Representation and Recognition of Free-Form Surfaces

Herve Delingette **Martial Hebert** **Katsushi Ikeuchi**

November 1992
CMU-CS-92-214

Computer Science Department
Carnegie Mellon University
Pittsburgh, PA 15213

This research was supported in part by the DARPA Image Understanding Program, through ARPA Order No. 4976, and monitored by the Air Force Avionics Laboratory under contract F33615-87-C-1499, and in part by DARPA, under contract DACA 76-89-C-0014 monitored by the Army Topographic Engineering Center.

The views and conclusions contained in this report are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Defense Advanced Projects Agency or the U.S. Government.

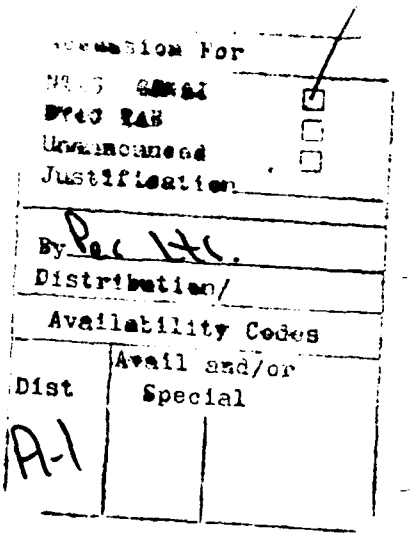
Keywords: Object Recognition, Range Data, Deformable Surfaces, Free-Form Surfaces, Spherical Representations

Representation and Recognition of Free-Form Surfaces

Herve Delingette **Martial Hebert** **Katsushi Ikeuchi**

November 1992
CMU-CS-92-214

Computer Science Department
Carnegie Mellon University
Pittsburgh, PA 15213



This research was supported in part by the DARPA Image Understanding Program, through ARPA Order No. 4976, and monitored by the Air Force Avionics Laboratory under contract F33615-87-C-1499, and in part by DARPA, under contract DACA 76-89-C-0014 monitored by the Army Topographic Engineering Center.

The views and conclusions contained in this report are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Defense Advanced Projects Agency or the U.S. Government.

ABSTRACT

We introduce a new surface representation for recognizing curved objects. Our approach begins by representing an object by a discrete mesh of points built from range data or from a geometric model of the object. The mesh is computed from the data by deforming a standard shaped mesh, for example, an ellipsoid, until it fits the surface of the object. We define local regularity constraints that the mesh must satisfy. We then define a canonical mapping between the mesh describing the object and a standard spherical mesh. A surface curvature index that is pose-invariant is stored at every node of the mesh. We use this object representation for recognition by comparing the spherical model of a reference object with the model extracted from a new observed scene. We show how the similarity between reference model and observed data can be evaluated and we show how the pose of the reference object in the observed scene can be easily computed using this representation.

We present results on real range images which show that this approach to modelling and recognizing three-dimensional objects has three main advantages: First, it is applicable to complex curved surfaces that cannot be handled by conventional techniques. Second, it reduces the recognition problem to the computation of similarity between spherical distributions; in particular, the recognition algorithm does not require any combinatorial search. Finally, even though it is based on a spherical mapping, the approach can handle occlusions and partial views.

Contents

List of Figures

1 Introduction

Recognition of curved objects is one of the key issues in computer vision. It is a problem not only in traditional applications such as industrial object recognition and face recognition but also in emerging applications such as navigation and manipulation in natural environments. To aid in overcoming this problem, we have designed a new approach that uses as a starting point a combination of several traditional object recognition and representation methods.

Traditionally, there are two ways to represent objects for recognition: *local* and *global*. Local methods attempt to represent objects as a set of primitives such as faces or edges. Most early local methods handle polyhedral objects and report effective and encouraging results. Representative systems include [12][20][14]. Few systems can handle curved surfaces. Some systems include early work in which primitive surfaces enclosed by orientation discontinuity boundaries are extracted from range data [21]. Other systems determine primitive surfaces which satisfy planar or quadric equations [9]. Techniques based on differential geometry such as [3] segment range images using Gaussian curvatures. More recent local techniques use points of interest and edges of surfaces to match observed surfaces with stored representation [23]. These local methods, however, are noise-sensitive and are still limited in reliably extracting primitives of curved objects from input images.

The global methods assume one particular coordinate system attached to an object and represent the object as an implicit or parametric function in this coordinate system. The resulting representation is global in that the implicit function represents the entire shape of the object or of a large portion of the object. The generalized cylinder (GC) is a representative of this group. A generalized cylinder is defined as an axis, a cross-sectional shape and its sweeping rule along the axis. Although encouraging results have been obtained in recognizing GCs in intensity images by minimizing the distance between observed and predicted occluding edges, using generalized cylinders for recognition is difficult due to the difficulty of extracting GC parameters from input images.

Superquadrics (SQ) representation also belongs to the class of global representations [22]. Superquadrics are generalizations of ellipsoids. Object representations are built by fitting an implicit equation to a set of input data points. Recognition using SQs proceeds by comparing the parameters of the SQs extracted from the scene with the SQs stored in the model. The SQs represent a limited set of shapes which can be extended by adding parameters to the generic implicit equation of SQs. This limitation has the undesirable effect of making the fitting process much more expensive and numerically unstable. A possible extension is to segment objects into sets of superquadrics [10], although the computational complexity of the scene analysis may become prohibitive. An interesting attempt to handle a large class of natural objects in discussed in [4] in which multiple surface representations, ranging from quadrics to superquadrics to generalized cylinders, are used. The type of representation is selected based on the level of detail available from the range image.

EGI and CEGI map surface orientation distributions to the Gaussian sphere [13][18][15]. Since the Gauss map is independent on translation, the representation is quite suitable to handle convex curved objects. In this case, recognition proceeds by finding the rotation

that maximizes the correlation between two EGIs [13][7]. However, when part of the object is occluded, those techniques cannot reliably extract the representation.

Recently, new approaches for modeling objects have been developed. These approaches are based on the idea of fitting a bounded algebraic surface of fixed degree to a set of data points [24][25]. With this representation, recognition proceeds by comparing the polynomials describing observed and stored surfaces, although it is not yet clear how the comparison would be performed. Using algebraic surfaces is convenient because powerful tools can be used to compute limbs and other properties of the object. Recognition proceeds by comparing invariant properties computed from the algebraic equations of observed and reference surfaces [11]. Although encouraging results have been obtained in this area, more research is needed in the areas of bounding constraints, convergence of surface fitting, and recognition before this approach becomes practical. Occlusion remains a problem since there is no guarantee that the polynomial computed from a partial view is similar to the polynomial computed from a complete model of the object. For a survey of other techniques can be used for global surface fitting, see [5].

All these approaches attempt to fit some known parametric surface, either locally or globally, to the object. Consequently, these approaches tend to limit the set of shapes that can be represented and recognized. Moreover, the cost of building the representations from data sets increases rapidly as parameters are added to expand the set of allowable shapes. To address these two problems, another class of approaches attempts to match sets of points directly without any prior surface fitting. An example is the work by Besl [2] in which the distance between point sets is computed and minimized to find the best transformation between model and scene. This approach has many advantages since it does not require any surface segmentation or surface fitting and it does not require to search for an explicit correspondence between model and scene features for recognition. Recent results show that these algorithms can perform remarkably well by using new numerical techniques for minimizing distances between two arbitrary point sets. The main drawback of this approach is that, like any minimization technique, it is not guaranteed to find the global optimum especially if the scene contains occlusions, different point density in model and scene representation, and large number of extra points from different objects.

Our approach begins with a combination of the point set matching and of the original EGI approach. As in the case of the point set matching, we want to avoid fitting analytical surfaces to represent an object. Instead, we use a representation that simply consists of a collection of points, or nodes, arranged in a mesh covering the entire surface of the object. This has the advantage that the object can have any arbitrary shape, as long as that shape is topologically equivalent to the sphere. To avoid problems with variable density of nodes on the mesh, we need to define regularity constraints that must be enforced when the mesh is built. Constructing meshes that fit input data and that satisfy some constraints is possible based on the optimization techniques originally introduced in [26] and [16]. We use an extension of the deformable surfaces algorithms introduced in [8] to compute the meshes. As in the EGI algorithms, each node of the mesh is mapped onto a regular mesh on the unit sphere, and a quantity that reflects the local surface curvature at the node is stored at the corresponding node on the sphere. Instead of using a discrete approximation of the curvature, we develop a new curvature indicator, the *simplex angle*, which is entirely defined from a node and its neighbors in the mesh without any reference to the underlying

continuous surface. We call the corresponding spherical representation the *Simplex Angle Image* (SAI). Finally, we define the regularity constraints such that if $\mathcal{M}$ is the mesh representing an object, and $\mathcal{M}'$ is the mesh representing the same object after transformation by a combination of rotation, translation, and scaling, then the corresponding distributions of simplex angles on the spherical representations S and S' are the same up to a rotation. In other words, the SAI is an invariant representation. Therefore, to determine whether two objects are the same, we need only compare the corresponding spherical distributions. The overall approach is illustrated in Figure 1: A regular mesh is computed from input object description, sensor data or CAD model, simplex angle is computed at each node of the meshes and the meshes are mapped onto a sphere, the SAI. If a rotation between the two spherical images exists, the two meshes correspond to the same object. This approach is similar in principle to the EGI approach. However, one fundamental difference is that a unique mesh, up to rotation, translation and scale, can be reconstructed from a given SAI. In the case of the EGI, this property is true only for convex objects. Another fundamental difference is that the SAI preserves connectivity in that patches that are connected on the surface of the input object are still connected in the spherical representation. The latter is the main reason why our approach can handle arbitrary non-convex objects. Connectivity conservation is also the reason why the SAI can be used for recognition even in the presence of significant occlusion, as we will see later in the paper, whereas EGI and other global representation cannot.

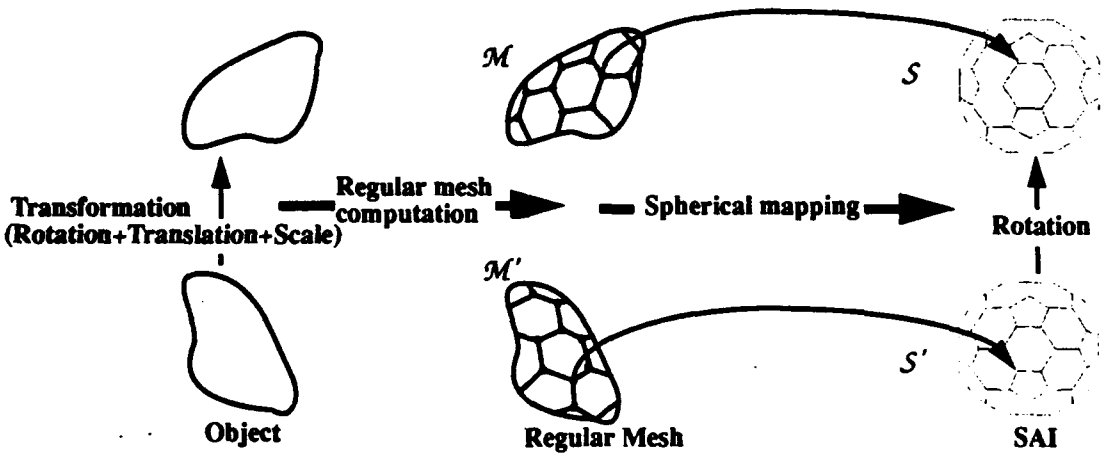


Figure 1 Object Recognition Using SAIs

Another way to describe the properties of the SAI is in terms of *intrinsic coordinate systems*. An intrinsic coordinate is such that any given surface point has the same coordinates regardless of the orientation, position, and scale of the underlying object. In the case of two-dimensional contours, intrinsic coordinate systems are very easy to define. For example, they are the basis of geometric hashing techniques [17]. For three-dimensional surfaces, a general definition of intrinsic coordinate systems on curved objects is much more difficult to define. For example, the geodesics of a surface can be used to define an intrinsic coordinate system. Still other efforts focus on lines of curvatures and other differential geometry invariants [6]. The problem with these approaches is that they are based on defi-

nitions and properties that are valid for *continuous* surfaces, whereas we typically have to handle *discrete* surfaces from sensor data.

To describe our approach, we have organized the paper as follows. In Section 2, we describe a simple representation of closed 2-D curves which then extend to three dimensional surfaces in Section 3. In both sections, we investigate the notions of global and local regularity, and of simplex angle. The two-dimensional representation is fairly standard and is used here only as introduction to the definition of the equivalent representation in three dimensions. In Section 4, we show how to obtain SAIs from range data. In Section 5, we describe the SAI matching. In Sections 3 to 5 we describe the fundamentals of the SAI algorithms in the case of complete object models. We address the problem of occlusion and partial models in Section 6. In that section, we also present several results of recognition in complex scenes and a discussion of performance and robustness of the recognition algorithm.

2 Intrinsic Representation of 2-D Curves

A standard approach to representing and recognizing contours is to approximate contours by polygons, and to compute a quantity that is related to the curvature of the underlying curve. The similarity between contours can then be evaluated by comparing the distribution of curvature measurement at the vertices of the polygonal representations. Under certain conditions, the curvature distribution can be mapped unambiguously on the unit circle, allowing for a representation that is independent of orientation and position of the contour. In this section, we introduce the basic concepts that can be used to manipulate polygonal representations of contours. Starting with the definition of a curvature indicator, we then define conditions of local and global regularity for building intrinsic representations of contours. The concepts discussed in this section are well known and have been studied and applied extensively in previous works. Our purpose here is to introduce them in a way that facilitates their extension to three-dimensional surfaces. In particular, the circular mapping is equivalent to the classical φ-s representation.

