A Spherical Representation for the Recognition of Curved Objects

H. Delingette, M. Hebert, K. Ikeuchi

The Robotics Institute
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh PA15213

Abstract[1]

In this paper, 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 which 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.

1.0 Introduction

In this paper, we propose a new representation for 3-D objects which is suitable for object recognition and pose determination. Many representations have been proposed to address this recognition problem. Traditionally, there are two ways to represent objects for recognition: *local* and *global*. Local methods attempt to represent objects as sets of primitives such as faces or edges. Most early local methods handle polyhedral objects and report effective and encouraging results. Representative systems include [12][21][15]. Few systems can handle curved surfaces. Examples of such systems include early work in which primitive surfaces enclosed by orientation discontinuity boundaries are extracted from range data [22]. 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 [24]. These local methods, however, are noise-sensitive and are still limited in reliably extracting primitives of curved objects from input images.

[1]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 DARPA or the U.S. Government.

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. Although encouraging results have been obtained in recognizing GCs in intensity images, 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 [23]. The SQs represent a limited set of shapes which can be extended by adding parameters to the generic implicit equation of SQs. 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].

EGI and CEGI map surface orientation distributions to the Gaussian sphere [13] [14] [19] [16]. Since the Gauss map is independent of 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 [14][7]. However, when part of the object is occluded, these techniques cannot reliably extract the representation.

Recently, new approaches based on the idea of fitting a bounded algebraic surface of fixed degree to a set of data points [25][26][11] have been proposed. 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. For a survey of other techniques that 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 any search.

Our approach is 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 sur-

0-8186-3870-2/93 $3.00 © 1993 IEEE

faces 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. Constructing meshes which fit input data satisfy regularity constraints is possible based on the optimization techniques originally introduced in [27] and [17]. 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 measure, the *simplex angle*, which is entirely defined from a node and its neighbors in the mesh. 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 only need to compare the corresponding spherical distributions. The overall approach is illustrated in Figure 1. This approach appears similar 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 applied to 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 representations cannot.

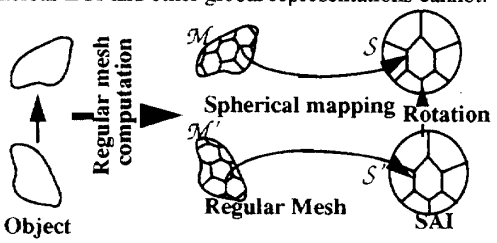


Figure 1: Object Recognition Using SAIs

The paper is organized as follows. In Section 2.0, we describe a simple representation of closed 2-D curves which we extend to three dimensional surfaces in Section 3.0. In both sections, we investigate the notions of global and local regularity, and of simplex angle. In Sections 3.0 to 5.0 we describe the fundamentals of the SAI algorithms in the case of complete object models. In Section 4.0, we show how to obtain SAIs from range data. In Section 5.0, we describe the SAI matching. We address the problem of occlusion and partial models in Section 6.0. In that section, we also present several results of recognition in complex scenes and a discussion of performance and robustness of the recognition algorithm.

2.0 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 polygons. In this section, we introduce the basic concepts that can be used to manipulate polygonal representations of contours. The concepts discussed in this section are well known and have been studied and applied extensively. Our purpose here is to introduce them in a way that facilitates their extension to three-dimensional surfaces.

In order to quantify the curvature of a contour, we use the angle φ between consecutive segments. Like the curvature, the angle φ is independent of rotation and translation. 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. One way to enforce uniform distribution 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. Another geometric definition of this condition is illustrated in Figure 2. The condition that the length of the two segments $\boldsymbol{PP}_1$ and $\boldsymbol{PP}_2$ are the same is equivalent to the condition that the projection of node $\boldsymbol{P}$ on the line joining its two neighbors $\boldsymbol{P}_1$ and $\boldsymbol{P}_2$ coincides with the center of $\boldsymbol{P}_1$ and $\boldsymbol{P}_2$. 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.

