithm. Each range image can see a portion of the model and the space surrounding it. To find out where these holes exist, each voxel can have one of three states: unseen, empty, or near the surface. The voxel space starts out with all cells marked as unseen, and as range images are looked at, those voxels near the surface are marked as so and all cells between the surface and the sensor are marked as empty. The boundary of the holes can be constructed by extracting the surface between empty and unseen regions.

The algorithm fails in a few cases, such as sharp corners or thin surfaces. This is due to the fact that more range images focusing on these corners would be needed to get an accurate weighting to replicate the sharp edge. Also, the current scanning technology is a limitation of the robustness of the algorithm, since more complex models would need more scanned images and crevices in the model will not be picked up easily.

6. Future Work

When the Marching Cubes algorithm was first published, the fact that there were a few ambiguities in some of the vertex configurations was not completely known. As this problem arose, the result was holes in the extracted surface and was solved by Marching Tetrahedra and adaptive grid refinement algorithms.

At the time of the extended Marching Cubes algorithm, the use of balanced octrees was proposed to help maintain a varying degree of refinement in the model. Additionally, changes were expected to be made to improve the speed of the Marching Cubes algorithm by exploiting the parallelism of the algorithm.

To take advantage of the Voronoi-based surface reconstruction, the problems of boundaries and sharp edges needed to be fixed. This was accomplished by detecting undersampling regions and adding points to compensate. One of the new research ideas was to use this algorithm for lossless mesh compression. This would allow for the just the vertices to be stored with no data on the connectivity. To decompress the vertices into a mesh, the algorithm is run to construct the surface. Also, the idea of using sample points with the additional information of the normals at these points will relax the assumption that the sampling is very dense. The idea of co-cones was introduced that creates a cone aligned with the poles which removes one Voronoi diagram

computation along with the normal trimming step. The power crust has been developed as a combination of power diagrams and polar balls which approximate the medial axis to separate the inside and outside of the surface.

For the creation of complex models from range images, some future directions proposed at the time of the paper also starts with basic extensions of improving execution time and exploiting the parallelism of the algorithm. Also, the problem of sharp edges and thin surfaces has been proposed to be fixed by considering the estimated normals of the surfaces.

7. References

[1] N. Amenta, M. Bern, and M. Kamvysselis. A new Voronoi-based surface reconstruction algorithm. In *Proceedings of SIGGRAPH 98*, pp. 415-422, July 1998.

[2] B. Curless and M. Levoy. A volumetric method for building complex models from range images. In *Proceedings of SIGGRAPH 96*, PP. 303-312, 1996.

[3] H. Edelsbrunner, D.G. Kirkpatrick, and R. Seidel. On the shape of a set of points in the plane. *IEEE Transactions on Information Theory*, pp. 551-559, 1983.

[4] H. Edelsbrunner and E.P. Mücke. Three-dimensional alpha shapes. *ACM Transactions on Graphics*, pp. 43-72, 1994.

[5] E. J. Farrell. Color display and interactive interpretation of three-dimensional data. *IBM J. Res. Develop 27*, pp 13-19, August 1983.

[6] E. Grosso, G. Sandini, and C. Frigato. Extraction of 3D information and volumetric uncertainty from multiple stereo images. In *Proceedings of the 8th European Conference on Artificial Intelligence*, pp. 683-688, August 1988.

[7] K. H. Hohne and R. Bernstein. Shading 3D images from CT using gray-level gradients. *IEEE Transactions On Medical Imaging MI-5*, pp. 45-47, March 1986.

[8] E. Keppel. Approximating complex surfaces by triangulation of contour lines. *IBM J. Res. Develop 19*, pp. 2-11, January 1975.

[9] L. Kobbelt, M. Botsch, U. Schwanecke, and H.-P. Seidel. Feature sensitive surface extraction from volume data. In *Proceedings of SIGGRARPH 2001*, pp. 57-66, 2001

[10] M. Levoy and G. Turk. Zippered polygon meshes from range images. In *Proceedings of SIGGRAPH 94*, pp. 311-318, July 1994.

[11] W. Lorensen and H.Cline. Marching Cubes: A high resolution 3D surface construction algorithm. In *Proceedings of SIGGRAPH 87*, pp. 163-169, 1987.

[12] M. Soucy and D. Laurendeau. A general surface approach to the integration of a set of range views. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 344-358, April 1995.

[13] G. Succi, G. Sandini, E. Grosso, and M. Tistarelli. 3D feature extraction from sequences of range data. In *Robotics Research, Fifth International Symposium*, pp. 117-127, August 1990.