Rather than attempting to approximate the curvature of a discrete curve at each node of the polygonal approximation, we use a different quantity, the angle φ between consecutive segments (Figure 2), which is related to the curvature but has more desirable properties in dealing with discrete representations. The relation between φ and the curvature k is $k \approx \varphi / l$ as l becomes small, or, equivalently, as the density of points increases. Like the curvature, the angle φ is independent of rotation and translation. Unlike the curvature, φ is also independent of scaling.

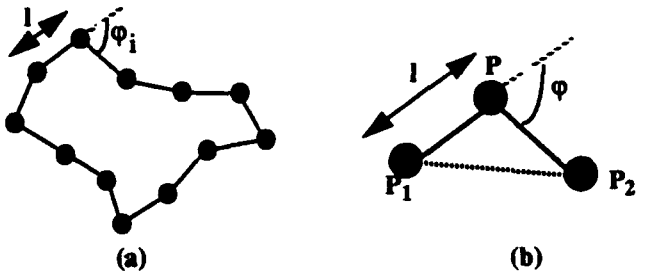


Figure 2 Discrete Curvature Measure of a 2-D Curve

One problem is that if the lengths of the segments representing the curve are allowed to vary, the value of φ depends not only on the shape of the curve but also on the distribution of points on the curve. In particular, it is important for the same curve shape to generate the same value of φ to unable the comparison of discrete curves. One way to avoid this problem is to impose a *local regularity* condition on the distribution of vertices. The local regularity condition simply states that all the segments must have the same length. This can be restated as a local property by saying that the length of a segment must be equal to the length of each neighboring segments. Another geometric definition of this condition is illustrated in Figure 3. The condition that the length of the two segments PP_1 and PP_2 are the same is equivalent to the condition that the projection of node P on the line joining its two neighbors P_1 and P_2 coincides with the center of P_1 and P_2. This is obviously a more complicated way of formulating the simple regularity condition, but it will become useful when we extend this notion to three dimensions. One consequence of the regularity condition is that there is only one degree of freedom in the polygonal representation in that, for a given number of vertices, specifying one vertex on the curve determines uniquely the locations of all the other nodes along the curve.

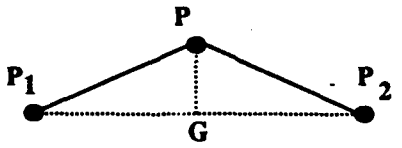


Figure 3 Local Regularity

The last step in representing two-dimensional contours is to build a circular representation that can be used for recognizing contours. Let us assume that the contour is divided into N segments with vertices $P_1,..,P_N$, and with corresponding angles $\varphi_1,..,\varphi_N$. Let us divide the unit circle using N equally spaced vertices $C_1,..,C_N$. Finally, let us store the angle φ_i associated with P_i at the corresponding circle point C_i (Figure 4). The circular representation of the contour is invariant by rotation, translation, and scaling. In other words, given the position of two vertices, a circular representation defines a unique polygonal contour. The two vertices are necessary to define the rotation, translation, and scaling. Conversely, as the density of points becomes large, the circular representations of two instances of the same contour are identical up to a rotation of the unit sphere. This property allows for comparing contours by deciding that two contours are identical if there

exists a rotation of the unit circle that brings their representation in correspondence (Figure 5). The unicity property is true because of the local regularity condition and because of the invariance of φ. Also, when comparing contours, the distribution of the vertices C_i on the circle must be uniform, that is the distance between consecutive vertices on the unit circle is constant. We refer to this property as *global regularity*. Global regularity can always be achieved in the case of planar curves. We mention it here because it becomes a non-trivial notion when we try to extend the discrete representation to 3-D surfaces.

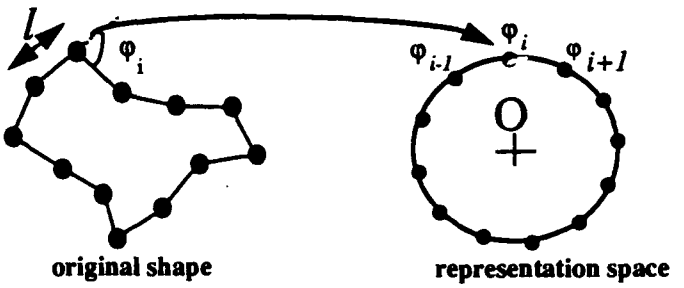


Figure 4 Mapping Between Shape and Circular Representation Space

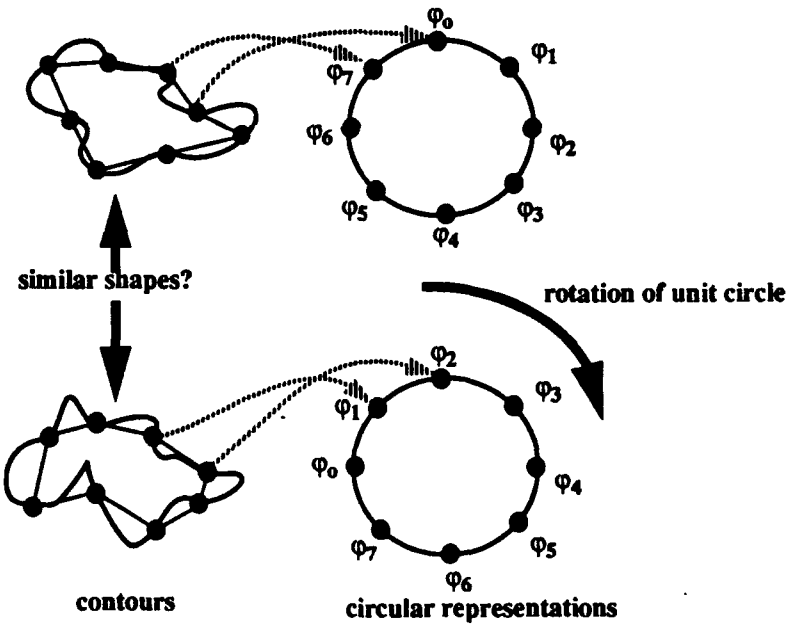


Figure 5 Comparing Contours in Representation Space

3 Intrinsic Representation of 3-D Surfaces

In this section we extend the concepts of curvature indicator, local and global regularity, and circular representation to three-dimensional surfaces. We consider the case of representing surfaces topologically equivalent to the sphere. (Cases in which only part of the surface is visible will be addressed in Section 6.) We follow the same approach as for two-

dimensional contours. We first define a discrete representation of surfaces, the equivalent of the polygonal representation, in Section 3.1. We introduce the three-dimensional equivalent of the concepts of global and local regularity in Sections 3.2 and 3.3, respectively. In Section 3.4, we propose a new indicator of curvature, the simplex angle, that is a direct extension of the angle used in the two-dimensional case. Finally, we define an intrinsic spherical representation as an extension of the circular representation of contours in Section 3.5. At the end of this section, we will have defined a representation that is invariant by translation and scale and is unique for a given object and a given resolution up to a rotation of the representation space. Detailed presentations of the basic results on semi-regular tessellations, triangulations, and duality can be found in [19][27][28].

3.1 Triangulation and Duality

The most natural discrete representation of a surface is a triangulation, that is a polyhedron with triangular faces whose vertices are on the surface. Each face defines a plane which is the local approximation of the surface. It is desirable for many algorithms to have a constant number of neighbors at each node. We use a class of meshes such that each node has exactly three neighbors. Such meshes can always be constructed as the dual of a triangulation. The dual of a triangulation is a graph with one node for each face of a triangulation. Nodes are connected in the dual graph if they correspond to connected faces in the original triangulation. Figure 6 shows a triangulation and its dual. In switching from a triangulation to its dual, the property of planarity of the faces is lost since faces, defined as the cycles with the minimum number of vertices, may have more than three vertices. Therefore, the dual mesh should be viewed as a graph of points with the desired connectivity; the triangulation may be viewed as a polyhedral approximation of the object. The dual of any triangulation is a graph of degree three. In the remainder of this paper, we will use only dual meshes with the understanding that they can be derived from an initial triangulation.

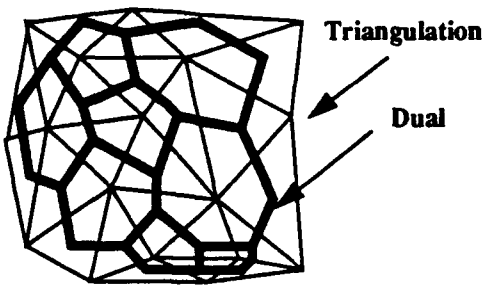


Figure 6 Triangulation and Dual Mesh

3.2 Global Regularity and Mesh Topology

As mentioned in the previous section, global regularity can be easily achieved in two dimensions since a curve can be always divided into an arbitrary number of segments of equal length. The equivalent in three dimensions would be a mesh covering a closed surface such that the distance between vertices is constant and is the dual of a triangulation,

that is, each node has exactly three neighbors. Extending the notion of global regularity to a mesh covering a two dimensional plane, there are three possible topologies: The triangular and hexagonal meshes which are dual of each other, and the square mesh which is its own dual. The problem is that, even though these meshes provide global regularity for an open surface, they cannot be extended to a close surface. In fact, tetrahedron, cube and dodecahedron (Figure 7).are the only regular triangulation-dual tessellations of a closed surface, corresponding to the triangular, square, and hexagonal topologies, respectively. Therefore, only approximate global regularity can be achieved in three dimensions.

The approach that we use is recursive subdivision of the dodecahedron which yields a mesh that is "almost" regular in that all but 12 pentagonal cells have hexagonal connectivity. If the number of cells is large enough, typically several hundred to a few thousand, the ratio of the number of regular hexagonal cells to the number of singular pentagonal cells becomes very small. Therefore, the mesh is almost regular for a large number of cells. In practice, a triangulation with the appropriate number of nodes is first constructed. The triangulation is built by subdividing each triangular face of a 20-face icosahedron into N^2 smaller triangles. The final mesh is built by taking the dual of the $20N^2$ faces triangulation, yielding a mesh with the same number of nodes. Figure 8 shows the mesh obtained by recursive subdivision of the dodecahedron for $N = 2$, 3, and 5. For the experiments presented in this paper, we used a subdivision frequency of $N = 7$ for a total number of nodes of 980.

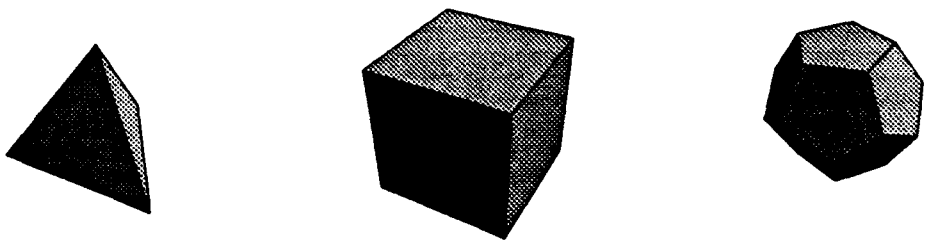
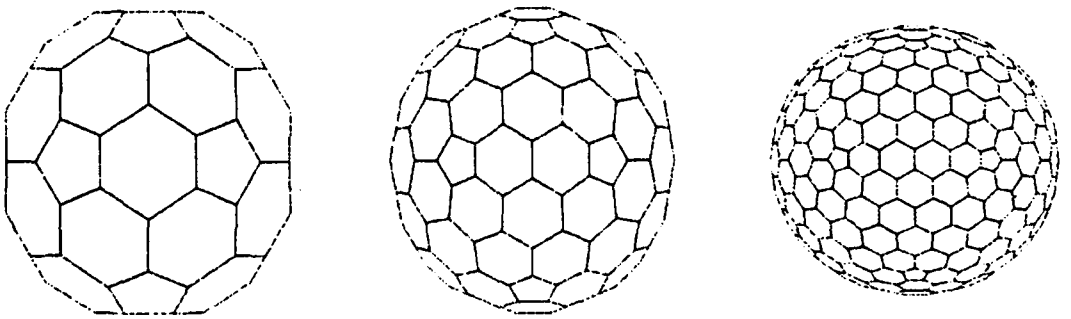


Figure 7 Regular Meshes



(a) N = 2 (b) N = 3 (c) N = 5

Figure 8 Meshes from Recursive Subdivision of Dodecahedron

3.3 Local Regularity

The next step in going from two to three dimensions is to define a notion of local regularity that leads to invariance properties of the mesh and curvature indicator definition similar to the properties used for 2-D curves. The definition of local regularity in three dimensions is a straightforward extension of the definition of Section 2. Let P be a node of the mesh, P_1, P_2, P_3 be its three neighbors, G be the centroid of the three points, and Q be the projection of P on the plane defined by P_1, P_2, and P_3 (Figure 9). The local regularity condition simply states that Q coincides with G. This is the same condition as in two dimensions, replacing the triangle (P_1, P_2, P) of Figure 3 by the tetrahedron (P_1, P_2, P_3, P). The local regularity condition is invariant by rotation, translation, and scaling because it is purely local and involves only relative positions of the nodes with respect to each other, not absolute distances.