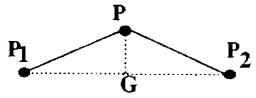


Figure 2: Local Regularity

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 $\boldsymbol{P}_1,..\boldsymbol{P}_N$, and with corresponding angles $\varphi_1,..,\varphi_N$. Let us divide the unit circle using N equally spaced vertices $\boldsymbol{C}_1,..\boldsymbol{C}_N$. Finally, let us store the angle φ_i associated with $\boldsymbol{P}_i$ at the corresponding circle point $\boldsymbol{C}_i$ (Figure 3). The circular representation of the contour is invariant by rotation, translation, and scaling. As the density of points imcreases, the circular representations of two contours are identical up to a rotation of the unit circle. This property allows for comparing contours by declaring that two contours are identical if there exists a rotation of the unit circle that brings their representation in correspondence. Also, when comparing contours, the distribution of

the vertices C_i on the circle must be uniform. We refer to this property as *global regularity*.

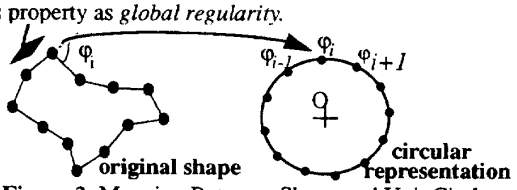


Figure 3: Mapping Between Shape and Unit Circle

3.0 Representation of 3-D Surfaces

In this section we extend the concepts of curvature measure, 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 6.0.) We follow the same approach as for two-dimensional contours. We introduce the three-dimensional equivalent of the concepts of global and local regularity in Sections 3.1 and 3.2, respectively. In Section 3.3, 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, in Section 3.4, we define an intrinsic spherical representation as an extension of the circular representation of contours. Detailed presentations of the basic results on semi-regular tessellations, triangulations, and duality can be found in [20][28][29].

3.1 Global Regularity and Mesh Topology

A natural discrete representation of a surface is a graph of points, or mesh, such that each node is connected to each of its closest neighbors by an arc of the graph. It is desirable in 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.

As mentioned in the previous section, global regularity can be easily achieved in two dimensions since a curve can always be divided into an arbitrary number of segments of equal length. It is well known that only approximate global regularity can be achieved in three dimensions. We use a quasi-regular triangulation constructed 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. For the experiments presented in this paper, we used a subdivision frequency of $N = 7$ for a total number of nodes of 980.

3.2 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 a curvature measure definition similar to the ones used for 2-D curves. The definition of local regularity in three dimensions is a straightforward extension of the definition of Section 2.0. 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 4). 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 2 by the tetrahedron (P_1, P_2, P_3, P). The local regularity condition is invariant by rotation, translation, and scaling.

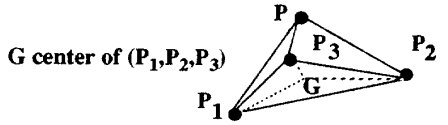


Figure 4: Local Regularity in Three Dimensions

3.3 Discrete Curvature Measure

The last step in building a discrete surface representation is to define a measure of curvature that can be computed from a mesh with the appropriate regularity properties. We need to define some notations (Figure 5 (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 P is in the plane (P_1,P_2,P_3) (Figure 5 (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. 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 in Figure 2 for a two-dimensional simplex, the triangle.

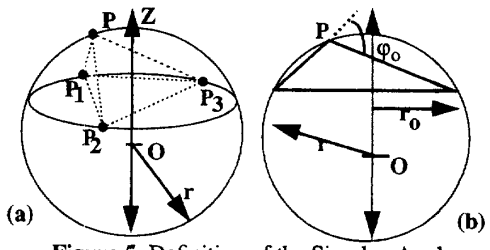


Figure 5: Definition of the Simplex Angle

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 outward. This behavior of the simplex angle corresponds to the intuitive notion of local "curvature" of a surface. It is clear that the simplex angle is invariant by rotation, translation, and scaling. In the remainder of the paper, the simplex angle φ_o at a node P will be denoted by $g(P)$.