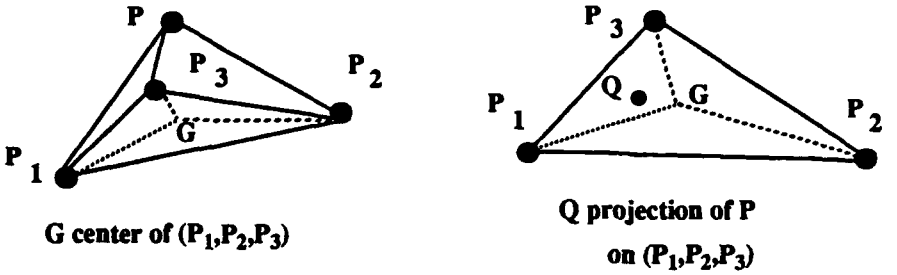


Figure 9 Local Regularity in Three Dimensions

3.4 Discrete Curvature Measure: Simplex Angle

The last step in building a discrete surface representation is to define an indicator of curvature that can be computed from a mesh with the appropriate regularity properties.We propose a definition in terms of angular variation between neighbors in the mesh according to the definition of Figure 2. We need to define some notations (Figure 10 (a)). Let P be a node of the mesh, P_1, P_2, P_3 its three neighbors, O the center of the sphere circumscribed to the tetrahedron (P, P_1, P_2, P_3), Z the line passing through O and through the center of the circle circumscribed to (P_1,P_2,P_3). Now, let us consider the cross section of the surface by the plane Π containing Z and P. The intersection of Π with the tetrahedron is a triangle. One vertex of the triangle is P, and the base opposite to P is in the plane (P_1,P_2,P_3) (Figure 10 (b)). We define the angle φ_o as the angle between the two edges of the triangle intersecting at P. By definition, φ_o is the discrete curvature measure at node P. It is easy to see that this definition is consistent with the 2-D definition since the geometry in the plane Π is the same as the initial geometry for a two dimensional curve illustrated in Figure 2. We call φ_o the simplex angle at P, since it is the extension to a three-dimensional simplex, the tetrahedron, of the notion introduced for a two-dimensional simplex, the triangle.

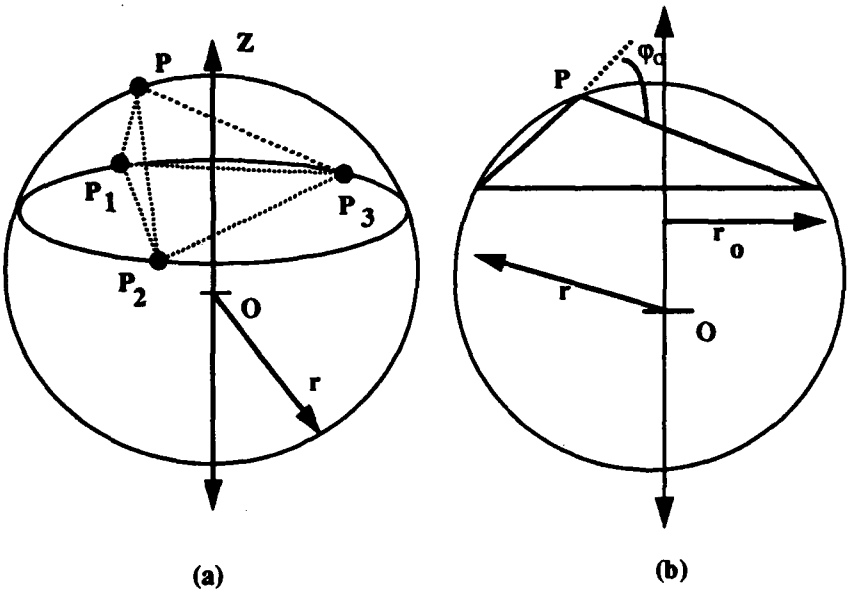
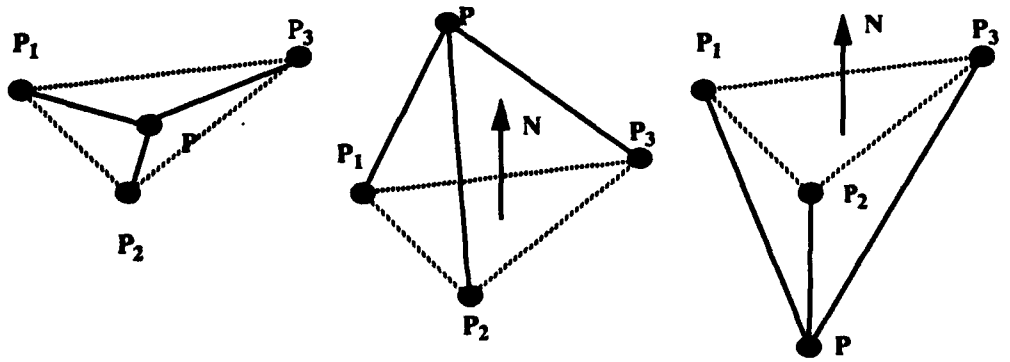


Figure 10 Definition of the Simplex Angle

The simplex angle varies between $-\pi$ and π. The angle is 0 for a flat surface, and is large in absolute value if ***P*** is far from the plane of its three neighbors. The simplex angle is negative if the surface is locally concave, positive if it is convex, assuming that the set of neighbors is oriented such that the normal to the plane they form is pointing toward the outside of the object (Figure 11). This behavior of the simplex angle corresponds to the intuitive notion of local "curvature" of a surface. Another desirable property is that the simplex angle is *sphere-invariant* in that φ_o remains the same no matter where ***P*** is located on the circumscribed sphere. In particular, this implies that if the nodes of the mesh are on a surface whose curvature is constant in a region, then the simplex angle will also be constant in this region no matter what the distribution of the points is. Finally, it is clear that the simplex angle is invariant by rotation, translation, and scaling.



(a) Flat plane: $\varphi_o = 0$ (b) Convex peak: φ_o close to π (c) Concave peak: φ_o close to $-\pi$

Figure 11 Typical Values of the Simplex Angle

There are two practical problems with this definition of the simplex angle. First, this definition requires the computation of a circumscribed circle and of a circumscribed sphere, both of which are expensive operations. This computation becomes a problem in the algorithm of Section 4 in which the angle has to be evaluated many times at every node of a 1000-node mesh. Second, the radius of the circumscribed sphere becomes infinite as φ_o vanishes. As a result, a direct implementation using the radius of the circumscribed sphere is expected to be numerically unstable in areas in which the mesh is nearly flat.

In contrast, we use a different approach that leads to an efficient and stable way to compute φ_o. The details of the algorithm are given in Appendix A.

In the rest of the paper, we will denote by g the function that maps a node to its simplex angle; the simplex angle φ_o at a node P will be denoted by $g(P)$.

3.5 Simplex Angle Image

We have extended the notions of regularity and simplex angle to three-dimensional surfaces; we can now extend the circular representation developed in two dimensions to a spherical representation in three dimensions. Let $\mathcal{M}$ be mesh of points on a surface such that it has the topology of the quasi-regular mesh of Section 3.2. Let $\mathcal{S}$ be a reference mesh with the same number of nodes on the sphere of unit one. We can establish a one-to-one mapping h between the nodes of $\mathcal{M}$ and the nodes of $\mathcal{S}$. The mapping h depends only on the topology of the mesh and the number of nodes. Specifically, for a given size of the mesh $M = 20xN^2$, where N is the frequency of the mesh (Section 3.2), we can define a canonical numbering of the nodes that represents the topology of any M-mesh. In other words, if two nodes from two different M-mesh have the same index, so do their neighbors. With this indexing system, $h(P)$, where P is a node of the spherical mesh, is the node of the object mesh that has correspond to the same index as P.

In the current implementation, the nodes are stored in a two-dimensional array. The first dimension of the array, sometimes called the *major* index, is between 1 and 20, the second dimension, sometimes called the *minor* index, is between 1 and N^2. For any node $P(major,minor)$, $h(P)$ is the point stored at the same location, $(major,minor)$, in the object mesh table. The connectivity table is computed only once for any given frequency.

Given h, we can store at each node P of $\mathcal{S}$ the simplex angle of the corresponding node on the surface $g(h(P))$. The resulting structure is a quasi-regular mesh on the unit sphere, each node being associated with a value corresponding to the simplex angle of a point on the original surface. By analogy with the Extended Gaussian Image, we call this representation the Simplex Angle Image (SAI). In the remainder of the paper, we will denote by $g(P)$ instead of $g(h(P))$ the simplex angle associated with the object mesh node $h(P)$ since there is no ambiguity.

If the original mesh $\mathcal{M}$ satisfies the condition of local regularity, then the corresponding SAI has several important properties. First, the SAI is invariant by translation and scaling of the original object, given a mesh $\mathcal{M}$. This condition is because the simplex angle itself is invariant by translation and scaling (Section 3.5), and because $\mathcal{M}$ still satisfies the local regularity condition after translation and scaling (Section 3.3).

The fundamental property of the SAI is that it represents an object unambiguously up to a rotation. More precisely, if $\mathcal{M}$ and $\mathcal{M}'$ are two meshes on the same object with the same number of nodes both satisfying the local regularity condition, then the corresponding SAIs S and S' are identical up to a rotation of the unit sphere. Strictly speaking, this is true only as the number of nodes becomes very large because the nodes of one sphere do not necessarily coincide with the nodes of the rotation version of the other sphere. (This problem is addressed in Section 5.1.) One consequence of this property is that two SAIs represent the same object if one is the rotated version of the other.

From this definition of the mapping h, we can now easily see the origin the property of connectivity conservation mentioned in the Introduction. If two nodes P_1 and P_2 are connected on the spherical mesh, then the two corresponding nodes $M_1=h(P_1)$ and $M_2=h(P_2)$ on the object mesh are also connected by an arc of the object mesh. The property holds because of the definition of h which depends only on the topology of the mesh, not on the positions of the nodes.

Another way to look at these properties of SAIs is in terms of unicity of representation. A given SAI defines a mesh size and a distribution of simplex angles. The unicity property is that an SAI represents a unique object mesh up to rotation, translation, and scale. The unicity property holds even in the case of arbitrary non-convex objects because of the connectivity conservation property. In fact, we will show in Section 4.4 that the object can be explicitly reconstructed from its SAI.

3.6 Alternate Definition of Simplex Angle

Other definitions of the simplex angle are possible. In particular, a common definition of a discrete curvature index is the angle defined as [1]:

$$\kappa = 2\pi - (\theta_1 + \theta_2 + \theta_3)$$

In this definition, the θ_is are the angles between the vectors joining the center point at which κ is calculated and its three neighbors (Figure 12). To be consistent with φ, the sign of κ is positive for a convex configuration in which the center point is in the positive side of the plane defined by its three neighbors, and negative for a concave configuration.

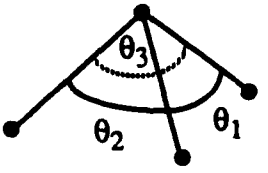


Figure 12 Definition of Alternate Curvature Angle κ

The behaviors of κ and φ as local shape varies are qualitatively the same. As an example, let us consider the configuration of Figure 13 in which three points form a right triangle with the lengths of two of the sides being equal to one, and in which a point is free to move on a line orthogonal to the plane of the three points. We denote by D the signed distance between the point and the plane. Figure 14 shows the change in κ and φ as D varies. In this figure, φ is scaled to be between -2π and $+2\pi$ to be consistent with κ. As expected, the shapes of the two curves are similar: both angles are close to zero as D vanishes, which

corresponds to a locally flat surface; and they increase monotonically as D increases, which corresponds to increasing local curvatures of the surface. The graphs show that κ and φ are essentially the same for large values of D. The main difference between the two occurs near the origin. To illustrate this difference, Figure 15 shows the graphs in the neighborhood of the origin. The difference is now apparent: κ has a zero tangent at the origin, while the φ graph exhibits a large slope at the origin. In practice, this means that κ cannot discriminate well between local configurations that correspond to relatively small values of D since κ varies very slowly. By contrast, because of the slope, different local configurations yield very different values of φ, even near the origin. In practice, most shapes have local configurations located toward the center part of the graph where the difference between the two angles is most pronounced. High values of κ and φ, where the two angles are essentially the same, occur mostly at isolated points of the surface. This is the main reason for selecting φ over κ to build our representation because we are interested in using the curvature index that best discriminates among shapes. From a computational standpoint, both angles require essentially the same amount of computation even though the geometric definition of κ is simpler.

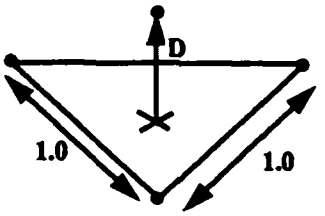


Figure 13 Reference Configuration for the Graphs of Figure 14 and Figure 15

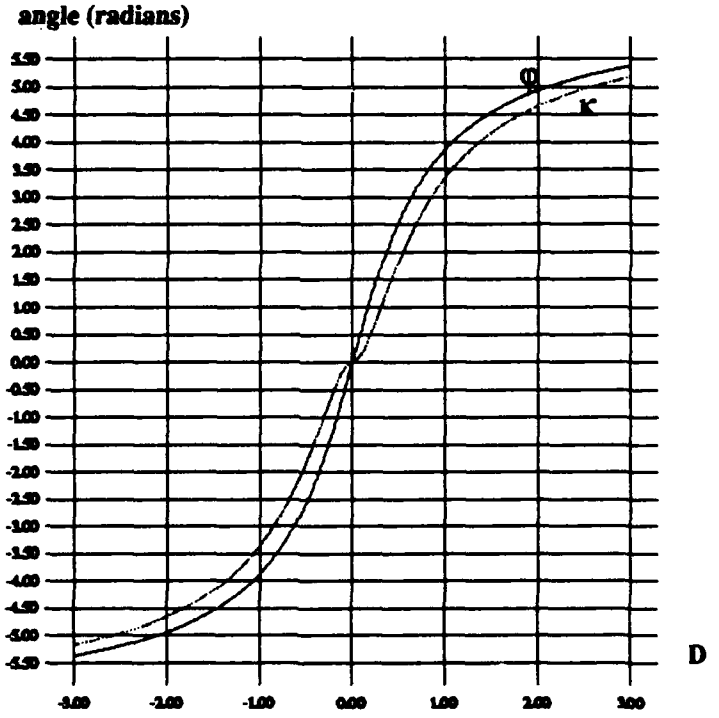


Figure 14 Values of φ and κ as Functions of Distance to Reference Plane

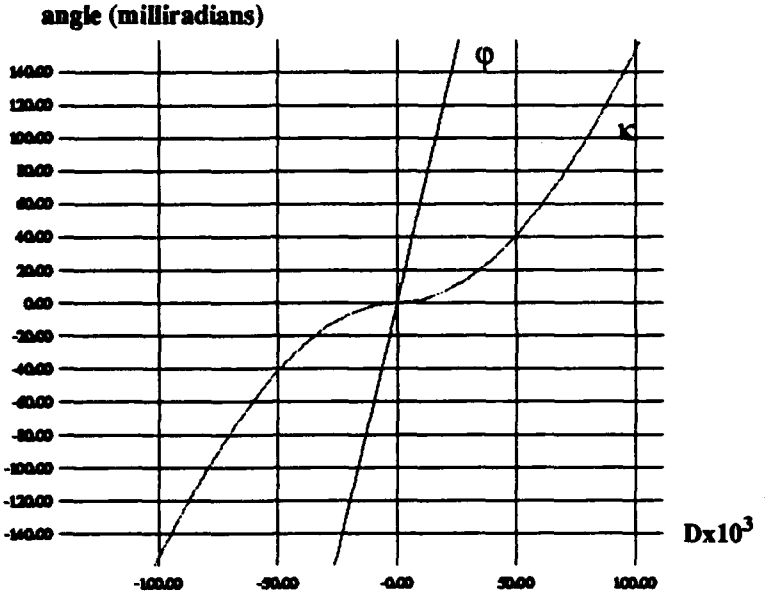


Figure 15 Values of φ and κ as Functions of Distance to Reference Plane Near the Origin

4 Building Intrinsic Representations from 3-D Data

In the previous sections, we have defined the notion of locally regular mesh and its associated SAI. In this section, we describe the algorithm developed for computing such a mesh from an input object. We assume that we have some input description of an object. The only requirement is that the input description allows for computing the distance between an arbitrary point in space and the surface of the object. Therefore, input object representations such as polynomial surface patches and polyhedra from CAD models, or arbitrary triangulations of the surface are acceptable. Even unstructured sets of points from raw range data can be used provided that the density of points is high enough that the distance to the surface can be estimated reasonably accurately.

The general approach is to first define an initial mesh near the object with the topology of Section 3.2 and to slowly deform it by moving its nodes until the mesh satisfies two conditions: It must be close to the input object, and it must satisfy the local regularity condition. The first condition ensures that the resulting mesh is a good approximation of the object, while the second condition ensures that a valid SAI can be derived from the mesh. Section 4.1 describes the basic algorithm for deforming the mesh; Section 4.2 describes the construction of the mesh used to initiate the deformation algorithm. Section 4.3 describes the algorithm for converting the final mesh to a spherical representation and gives examples of building meshes and SAIs from range data and from CAD models. Finally, Section 4.4 shows how the same algorithm can be used to perform the inverse operation, that is, reconstructing an object from a given SAI.

4.1 Mesh Deformation

The problem is now to deform the mesh such that all the nodes satisfy two fundamental properties:

- Mesh nodes must be as close as possible to the original surface.
- Mesh nodes must satisfy the normal constraints: a node is on the line parallel to the normal vector of the plane formed by its three neighbors and passing by the center of the neighbors.

These two conditions ensure that the mesh is a good approximation of the surface while guaranteeing that it is an intrinsic representation. The formalism of deformable surfaces [8] is applied to deform the mesh until it satisfies these criteria. Specifically, each node is subject to two types of forces. The first type of forces brings a node closer to the input surface, while the second type forces the node to satisfy the normal constraint. Let F_o be the force of the first type applied at a given node N, and F_g be the force of the second type at the same node. Node P is iteratively moved according to those forces. If P_{t+1}, P_t, and P_{t-1} are the positions of node P at three consecutive iterations, the update rule is defined as:

$$P_{t+1} = P_t + F_o + F_g + D\,(P_t - P_{t-1}) \tag{1}$$

This expression is simply the discrete version of the fundamental equation describing a mechanical system subject to two forces and to a damping coefficient D. D must be between 0 and 1 to ensure convergence. As long as it is within these bounds, D affects only the rate of convergence. A typical value is $D = 0.5$. Theoretically, the combination of forces brings the mesh to a state such that $F_o \approx 0$ and $F_g \approx 0$. In practice, the iterative update of the mesh is halted when the relative displacements of the nodes from one iteration to the next are small.

F_o is defined by calculating the point P_c from the original surface that is closest to the node, that is:

$$F_o = kPP_c \tag{2}$$

Where k is the spring constant of the force which must be between 0 and 1. The effect of the force is negligible if the node is already very close to the surface. Conversely, the force pulls nodes that are far from the surface, the strength of the force increasing with distance. When the points are far away, it is desirable to limit the strength of the force to avoid unstable situations in which a node would move toward the surface too quickly and overshoot the optimal position by a large distance. In practice, k varies between 0.01 at the beginning of the iterations to 0.4 at the end of the iterations, that is, when the nodes of the mesh have reached a stable position.

The curvature force F_g is calculated by computing the point P_g that is on the line normal to the triangle formed by the three neighbors of P and containing G (Figure 9), and such that the mesh curvature at P and P_g are the same: $g(P_g) = g(P)$. Those two conditions ensure that P_g satisfies the local regularity condition while keeping the original mesh curvature. F_g is defined as a spring force proportional to the distance between P and P_g:

$$F_g = aPP_g \tag{3}$$

To avoid unstable behavior of the system, the spring constant a should be between 0 and 1/2. In practice, $a = 1/2$.

4.2 Initialization

For the iterative mesh update to converge, the mesh must be initialized to some shape that is close to the initial shape. We use two different approaches depending on whether the input data is a set of data measured on the object by a sensor, or a synthetic CAD model.

In the case of sensor data, we use the techniques presented in [8] using deformable surfaces to build a triangulated mesh that approximates the object. The deformable surface algorithm fits a discrete surface to the input data, interpolating over the unknown regions, retaining salient features of the surface, if any, and smoothing the input data. When the representation is to be computed from sensor data, this technique is particularly effective because the deformable surface algorithm tends to filter out noise in the data. This algorithm is also effective in performing segmentation by separating an object from its surroundings in a complex scene. Once a triangulation is obtained, the mesh is initialized by tessellating the ellipsoid of inertia of the input shape. The ellipsoid of inertia is easily computed from the input surface, while the tessellation is computed by deforming a sphere tessellated using the topology defined in Section 3.2. Although the ellipsoid is only a crude approximation of the object, it is close enough for the mesh deformation process to converge. The distance between a node and the triangulated surface is computed by finding the closest vertex of the triangulation and by computing the minimum distance from the mesh node to the set of triangles around the vertex.

In the case of a synthetic CAD model as input, for example a polyhedron, the ellipsoid of inertia is computed directly from the synthetic model. A regular mesh is mapped on the ellipsoid in the same manner as in the previous case. In this case, the intermediate representation using the deformable surface algorithm is not necessary since there is no noise to filter out. In fact, using an intermediate model would degrade the model by smoothing out corners and eliminating high curvature features.

Once the initial ellipsoid is generated, the mesh generation is completely independent of the actual format of the input data. In particular, the mesh generation algorithm can handle a variety of representations as input, including triangulations, curved or polyhedral CAD models, and sets of data points from range images. The only operation that is data-dependent is the computation of the object point closest to a given node.

4.3 From Mesh to SAI

Once a regular mesh is created from the input data, a reference mesh with the same number of nodes is created on the unit sphere. The value of the angle at each node of the mesh is stored in the corresponding node of the sphere.

The sequence of operations from input surface description to SAI is summarized in Figure 16. The SAI building algorithm is illustrated in Figure 17 with range data as input and in Figure 18 with a polyhedral model as input. Figure 17 (a) shows three views of a green pepper from which three 240x256 range images were taken using the OGIS range finder. The images are merged and an initial description of the object is produced using the deformable surface algorithm. Figure 17 (b) and Figure 17 (c) show the initial mesh mapped on the ellipsoid and the mesh at an intermediate stage. Figure 17 (d) shows the final regular mesh on the object. Figure 17 (e) shows the corresponding SAI. The meshes are displayed as depth-cued wireframes.The SAI is displayed by placing each node of the sphere at a distance from the origin that is proportional to the angle stored at that node. Figure 18 (a) to Figure 18 (d) show the same sequence in the case of an object initially described as a polyhedron as generated by the VANTAGE CAD system. However, the initial surface is computed using the faces of the CAD model rather than a set of data points as in Figure 17. Once this intermediate representation is generated, the mesh deformation and SAI generation algorithms proceed in the same manner. The arrow between Figure 18 (c) and Figure 18 (d) shows the correspondence between object mesh and its SAI. The vertical crease in the middle of the SAI corresponds to the concave region between the two cylinders. The top and bottom regions of the SAI exhibit large values of the angle corresponding to the transition between the cylindrical and planar faces at both extremities of the object. In this example, the SAI exhibits some noise in regions that are near the edges between faces of the object. In practice, the SAI is smoothed before being used for recognition.

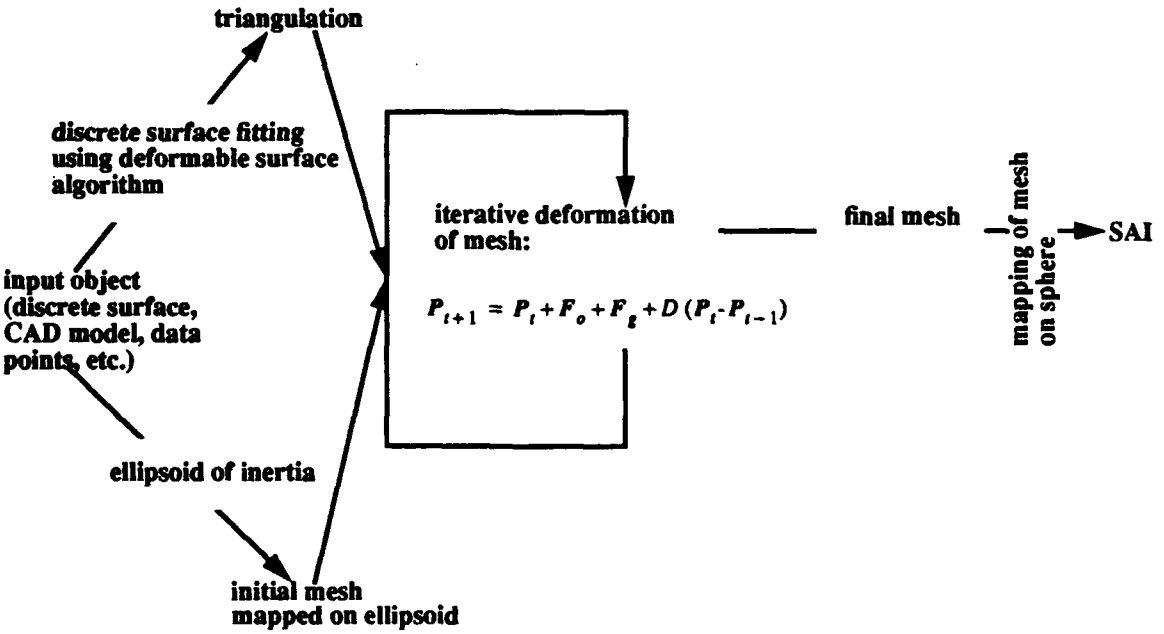
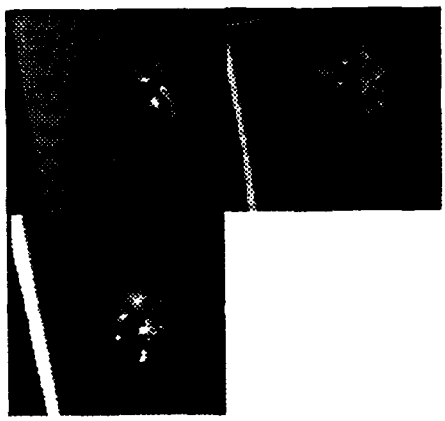
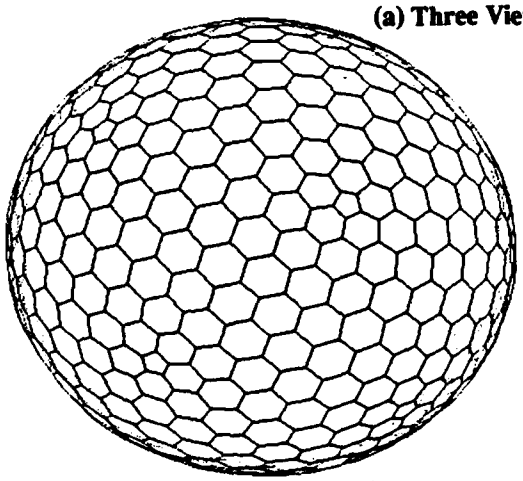


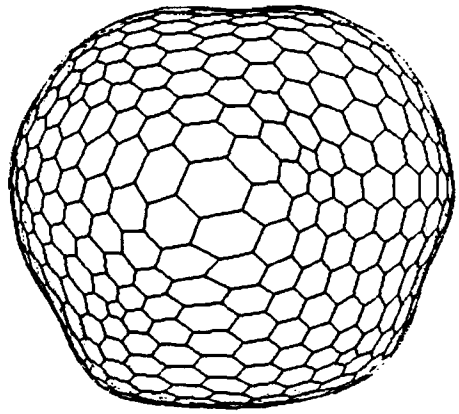
Figure 16 Summary of Building SAI from Input Object Description