It is important to note that other definitions of $g(P)$ are possible as long as the definition guarantees that $g(P)$ is invariant by rigid transformations and by scaling. In particular, any scale-independent curvature measure could be used instead of our simplex angle. We selected this definition because it is easy to compute and it is reasonably stable with respect to small variation of the surface.

3.4 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 a mesh of points on a surface such that it has the topology of the quasi-regular mesh of Section 3.1. 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 3.1), 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, $h(\boldsymbol{P})$, where $\boldsymbol{P}$ is a node of the spherical mesh, is the node of the object mesh that has the same index as $\boldsymbol{P}$.

Given h, we can store at each node $\boldsymbol{P}$ of $\mathcal{S}$ the simplex angle of the corresponding node on the surface $g(h(\boldsymbol{P}))$. 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 EGI, we call this representation the Simplex Angle Image (SAI). In the remainder of the paper, we will denote by $g(\boldsymbol{P})$ instead of $g(h(\boldsymbol{P}))$ the simplex angle associated with the object mesh node $h(\boldsymbol{P})$ 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 satisfied because the simplex angle itself is invariant by translation and scaling (Section 3.4), and because $\mathcal{M}$ still satisfies the local regularity condition after translation and scaling (Section 3.2).

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 $\boldsymbol{P}_1$ and $\boldsymbol{P}_2$ are connected on the spherical mesh, then the two corresponding nodes $\boldsymbol{M}_1=h(\boldsymbol{P}_1)$ and $\boldsymbol{M}_2=h(\boldsymbol{P}_2)$ 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.

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 $\mathcal{S}$ and $\mathcal{S}'$ 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.) This property holds even in the case of arbitrary non-convex objects because of the connectivity conservation property.

4.0 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 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 with reasonable accuracy.

The general approach is to first define an initial mesh near the object with the topology of Section 3.1 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.

4.1 Mesh Deformation

The problem is now to deform the mesh such that all the nodes satisfy two fundamental properties. First, mesh nodes must be as close as possible to the original surface. Second, 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.

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 $\boldsymbol{F}_o$ be the force of the first type applied at a given node N, and $\boldsymbol{F}_g$ be the force of the second type at the same node. If $\boldsymbol{P}_{t+1}$, $\boldsymbol{P}_t$, and $\boldsymbol{P}_{t-1}$ are the positions of node $\boldsymbol{P}$ at three consecutive iterations, the update rule is defined as:

$$\boldsymbol{P}_{t+1} = \boldsymbol{P}_t + \boldsymbol{F}_o + \boldsymbol{F}_g + D\,(\boldsymbol{P}_t - \boldsymbol{P}_{t-1})$$

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. 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.

$\boldsymbol{F}_o$ is defined by calculating the point $\boldsymbol{P}_c$ from the original surface that is closest to the node, that is: $\boldsymbol{F}_o = k\boldsymbol{P}\boldsymbol{P}_c$, 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 oscillations. In practice, k varies between 0.01 at the beginning of the iterations to 0.4 at the end of the iterations.

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 4), 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: $F_g = aPP_g$

To avoid unstable behavior of the system, the spring constant a should be between 0 and 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 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. 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.

Once the initial ellipsoid is generated, the mesh generation is completely independent of the actual format of the input data. 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 SAI building algorithm is illustrated in Figure 7 with range data as input and in Figure 9 with a polyhedral model as input. Figure 6 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 7 (a) and Figure 7 (b) show the initial mesh mapped on the ellipsoid and the mesh at an intermediate stage. Figure 7 (c) shows the final regular mesh on the object. Figure 7 (d) 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.