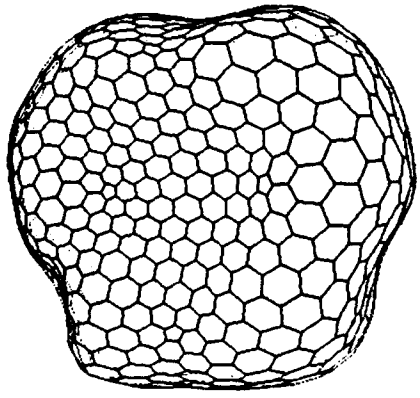
(a) Three Views of an Object



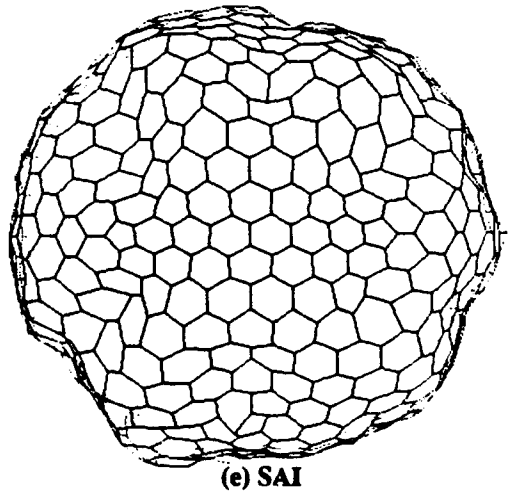
(b) Initial Ellipsoid



(c) Mesh After 10 Iterations

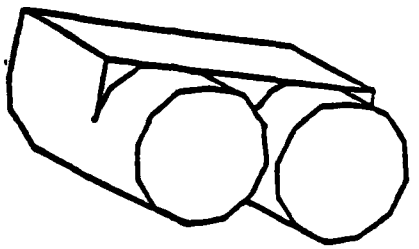


(d) Final Mesh

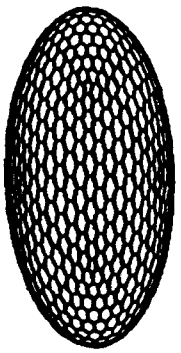


(e) SAI

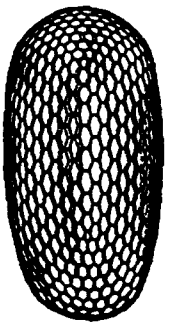
Figure 17 Building SAI from Range Data



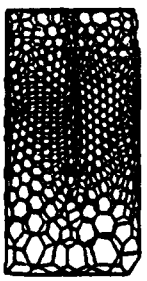
(a) Input Object Description



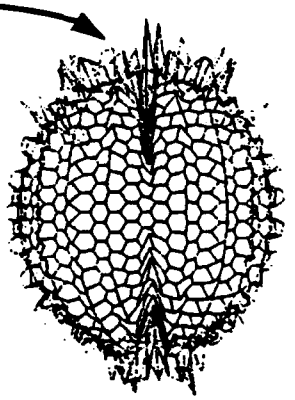
(b) Initial Ellipsoid



(c) Mesh After 10 Iterations



(d) Final Mesh



(e) SAI

Figure 18 Building the SAI from a Polyhedral Model

4.4 Reconstructing Shape from SAI

A fundamental property of the SAI representation is that the original shape can be reconstructed from its SAI up to a rigid transformation and a scale factor. In fact, the same algorithm that is used for building the mesh from an input surface can be used to perform the inverse operation. Starting with the standard regular mesh on the sphere, iteratively apply the deformation given by (1) until the mesh settles in a stable configuration. There are two differences between the inverse and the direct algorithm. First, the point P_g of (3) is defined as the point on the line normal to the triangle formed by the neighbors that has the same angle as the angle stored in the SAI, whereas in the direct algorithm it is defined as the point that has the same angle as the mesh. Second, $F_o = 0$ since there is no reference surface to attract the node.

5 Matching Objects

We now address the matching problem: Given two SAIs, determine whether they correspond to the same object. If so, find the rigid transformation between the two instances of the object. As discussed in Section 3, the representations of a single object with respect to two different reference frames are related by a rotation of the underlying sphere. Therefore, the most straightforward approach is to compute a distance measure between the SAIs for every possible rotation. Once the rotation yielding minimum distance is determined, the full 3-D transformations can be determined. Because it requires the testing of the entire 2-D space of rotations, it is expensive. We discuss strategies to reduce the search space in Section 5.3. Before that, in Sections 5.1 and 5.2, we discuss the distance measure and the computation of the final rigid transformation, respectively.

5.1 Finding the Best Rotation

Let S and S' be the spherical representations of two objects. Denoting by $g(P)$, *resp.* $g'(P)$, the value of the simplex angle at a node P of S, *resp.* P of S', S and S' are representations of the same object if there exists a rotation R such that:

$$g'(P) = g(RP) \tag{4}$$

For every point P of S'. Since the SAI is discrete, $g(RP)$ is not defined because in general RP will fall between nodes of S'. We define a discrete approximation of $g(RP)$, $G(RP)$, as follows: Let P_1, P_2, P_3, and P_4 be the four nodes of S' nearest to RP. $G(RP)$ is the weighted sum of the values $g(P_i)$. Formally:

$$G(RP) = \sum_{1}^{4} W(\| RP - P_i \|) g(P_i) \tag{5}$$

Where $W(d)$ is a weighting function that is *1* if $d = 0$ and *0* if d is greater than the average distance between nodes. This definition of G amounts to computing an interpolated value of g using the four nearest nodes.

The problem now is to find this rotation using the discrete representation of S and S'. This is done by defining a distance $D(S, S', R)$ between SAIs as the sum of squared differences between the simplex angles at the nodes of one of the sphere and at the nodes of the rotated sphere. Formally, the distance is defined as:

$$D(S, S', R) = \sum_{S} (g'(P) - G(RP))^2 \qquad (6)$$

The minimum of D corresponds to the best rotation that brings S and S' in correspondence. The simplest strategy is to sample the space of all possible rotations, represented by their angles (φ,θ,ψ), and to evaluate D for each sample value $(\varphi_i,\theta_i,\psi_i)$. The convention used for the rotation angles is shown in Figure 19: θ is the rotation about the X axis, φ is the rotation about the Z axis, and ψ is the rotation about the new Z axis. This approach is obviously expensive; Section 5.3 presents better strategies.

It is important to note that the rotation is *not* the rotation between the original objects; it is the rotation of the *representations*. An additional step is needed to compute the actual transformation between objects as described below.

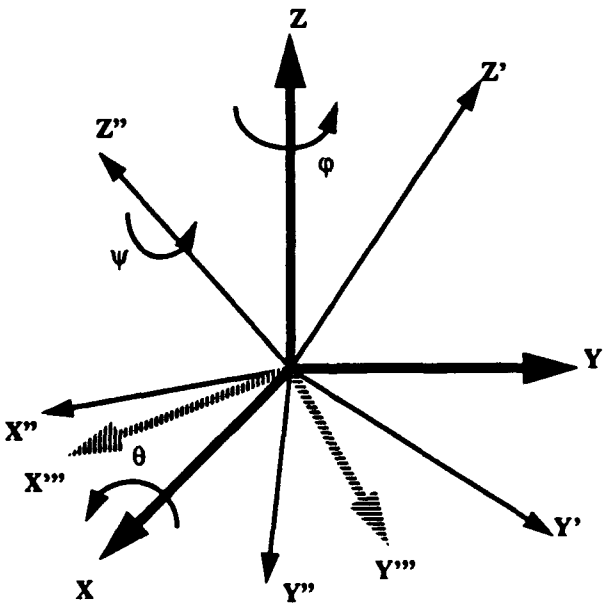


Figure 19 Rotation Angles

5.2 Computing the Full Transformation

The last step in matching objects is to derive the transformation between the actual objects, given the rotation between their SAIs. The rotational part of the transformation is denoted by R_O, the translational part by T_O. Given a SAI rotation R, for each node P' of S' we compute the node P of S that is nearest to RP'. Let M, *resp.* M', be the point on the

object corresponding to the node P of S, *resp.* P'. A first estimate of the transformation is computed by minimizing the sum of the distances between the points M of the first object and the corresponding points $R_o M'+T_o$ of the second object. Formally, the expression to minimize is:

$$E_o(R_o, T_o) = \sum \| R_o M' + T_o - M \|^2 \tag{7}$$

The sum in this expression is taken over the set of all the nodes of the mesh. The transformation derived by minimizing (7) is only an approximation because it assumes that the nodes from the two meshes correspond exactly. Due to the discretization, the assumption is not true in general. Furthermore, the fact that P' is the node nearest to P in the SAI does not necessarily mean that M' is the node nearest to M on the object. Therefore, the initial estimate needs to be refined to take into account this discretization effect. A more accurate criterion would be to require that each transformed node $R_o M'+T_o$ be as close as possible to the plane defined by the point M and the estimated normal vector N at M. This definition is more liberal in that it does not require $R_o M'+T_o$ and M to correspond exactly. Instead, the definition requires $R_o M'+T_o$ to be near the tangent plane at M. Denoting by (R_1, T_1) the new estimate of the transformation, this definition amounts to finding the minimum of the function:

$$E_1(R_1, T_1) = \sum (N \cdot (R_1 M' + T_1 - M))^2 \tag{8}$$

An iterative technique is used to find the minimum of E_1 using (R_o, T_o) as a starting point.

5.3 Reducing the Search Space

As mentioned in Section 5.1, the brute force approach to finding the best mesh rotation is too expensive to be practical. However, several strategies can be used to make it more efficient. The first strategy is to use a coarse-to-fine approach to locating the minimum of the function D of (6). In this approach, the space of possible rotations, defined by three angles of rotation about the three axis, (φ,θ,ψ), is searched using large angular steps $(\Delta\varphi, \Delta\theta, \Delta\psi)$. After this initial coarse search, a small number of locations are identified around which the minimum may occur. The space of rotations is again searched around each potential minimum found at the coarse level using smaller angular steps $(\delta\varphi, \delta\theta, \delta\psi)$. Typical values are $\Delta\varphi = \Delta\theta = \Delta\psi = 10^o$, corresponding to a 36x18x36 search space at the coarse level. The rotation space is then searched in an 18^o wide interval around each potential minimum found at the coarse level. More levels of search may be more efficient although we have not yet tried to determine the best combination of coarse-to-fine levels.

The second approach is to use a-priori knowledge about the relative poses of the objects to reduce the search space. For example, the rotation defined by the axis of inertia of the SAIs can be used as a starting point for the search. In general, a coarse-to-fine approach should be used in a relatively large region centered at the rotation calculated from the axis of inertia. The use of a large region is necessary because parts of the objects may be occluded, leading to variations in the axis of inertia, and because the rotation computed

from the axis of inertia is a crude approximation of the true rotation. In practice, using the axis of inertia is very effective in pruning the search space as long as the visible part of the object is large enough.

Finally, features on the spherical representation itself can be used to further narrow the search. Specifically, local maxima of g on both spherical representations can be matched to compute an initial rotation to narrow the search space. This initial rotation would be more accurate than the one from the axis of inertia. However, it would be computable only if the spherical representations exhibit well-defined local maxima. Although we did not implement this strategy, it seems to be more appropriate to the case in which a smaller portion of the object is actually visible.

5.4 Example

Figure 20 shows three views of the same object as in Figure 17 placed in a different orientation. A model is built from the three corresponding range images using the approach described in 4.3. Figure 21 illustrates the difference of pose between the two models computed from the two sets of images. Figure 21 (a) (resp. Figure 21 (b)) shows the superimposition of the cross sections of the two models in the plane YZ (resp. XY). Figure 22 shows the value of the SAI distance measure. The distance measure is displayed as a function of φ and θ only since the distance is a function of three angle that cannot be displayed easily. The displayed value at (φ,θ) is the minimum value found for all the possible values of ψ. The resolution of the graph is 10^{o} in both φ and θ, and the angles are defined using the convention of Figure 19. This display shows that there is a single sharp minimum corresponding to the rotation that brings the SAI in correspondence. Figure 23 (a) and (b) show the superimposition of the cross-sections of both models after the second was transformed using the transformation computed from the SAI correspondence using the algorithms of Section 5.2. Figure 24 shows one of the models backprojected in the image of the other using the computed transformation. Figure 24 (a) is the original image; Figure 24 (b) is the backprojected model. These displays show that the transformation is correctly computed in that the average distance between the two models after transformation is on the order of the accuracy of the range sensor. This example demonstrates the use of SAI matching in the case of complete models. In the next section, we address the problem of dealing with partial views.

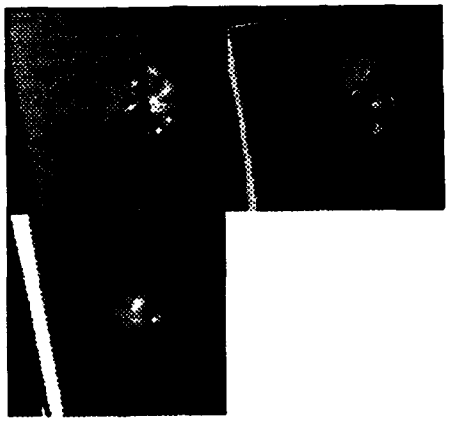
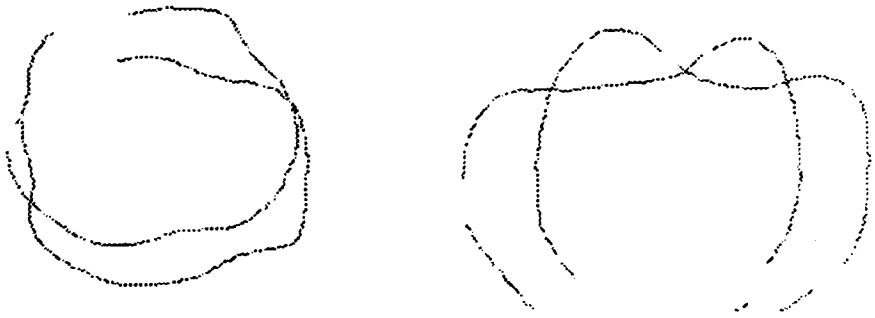


Figure 20 Three Views of the Object of Figure 17 in a Different Orientation



(a) Cross-Section in X **(b) Cross-Section in Z**

Figure 21 Relative Positions of the Models Before Matching

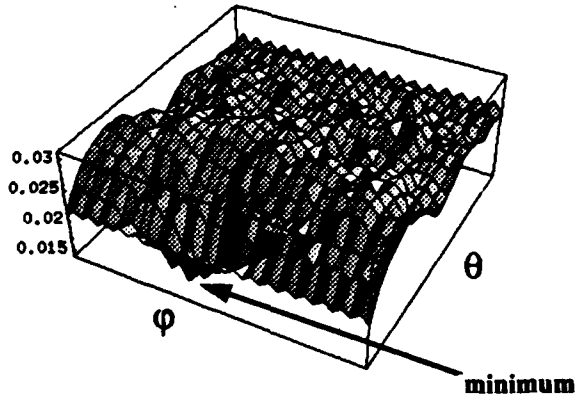
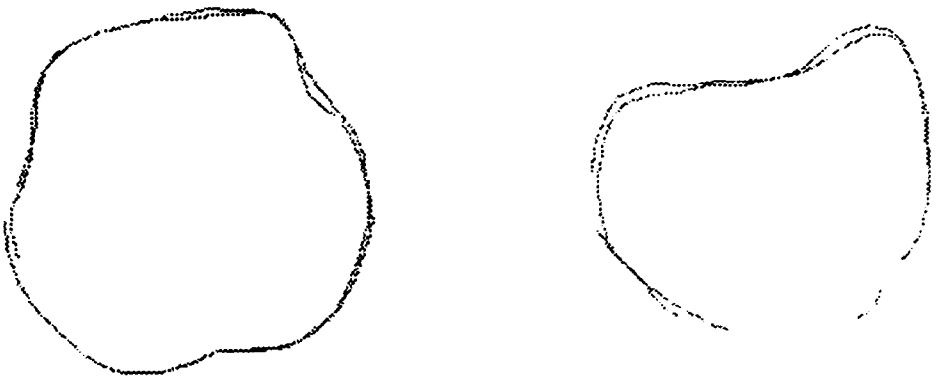
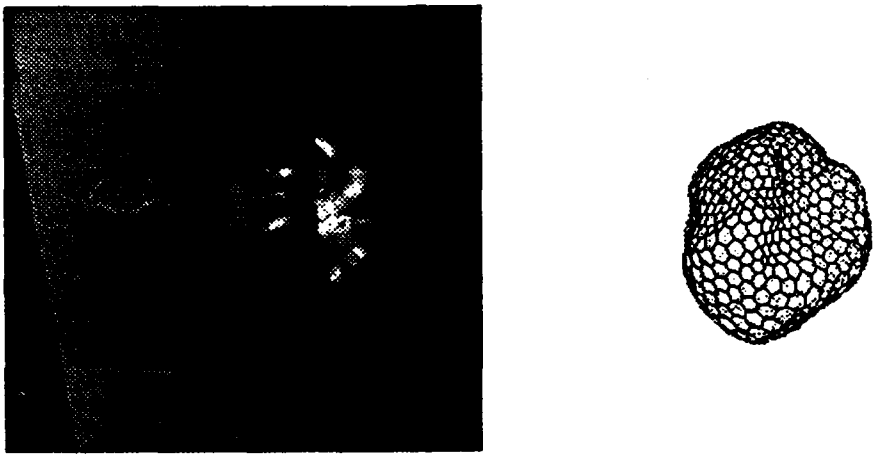


Figure 22 Graph of Distance Between SAIs as a Function of φ and θ



(a) Cross-Section in X (b) Cross-Section in Z

Figure 23 Relative Positions of the Models after Matching



(a) Image of First Model (b) Second Model Displayed Using Computed Pose

Figure 24 Display of the Model in the Computed Pose

6 Partial Views and Occlusion

Up to now we have assumed that we have a complete model of the object, as in Figure 18, or that we have data covering the entire surface of the object, as in Figure 17. This assumption is appropriate for building reference models of objects. During the recognition phase, however, only a portion of the object is visible in the scene. The matching algorithm of Section 5 must be modified to allow for partial representations. The algorithm used for extracting the initial surface model is able to distinguish between regions of the mesh that are close to input surfaces or to data points, and parts that are interpolated between input data. The first type of region is the visible part of the mesh, and the second type is the occluded part of the mesh. Therefore, even though the representation is always a mesh mapped on a closed surface, it is always possible to determine which nodes of the mesh represent valid data.

The situation is illustrated in Figure 25 in the case of a two dimensional contour. In Figure 25 (a) a contour is approximated by a mesh of eight points. The mesh is assumed to be regular, that is all the points of the mesh are equidistant. Let $L = 8l$ be the total length of the mesh. Figure 25 (b) shows the same contour with one portion hidden. The occluded portion is shown as a shaded curve. The visible section is approximated by a regular mesh of eight nodes of length $L_1 = 8l_1$. Since the occluded part is interpolated as a straight line, the length of this mesh is smaller than the total length of the mesh on the original object: $L > L_1$. Conversely, the length of the part of the representation corresponding to the visible part, L_2 shown in Figure 25 (d), is greater than the length of the same section of the curve on the original representation, L^* shown in Figure 25 (c). In order to compute the distance measure of Section 5, the SAI of the observed curve must be scaled so that it occupies the same length on the unique circle as in the reference representation of the object. If L^* were known, the scale factor would be:

$$k = \frac{L^*}{L_2} \tag{9}$$

In reality, L^* is not known because we do not yet know which part of the reference curve corresponds to the visible part of the observed curve. To eliminate L^*, we use the relation:

$$\frac{L_1}{L} = \frac{L^*}{2\pi} \tag{10}$$

This relation simply expresses the fact that the ratios of visible and total length in object and representation spaces are the same, which is always true when the mesh is regular. Since the left-hand side involves only known quantities, total length of model and observed visible length, L^* can be eliminated by combining (9) and (10):

$$k = \frac{2\pi}{L_2}\frac{L_1}{L} \tag{11}$$

The situation is the same in three dimension except that lengths are replaced by areas A, A_1, A_2, A^*, with obvious notations. Relation (11) becomes:

$$k = \frac{4\pi}{A_2}\frac{A_1}{A} \tag{12}$$

The direct extension from two to three dimension is only an approximation because the equivalent of relation (10), $A_1/A = A/4\pi$, holds only if the area per node is constant over the entire mesh. In practice, however, the area per node is nearly constant for a mesh that satisfies the local regularity condition.

Once k is computed, the appropriate scaling needs to be applied to the SAI. The scaling algorithm is illustrated in Figure 26: if C is the center of the visible region on the representation sphere, a node P such that θ is the angle (OP, OC) is moved to the point P' on the great circle that contains P and C such that:

$$1 - \cos\theta' = k(1 - \cos\theta) \tag{13}$$

where θ' is the angle (*OP'*, *OC*) and *k* is the scale factor. This mapping is chosen because it guarantees that the area of the visible region is scaled exactly by *k* if the region is circular. Even if the region is not circular, the mapping is a reasonable approximation.

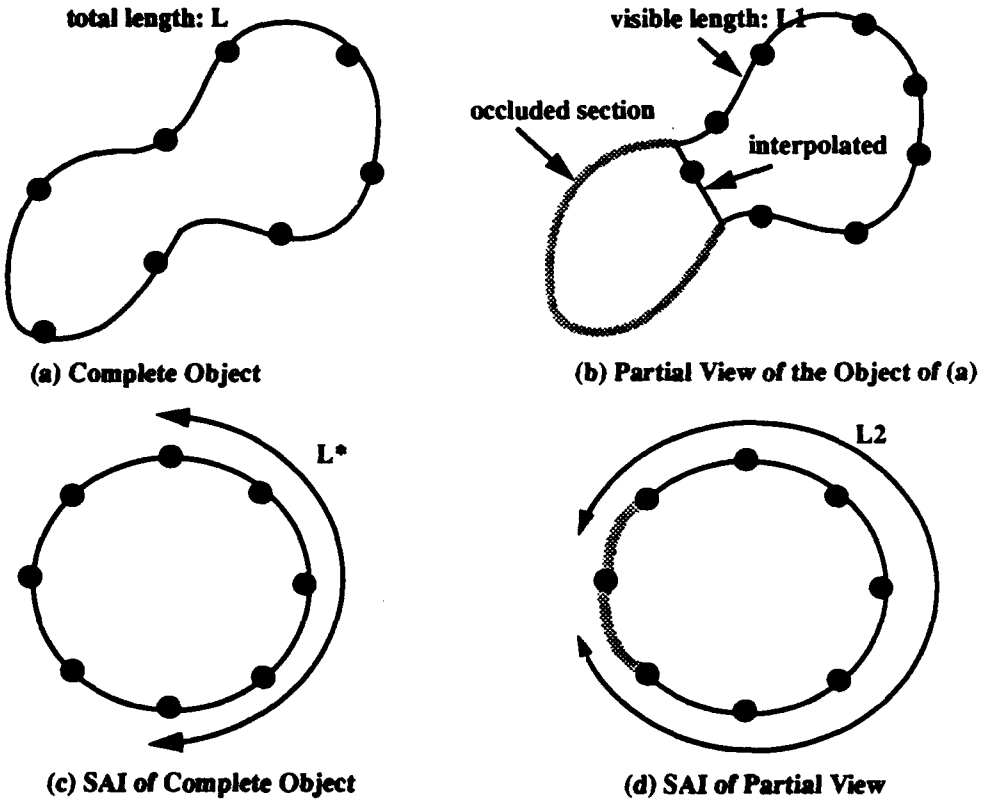


Figure 25 Matching Partial Representation in Two Dimensions

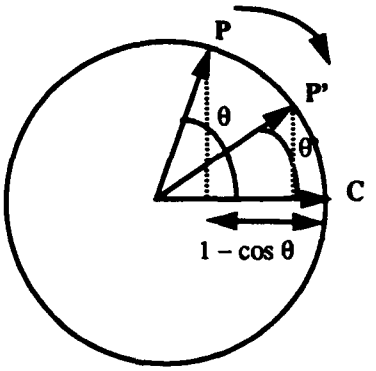


Figure 26 SAI Scaling Algorithm

The key in this algorithm is the connectivity conservation property of the SAI mentioned previously. Specifically, if a connected patch of the surface is visible, then its corresponding image on the SAI is also a connected patch on the sphere. This property allows us to

bring the two connected patches into correspondence using a simple scaling. Establishing the correspondence is not possible in the case of the EGI representation, in which the spherical representations of an object and of a patch on the object may be completely different. If the object is represented by an implicit equation, e.g., algebraic function or superquadrics, then the coefficients of the equation computed from the entire object surface may be completely different from the ones computed from only a patch on the object surface.

We now show two examples of recognition in the presence of occlusion. In the first example, a range image of an isolated object is taken. Then a complete model of the object is matched with the SAI representation from range data. Figure 27 shows the intensity image of the object. Only about 30% of the object is visible in the image. The remaining 70% of the representation built from the image is interpolated and is ignored in the estimation of the SAI distance. Figure 28 (a) shows the set of three registered views used to build the reference model, and Figure 28 (b) shows the SAI of the reference model used for matching. Figure 29 shows the superimposition of scene points and reference model after transformation. Figure 30 displays the graph of the distance between SAIs as function of rotation angles. Figure 30 (a) shows two views of the distance as a function of φ and θ. Figure 30 (b) shows the same function displayed in φ-ψ space. These displays demonstrate that there is a well-defined minimum at the optimal rotation of the SAIs. Figure 31 shows the model backprojected in the observed image using the computed transformation. In this example, the reference model was computed by taking three registered range images of the object as in the example of Figure 17.

In the second example, the reference model is the CAD model of Figure 18. The observed scene is shown in Figure 32. The result of the matching is shown in Figure 33 and Figure 34. Only part of the object is visible in the image because about of self occlusion and because of occlusion from other objects in the scene.

In both examples, the deformable surface algorithm is used to separate the object from the rest of the scene and to build an initial surface model. If there is no data point in its vicinity, the visible portion of the object mesh and the corresponding SAI are identified by marking a node of the mesh as interpolated. Using the algorithms presented in this section, the SAI of the observed object was scaled based on the size of the visible area. As an example, Figure 35 (a) shows the SAI computed from the image of Figure 27, Figure 35 (b) shows the SAI after the scaling in applied to compensate for occlusions. The density of points increases in the region that corresponds to the visible part of the object (indicated by the solid arrow). Conversely the density of points decreases in the region corresponding to the occluded part of the object (indicated by the shaded arrow). These examples show that the SAI matching algorithm can deal with occlusions and partial views, even when only a relatively small percentage of the surface is visible.