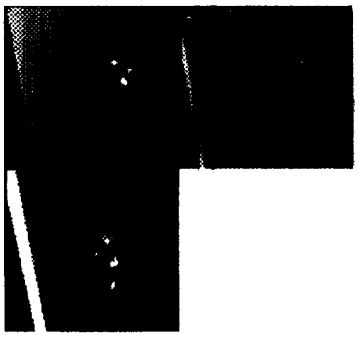
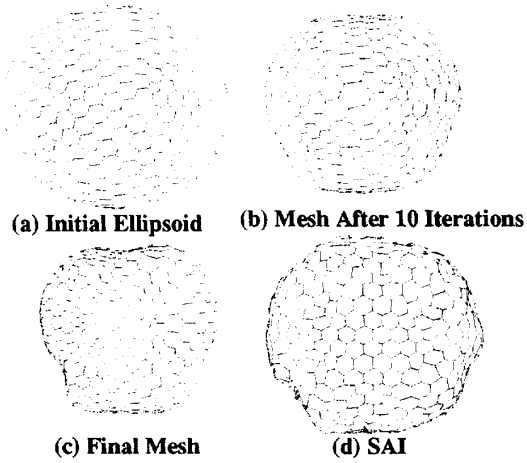


Figure 6: Three Views of an Object



(a) Initial Ellipsoid **(b) Mesh After 10 Iterations**

(c) Final Mesh **(d) SAI**

Figure 7: Building SAI from Range Data

Figure 9 (a) through Figure 9 (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 shown in Figure 8 rather than a set of data points. Once this intermediate representation is generated, the mesh deformation and SAI generation algorithms proceed in the same manner. The arrow between Figure 9 (c) and Figure 9 (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

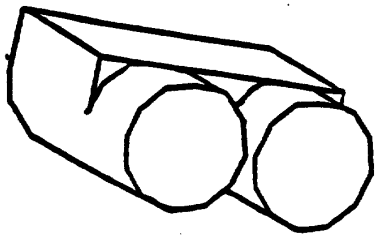
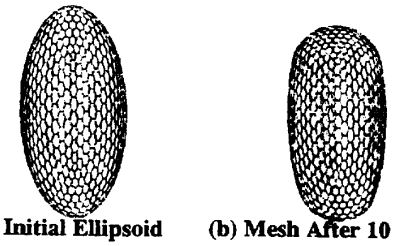
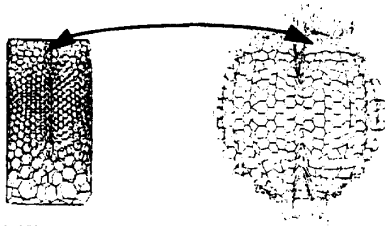


Figure 8: Geometric Model of an Object



(a) Initial Ellipsoid **(b) Mesh After 10 Iterations**



(c) Final Mesh **(d) SAI**

Figure 9: Building the SAI from a Polyhedral Model

5.0 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 3.0, 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 and to find the rotation yielding minimum distance is determined, the full 3-D transformations can be determined. This approach is expensive because it requires the testing of the entire 3-D space of rotations. 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 $\mathcal{S}$ and $\mathcal{S}'$ 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 $\mathcal{S}$, *resp.* P of $\mathcal{S}'$, $\mathcal{S}$ and $\mathcal{S}'$ are representations of the same object if there exists a rotation R such that $g'(P) = g(RP)$ for every point P of $\mathcal{S}'$. Since the SAI is discrete, $g(RP)$ is not defined because in general RP will fall between nodes of $\mathcal{S}'$. We define a discrete approximation of $g(RP)$, $G(RP)$, by interpolating the values of g at the four nodes of $\mathcal{S}'$ nearest to RP.

The problem now is to find this rotation using the discrete representation of $\mathcal{S}$ and $\mathcal{S}'$. This is done by defining a distance $D(\mathcal{S}, \mathcal{S}', R)$ 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:

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

The minimum of D corresponds to the best rotation that brings $\mathcal{S}$ and $\mathcal{S}'$ in correspondence. The simplest strategy is to sample the space of all possible rotations, represented by three angles (φ,θ,ψ), and to evaluate D for each sample value $(\varphi_i,\theta_i,\psi_i)$. 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 *spherical representations*. An additional step is needed to compute the actual transformation between objects as described below.