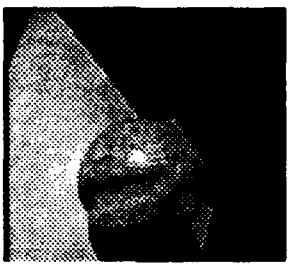
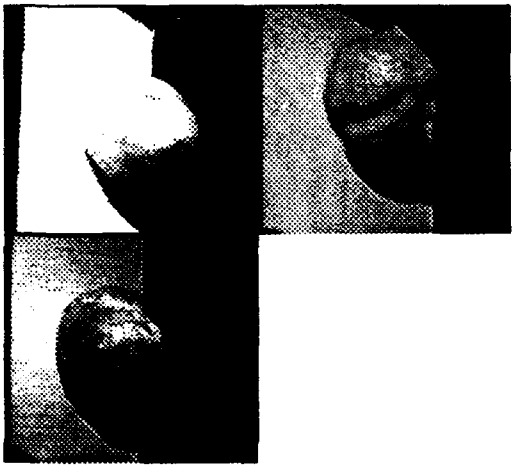
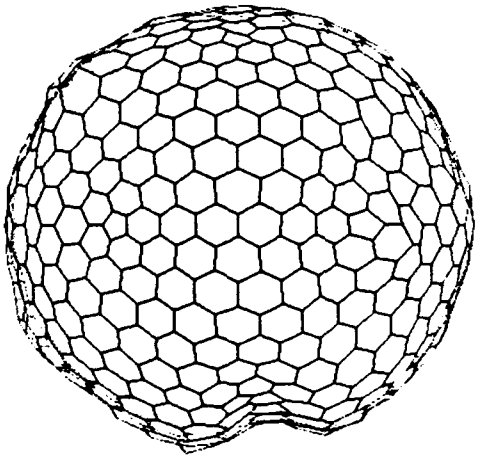


Figure 27 Input Image



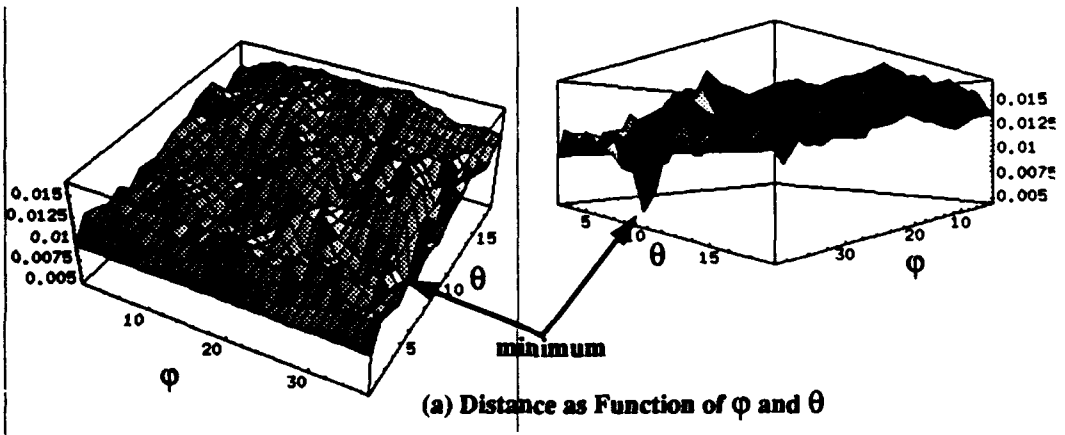
(a) Images Used to Build the Model



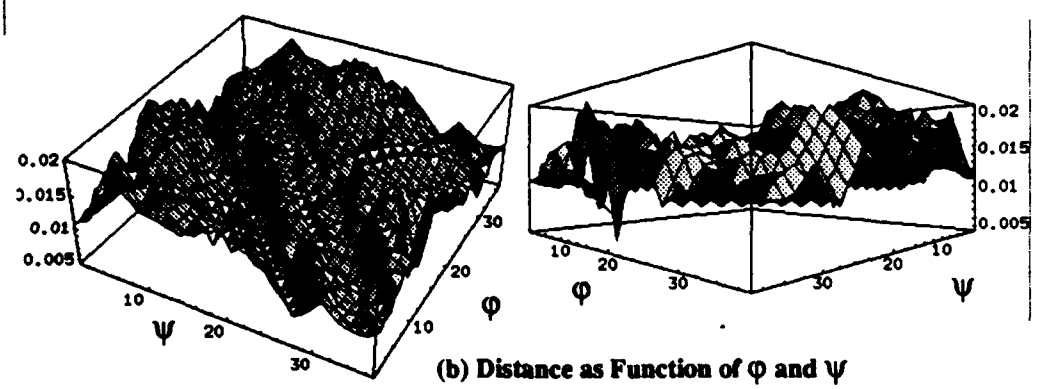
(b) SAI of Reference Model

Figure 28 Reference Model

Figure 29 Cross Sections of Registered Model and Scene Using the Image of Figure 27 and the Model from Figure 28

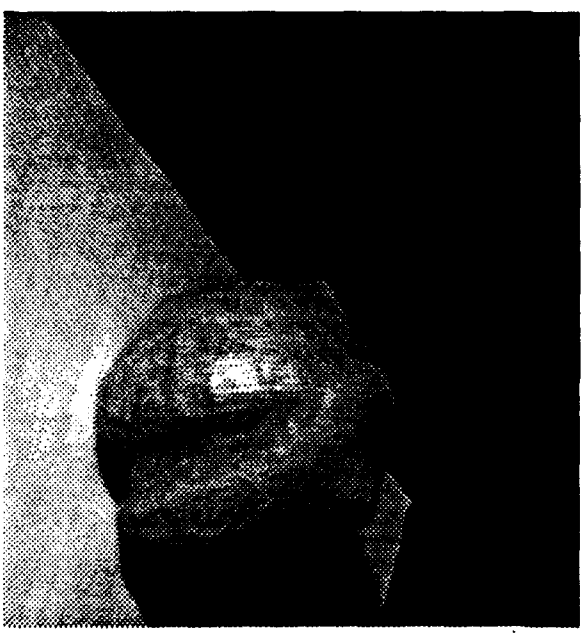


(a) Distance as Function of φ and θ

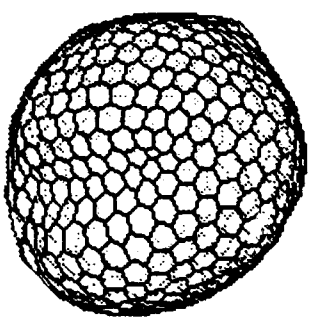


(b) Distance as Function of φ and ψ

Figure 30 Sum of Squared Differences of SAIs as Function of Rotation Angles



(a) Input Image



(b) Model after Transformation

Figure 31 Display of Model Using the Pose Computed from the Matching

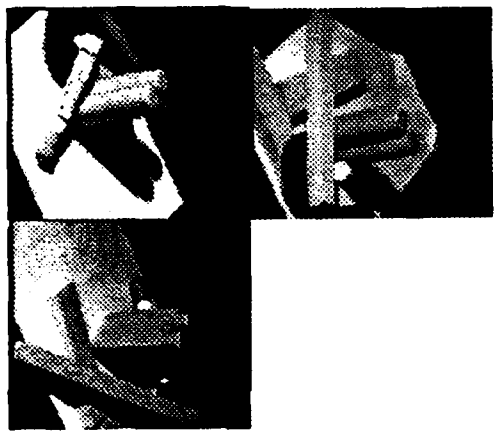
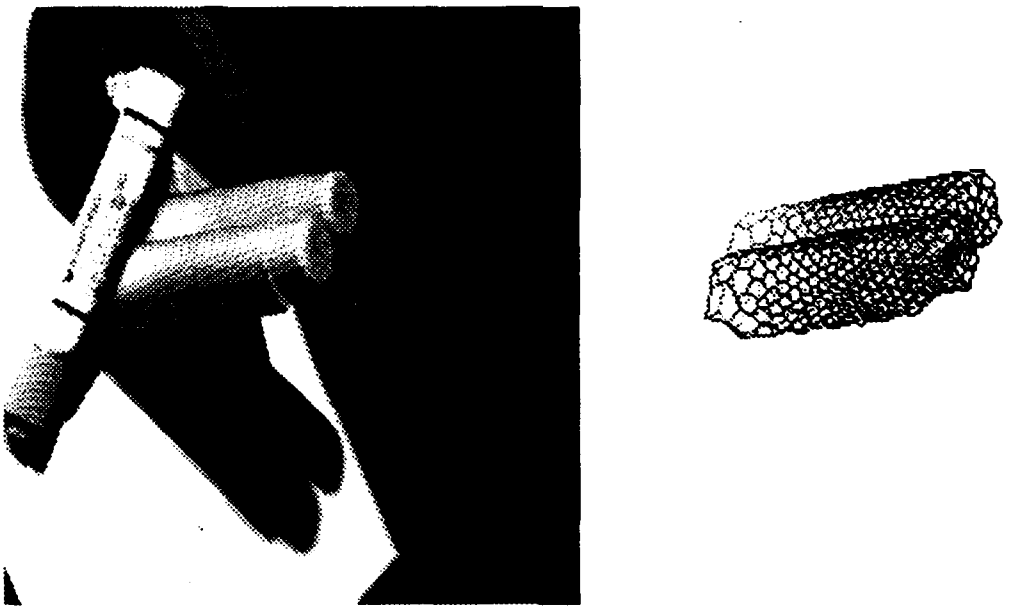


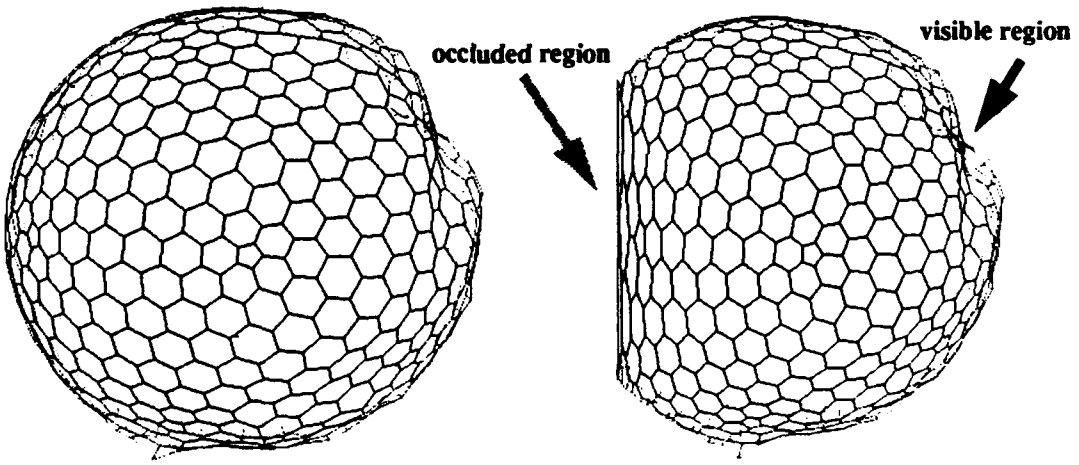
Figure 32 Input Image

Figure 33 Cross Sections of Registered Model and Scene Using the Image of Figure 32 and the Model of Figure 18



(a) Input Image (b) Model after Transformation

Figure 34 Display of Model Using the Pose Computed from the Matching



(a) SAI from Scene (b) SAI after Scaling

Figure 35 Effect of Occlusion-Compensating Scaling on SAI of Observed Object

7 Conclusion

In this paper, we introduced a new approach for building and recognizing models of curved objects. The basic representation is a mesh of nodes on the surface that satisfies certain regularity constraints. We introduced the notion of simplex angle as a curvature indicator stored at each node of the mesh. We showed how a mesh can be mapped into a

spherical representation in canonical manner, and how objects can be recognized by computing the distance between spherical representations.

The SAI representation has many desirable properties that make it very effective as a tool for 3-D object recognition:

- The SAI is invariant with respect to translation, rotation, and scaling of the object. This is not true of most other commonly used representations. This invariance allows the recognition algorithm to compare shapes through the computation of distances between SAIs without requiring explicit matching between object features or explicit computation of object pose.
- The SAI preserves connectivity between parts of the object in that nodes that are neighbors on the object mesh are also neighbors on the SAI. Thus the SAI does not exhibit the same ambiguity problem for non-convex objects as the EGI and CEGI representations.
- The SAI representation can handle partial views and occluded objects. The basic approach is to measure the area of the visible portion of an object observed in a scene, and deform the SAI mesh model so that the percentage of the sphere corresponding to the visible area is the same in both model and scene SAIs. This approach to recognition of occluded objects is practical thanks to the property of connectivity conservation described above. Specifically, a connected visible region of an object corresponds to a connected region on the corresponding SAI.

Results show that the SAI representation is successfully used to determine the pose of an object in a range image including occlusion and multiple objects. This approach is particularly well suited for applications dealing with natural objects. Typically, conventional object modeling and recognition techniques would not work due to the variety and complexity of shapes that may have to be handled. The approach is general enough that it can also convert manually built models to the SAI representation.

Many issues remain to be addressed. First, we need to improve the search for the minimum distance between SAIs during the recognition phase. This improvement can be achieved by improving the coarse-to-fine approach to extrema localization, and by using cues computed from the original data to restrict the area in which the extrema are searched. Another interesting direction of research is the parallel implementation of the algorithm which should be possible since computations at each node of the sphere and for each possible rotation are independent of each other. A third issue is the extension of those techniques to objects that are not topologically equivalent to the sphere, such as torus-shaped objects or open-surfaced objects. This extension requires the definition of mesh topology and global regularity for those classes of shapes. To apply the technology to more complex scenes, additional range image processing needs to be developed to isolate objects in the scene from one another. Currently, the algorithm introduced in [8] uses a fixed window around the center of each potential object in the scene. A better initial segmentation is needed to deal with complex scenes. Finally, an important issue is to determine the appropriate mesh resolution, as given by the number of nodes, that is needed for a particular application. Currently, the empirical selection of the size of the mesh can lead to two possible problems. If the size of the mesh is too small, important details of the object may be undersampled. On the other hand, if the mesh is too large, computation time

for both model building and recognition becomes prohibitive. The density of nodes should be computed based on the average and maximum curvatures of the surface.