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 $\mathcal{S}'$ we compute the node P of $\mathcal{S}$ that is nearest to RP'. Let M, *resp.* M', be the point on the object corresponding to the node P of $\mathcal{S}$, *resp.* P'. A first estimate of the transformation is computed by minimizing the sum of the squared distances between the points M of the first object and the corresponding points $R_oM'+T_o$ of the second object. The resulting transformation is only an approximation because it assumes that the nodes from the two meshes correspond exactly. We use an additional step to refine the transformation by looking for the node M closest to M' for every node of the mesh and by computing again the optimal pose.

5.3 Reducing the Search Space

As mentioned in Section 5.1, the direct 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 . 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 $\Delta\varphi = \Delta\theta = \Delta\psi = 10°$, corresponding to a 36x18x36 search space at the coarse level. The

rotation space is then searched in an 18° 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.

5.4 Example

Figure 10 shows three views of the same object as in Figure 7 placed in a different orientation. A model is built from the three corresponding range images using the approach described in 4.3. Figure 11 illustrates the difference of pose between the two models computed from the two sets of images. Figure 11 (a) (resp. Figure 11 (b)) shows the superimposition of the cross sections of the two models in the plane YZ (resp. XY). Figure 12 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 angles 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° in both φ and θ. This display shows that there is a single sharp minimum corresponding to the rotation that brings the SAI in correspondence. Figure 13 (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 14 shows one of the models backprojected in the image of the other using the computed transformation. Figure 14 (a) is the original image; Figure 14 (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.

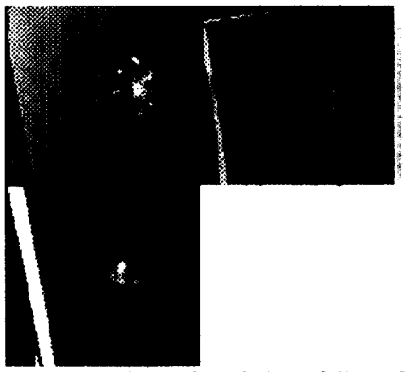


Figure 10: Three Views of the Object of Figure 7 in a Different Orientation

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

Figure 11: Relative Positions Before Matching

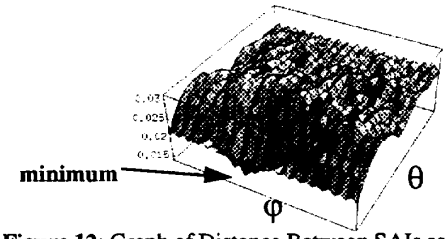


Figure 12: Graph of Distance Between SAIs as a Function of φ and θ

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

Figure 13: Relative Positions after Matching

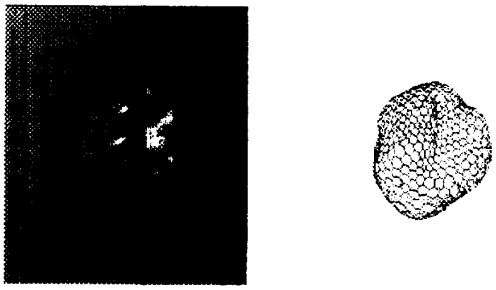


Figure 14: Display of the Model in the Computed Pose

6.0 Partial Views and Occlusion

Up to now we have assumed that we have a complete model of the object, as in Figure 9, or that we have data covering the entire surface of the object, as in Figure 7. 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 5.0 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.

The situation is illustrated in Figure 15 in the case of a two-dimensional contour. In Figure 15 (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 15 (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 15 (d), is greater than the length of the same section of the curve on the original representation, L^* shown in Figure 15 (c). In order to compute the distance measure of Section 5.0, 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 $k = L^*/L_2$. 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 $L1/L = L^*/2\pi$. 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. L^* can be eliminated from these two relations, yielding an expression of $k = 2\pi L_1/L_2 L$.

The situation is the same in three dimension except that lengths are replaced by areas A, A_1, A_2, A^*, with obvious notations. The previous expression becomes $k = 4\pi A_1/A_2 A$.

The direct extension from two to three dimension is only an approximation because the relation $A_1/A = A/4\pi$, 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

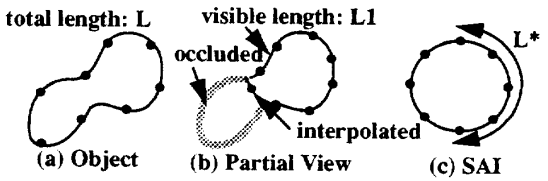


Figure 15: Matching 2-D Partial Representation

Once k is computed, the appropriate scaling is applied to the SAI by moving the nodes on the surface of the sphere given the scaling ratio k. After scaling, the distribution of nodes on the part of the SAI corresponding to the visible part of the object in the scene and the distribution in the corresponding region of the model SAI are identical.

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.

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 16 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 17 displays the graph of the distance between SAIs as function of rotation angles. Figure 17 (a) shows two views of the distance as a function of φ and θ. Figure 17 (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 18 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 7.

In the second example, the reference model is the CAD model of Figure 9. The observed scene is shown in Figure 19. The result of the matching is shown in Figure 20. Only part of the object is visible in the image because 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, as described in [8]. 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 21 (a) shows the SAI computed from the image of Figure 16, Figure 21 (b) shows the SAI after

the scaling is 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.

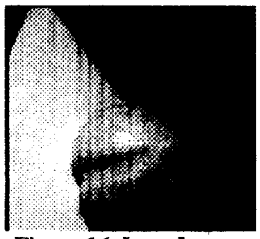
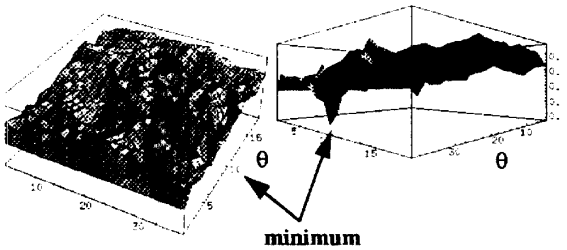
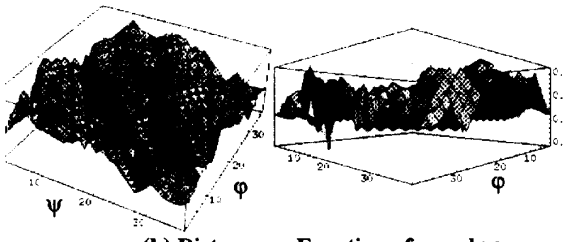