8 References

[1] Aleksandrov, A.D., Zalgaller, V.A., "*Intrinsic Geometry of Surfaces*", Translation of Mathematical Monographs Series, AMS Publisher, 1967

[2] Besl, P.J., Kay, N.D., "*A Method for Registration of 3-D Shapes*", IEEE Trans. Pattern Analysis and Machine Intelligence, PAMI, Vol. 14, No. 2, p. 239, 1992

[3] Besl, P., Jain, R., "*Segmentation Through Symbolic Surface Descriptions*", Proc. IEEE Conf. on Computer Vision and Pattern Recognition , IEEE Computer Society, pp 77-85, Miami, 1986

[4] Bobick, A.F., Bolles, R.C., "*The Representation Space Paradigm of Concurrent Evolving Object Descriptions*", IEEE Trans. Pattern Analysis and Machine Intelligence, PAMI, Vol. 14 , No. 2, p. 146, 1992

[5] Bolle, R.M., B.C. Vemuri, "*Three Dimensional Surface Reconstruction Methods*", IEEE Trans. Pattern Analysis and Machine Intelligence , Vol. 13, pp. 1-14, 1991.

[6] Brady, M. and Ponce, J. and Yuille, A. and Asada, H., "*Describing Surfaces*", Proc. 2nd International Symposium on Robotics, MIT Press, Cambridge, MA, 1985

[7] Brou, P. , "*Using the Gaussian Image to Find the Orientation of Object*" , The International Journal of Robotics Research , 3 , 4 , 89-125 , 1983

[8] Delingette, H., Hebert, M. and Ikeuchi, K. , "*Shape Representation and Image Segmentation Using Deformable Surfaces*" , Image and Vision Computing, Vol. 10, No. 3, p. 132April 1992

[9] Faugeras, O.D. and Hebert, M. , "*The Representation, Recognition, and Locating of 3-D Objects*" , The International of Robotics Research , 5 , 3 , 27-52 , 1986

[10] Ferrie, F.P., Lagarde, J. and Whaite, P. , "*Darboux Frames, Snakes, and Super-Quadrics: Geometry from the Bottom-Up*" , Proc. of the IEEE Workshop on Interpretation of 3D Scene , IEEE Computer Society , 1989 , 170-176

[11] Forsyth, D.A., Mundy, J.L., Zisserman, A., Coelho, C., Heller, A., Rothwell, C., "*Invariant Descriptors for 3-D Object Recognition and Pose*", IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 13, pp. 971-992, October 1992

[12] Grimson, W. E. L. and Lozano-Perez, T. , "*Localizing Overlapping Parts by Searching the Interpretation Tree*" , IEEE Trans. Pattern Analysis and Machine Intelligence , PAMI-9 , 4 , 1987 , July , 469-482

[13] Ikeuchi, K. , "*Recognition of 3-D Objects Using the Extended Gaussian Image*" , International Joint Conf. on Artificial Intelligence , 595-600 , 1981

[14] Ikeuchi, K. and Hong, K.S., "*Determining Linear Shape Change: Toward Automatic Generation of Object Recognition Program,*" Computer Vision, Graphics, and Image Processing: Image Understanding, Vol.53, No.2, pp.154-170, March 1991

[15] Kang, S.B. and Ikeuchi, K.,"*Determining 3-D Object Pose using the Complex Extended Gaussian Image,*" Proc. of IEEE Conf. on Computer Vision and Pattern Recognition: CVPR-91, Lahaina, Maui, Hawaii, pp.580-585, June 1991.

[16] Kass, M., Witkin, A., Terzopoulos, D., "*Snakes: Active Contour Models*", International Journal of Computer Vision, Vol. 2, No. 1, pp. 321-331, 1988

[17] Lamdan, Y., Schwartz, J.T., Wolfson, H.J., "*Affine Invariant Model-Based Object Recognition*", IEEE Trans. Robotics and Automation, Vol. 6, No. 5, pp. 578-589, October 1990

[18] Little, J.J. , "*Determining Object Attitude from Extended Gaussian Images*" , Proc. of 9th Intern. Joint Conf. on Artificial Intelligence , 960-963 , 1985

[19] Loeb, A.L., "*Space Structures*", Addison-Wesly, 1976

[20] Lowe, D.G. , "*Three-Dimensional Object Recognition from Single Two-Dimensional Images*" , Artificial Intelligence , 31 , 1987 , 355-395

[21] Oshima, M. and Shirai, Y. , "*Object Recognition Using Three-Dimensional Information*" , IEEE Trans. Pattern Analysis and Machine Intelligence , PAMI-5 , 4 , 353-361 , July , 1983

[22] Pentland, A. P. , "*Perceptual Organization and the Representation of Natural Form*" , Artificial Intelligence , 28 , 2 , 293-331 , 1986

[23] Stein, F., Medioni, G., "*Structural Indexing: Efficient 3-D Object Recognition*", IEEE Trans. Pattern Analysis and Machine Intelligence , PAMI, Vol. 14 , No. 2, p. 125, 1992

[24] Taubin, G., "*Recognition and Positioning of Rigid Objects Using Algebraic and Moment Invariants*", PhD Dissertation, Brown University, 1990

[25] Taubin, G., Cukierman, F., Sullivan, S., Ponce, J., and Kriegman, D.J., "*Parametrizing and Fitting Bounded Algebraic Curves and Surfaces*", Proc. Computer Vision and Pattern Recognition, June 1992.

[26] Terzopoulos, D., Witkin, A., Kass, M. , "*Symmetry-Seeking Models and 3D Object Recognition*" , Intern. of Computer Vision , 1 , 1 , 1987 , 211-221

[27] Wenninger, M., "*Polyhedron Models*", Cambridge University Press, London, 1971

[28] Wenninger, M., "*Dual Models*", Cambridge University Press, London, 1983

Appendix A: Simplex Angle Computation

Computing the simplex angle directly from the definition of Section 3.4 is inefficient and numerically unstable. In this Appendix, we describe the method used in the implementation. We first present the algorithm and then give a proof that the algorithm is consistent with the definition of Section 3.4. Starting with the geometry of Figure 36, we define the following notations:

- P is a node of the mesh with neighbors (P_1,P_2,P_3).
- N is the unit vector normal to the plane (P_1,P_2,P_3).
- V_i is the vector PP_i.
- L_i is the magnitude of V_i.
- t_i is the unit vector parallel to V_i: $\quad t_i = \dfrac{V_i}{L_i}$
- $T_i = (t_i \cdot N)\, t_i$

The vectors T_i and t_i can be represented on a unit sphere of center O (Figure 37 (a)). Geometrically, the vector T_i is the intersection of the unit vector t_i with a sphere of diameter N, the center of which is denoted by O_1. Another way to define the vectors T_i is to observe that (T_1,T_2,T_3) is the image by a spherical inversion of pole P of the points (P_1,P_2,P_3). The inversion maps the plane (P_1,P_2,P_3) onto the sphere of diameter N.

The triplet of points (T_1, T_2, T_3) defines a plane Π. Let d be the distance between the origin O_1 of the sphere of diameter N and Π (Figure 37 (b)). We compute the simplex angle as:

$$\cos\varphi = 2d \tag{14}$$

This definition gives an angle between 0 and π. The sign of the angle depends on whether P is on the positive or negative side of the plane defined by its neighbors and the normal vector N assumed to be pointing toward the outside of the object. This definition of φ can be computed more efficiently since it requires only three dot products and three normalizations to compute the vectors T_i, and another dot product to compute the distance to Π instead of the radii of a circumscribed sphere and a circumscribed circle using the definition of Section 3.4. It is also better conditioned in that when the surface becomes nearly flat, the vectors T_i converge toward the origin of the unit sphere and therefore $d = 1/2$ and $\varphi = 0$. By contrast, the sphere radius used in the previous definition goes to infinity as P becomes close to the plane of its neighbors.

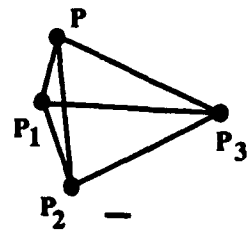
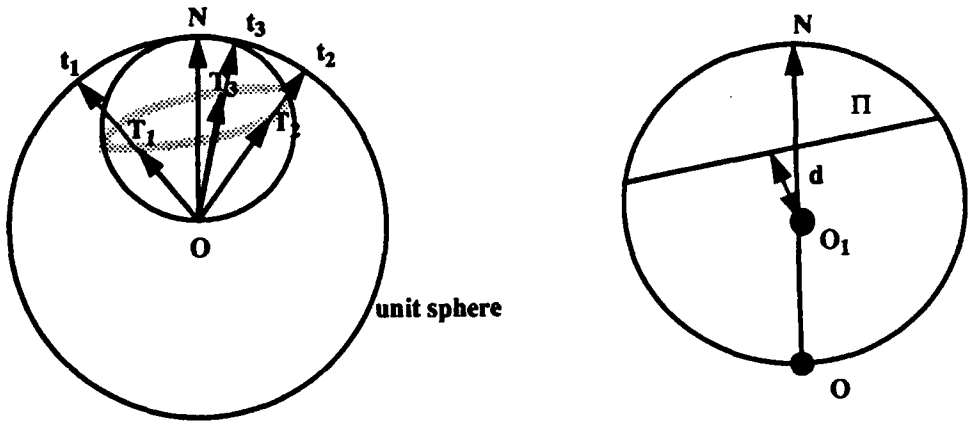


Figure 36 Geometry of Simplex Angle Computation



(a) Inversion of Unit Direction Vectors (b) Cross-Section of the Sphere of Diameter N

Figure 37 Configuration after Spherical Inversion

We now show that the angle calculated with this algorithm is indeed the same as the angle defined in Section 3.4. The proof will proceed by expressing simplex angle φ_o as a function of the vectors V_i, to use the relation between V_i and T_i to express φ_o as a function of vectors T_i, and to identify the resulting expression as the definition used in the algorithm presented above.

The original definition of φ_o can be written as:

$$\sin\varphi_o = \frac{r_o}{r} \tag{15}$$

where r_o is the radius of the circle circumscribed to the triangle (P_1, P_2, P_3), and r is the radius of the sphere circumscribed to the tetrahedron (P_1, P_2, P_3, P) (Figure 10). With the current notations, we have:

$$\begin{aligned} r &= \frac{\left\| L_1^2 (V_2 \times V_3) + L_2^2 (V_3 \times V_1) + L_3^2 (V_1 \times V_2) \right\|}{2|V_1, V_2, V_3|} \\ r_o &= \frac{1}{2} \frac{\|V_1 - V_2\| \|V_2 - V_3\| \|V_1 - V_3\|}{\|V_2 \times V_3 + V_3 \times V_1 + V_1 \times V_2\|} \end{aligned} \tag{16}$$

where $|V_1, V_2, V_3|$ is the determinant of the three vectors. Let D be distance of P to the plane of its neighbors, of normal vector N. By definition, $D = V_1 \cdot N = V_2 \cdot N = V_3 \cdot N$, a symmetrical way of writing these equalities is:

$$D = \frac{|V_1, V_2, V_3|}{\|V_2 \times V_3 + V_3 \times V_1 + V_1 \times V_2\|} \tag{17}$$

Replacing the right-hand side of (17) by D in the expression of r_o/r obtained from (16), we get:

$$\frac{r_o}{r} = D\frac{\|V_1 - V_2\|\|V_2 - V_3\|\|V_1 - V_3\|}{\|L_1^2(V_2 \times V_3) + L_2^2(V_3 \times V_1) + L_3^2(V_1 \times V_2)\|} \tag{18}$$

Replacing $V_i.N$ by D in the definition of T_i, we get the relations:

$$V_i = \frac{D}{\|T_i\|^2}T_i \qquad \|V_i - V_j\| = \frac{D}{\|T_i\|\|T_j\|}\|T_i - T_j\| \tag{19}$$

Substituting (20)the right-hand sides of (19) in (18), we get the new expression for the ratio of radii:

$$\frac{r_o}{r} = \frac{\|T_1 - T_2\|\|T_2 - T_3\|\|T_1 - T_3\|}{\|T_2 \times T_3 + T_3 \times T_1 + T_1 \times T_2\|} \tag{20}$$

The right hand side of (20) is exactly twice the radius R of the circle circumscribed to the triangle formed by the points T_1, T_2, and T_3. Therefore, from (15), $\sin\varphi_o$ is equal to $2R$. Equivalently, $\cos\varphi_o$ is the distance between the origin and the plane Π (Figure 38). Therefore the two definitions of the simplex angle, $\sin\varphi = r_o/r$ and $\cos\varphi = 2d$, are equivalent.

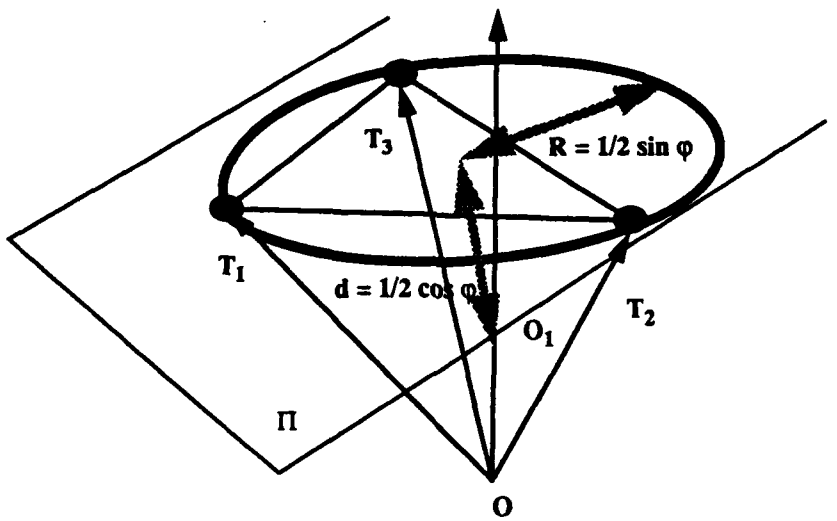


Figure 38 Relation Between φ and the Circumscribed Circle