Figure 16: Input Image

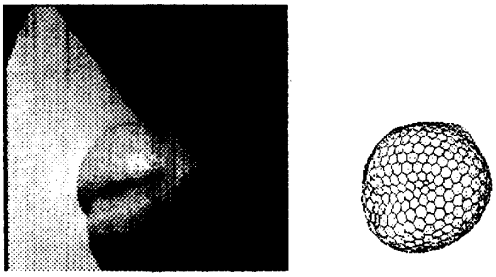


(a) Distance as Function of φ and θ



(b) Distance as Function of φ and ψ

Figure 17: Sum of Squared Differences of SAIs as Function of Rotation Angles



(a) Input Image **(b) Model after Transformatio**

Figure 18: Display of Model Using the Pose Computed from the Matching

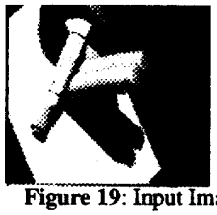
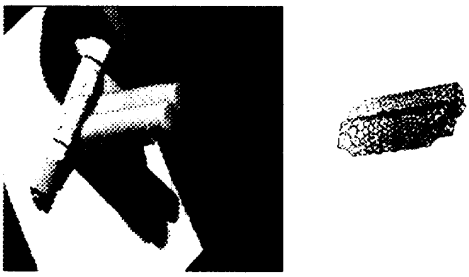
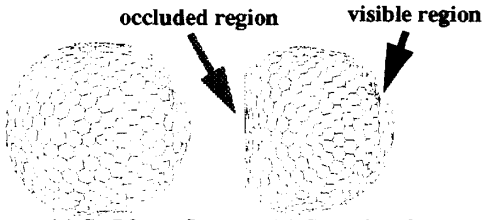


Figure 19: Input Image



(a) Input Image **(b) Model after Transformation**

Figure 20: Display of Model Using the Pose Computed from the Matching Using the Image of Figure 19 and the Model of Figure 9



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

Figure 21: Effect of Occlusion-Compensating Scaling on SAI of Observed Object

7.0 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 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 for non-convex objects as other representations. The SAI representation can handle partial views and occluded objects thanks to the property of connectivity conservation of the 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 natural shapes..

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 important extension is the use of appearance information such as hue or albedo, in addition to geometric information. Such information can be included in the SAI representation by storing appearance data at every node of the SAI in addition to the curvature measure. Appearance data can be incorporated in the distance measure between SAIs to help disambiguate between shapes that are similar geometrically but have different appearance features. We are now pursuing such an approach using color images.

8.0 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*", PAMI-14(2), 1992

[3] Besl, P., Jain, R., "*Segmentation Through Symbolic Surface Descriptions*", Proc. ICCV'86, pp 77-85, Miami, 1986

[4] Bobick, A.F., Bolles, R.C., "*The Representation Space Paradigm of Concurrent Evolving Object Descriptions*", PAMI, Vol. 14, No. 2, pp. 146, 1992

[5] Bolle, R.M., B.C. Vemuri, "*Three Dimensional Surface Reconstruction Methods*", PAMI, 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 10 (3), April 1992

[9] Faugeras, O.D. and Hebert, M., "*The representation, recognition, and locating of 3-D objects*", The International Journal of Robotics Research, Vol. 5, No. 3, pp. 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*", PAMI, Vol. 13, October 1992

[12] Grimson, W. E. L. and Lozano-Perez, T., "*Localizing Overlapping Parts by Searching the Interpretation Tree*", PAMI-9(4), July 1987

[13] Horn, B.K.P., "*Extended Gaussian Images*", Proc. of the IEEE, 72(12), December 1984

[14] Ikeuchi, K., "*Recognition of 3-D objects using the extended Gaussian image*", International Joint Conf. on Artificial Intelligence, 595-600, 1981

[15] Ikeuchi, K. and Hong, K.S., "*Determining Linear Shape Change: Toward Automatic Generation of Object Recognition Program,"CVGIP:IU, 53(2), pp.154-170, March 1991*

[16] Kang, S.B. and Ikeuchi, K.,"*Determining 3-D Object Pose using the Complex Extended Gaussian Image,*" Proc. CVPR-91, Lahaina, Maui, Hawaii, June 1991

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

[18] 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

[19] Little, J.J., "*Determining object attitude from extended Gaussian images*", Proc. of 9th Intern. Joint Conf. on Artificial Intelligence, 960-963, 1985

[20] Loeb, A.L., "*Space Structures*", Addison-Wesley, 1976

[21] Lowe, D.G., "*Three-dimensional object recognition from single two-dimensional images*", Artificial Intelligence, 31, 1987, 355-395

[22] Oshima, M. and Shirai, Y., "*Object recognition using three-dimensional information*", PAMI-5(4), July 1983

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

[24] Stein, F., Medioni, G., "*Structural Indexing: Efficient 3-D Object Recognition*", PAMI, Vol. 14, No. 2, 1992

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

[26] 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.

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

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

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