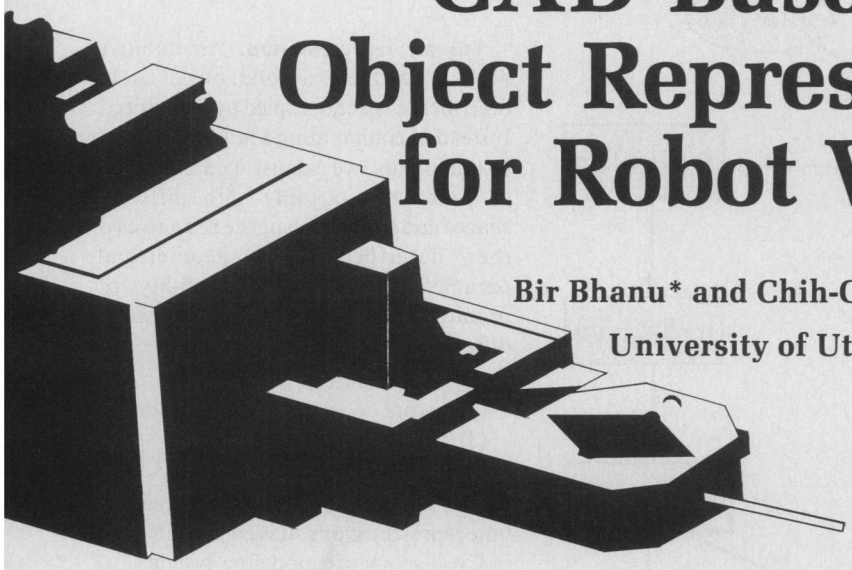
CAD-Based 3D Object Representation for Robot Vision



Bir Bhanu* and Chih-Cheng Ho

University of Utah

Computer vision researchers lack a systematic approach for building object models for industrial environments. We propose a CAD-based approach for building representations and models for applications involving 3D object recognition and manipulation.

In the automated manufacturing system shown in Figure 1 are three key components: the CAD/CAM (computer-aided design/computer-aided manufacturing) system, the vision system, and the intelligent robot system. The CAD/CAM system supports the design, analysis, and manufacturing of each part of a product. The vision system integrates information from sensors such as TV cameras, laser range finders, tactile and ultrasonic sensors. It provides the robot with information about the working environment and the location, identity, and quality of the designed parts. The intelligent robot aligns the inspected parts and performs assembly operations using tactile and force-torque sensors.

Most existing vision systems rely on models generated in an ad hoc manner and have no explicit relation to the CAD/CAM system originally used to design and manufacture these objects. We desire a more unified system that allows vision models to be automatically generated from an existing CAD database. A CAD system contains an interactive design interface, graphic display utilities, model analysis tools, automatic manufacturing interfaces, etc. Although it is a suitable environment for design purposes, its representations and the models it generates do not contain all the features that are important in robot vision applications.

Current vision systems use only one representation in their models. However, there is no single representation or a matching technique based on a single representation that can efficiently and reliably recognize different classes of objects. A systematic approach for building vision models employing multiple representations is to derive them from a CAD database and incorporate features crucial for object recognition and manipulation.[1]

In this article, we propose a CAD-based approach for building representations and models that can be used in diverse applications involving 3D object recognition and manipulation. There are two main steps in using this approach. First, we design the object's geometry using a CAD system, or extract its CAD model from the existing database if it has already been modeled. Second, we develop representations from the CAD model and construct features possibly by combining multiple representations that are crucial in 3D object recognition and manipulation.

In this work we used the Alpha_1 solid

*Bhanu is now at Honeywell Systems & Research Center.

0018-9162/87/0800-0019$01.00©1987IEEE

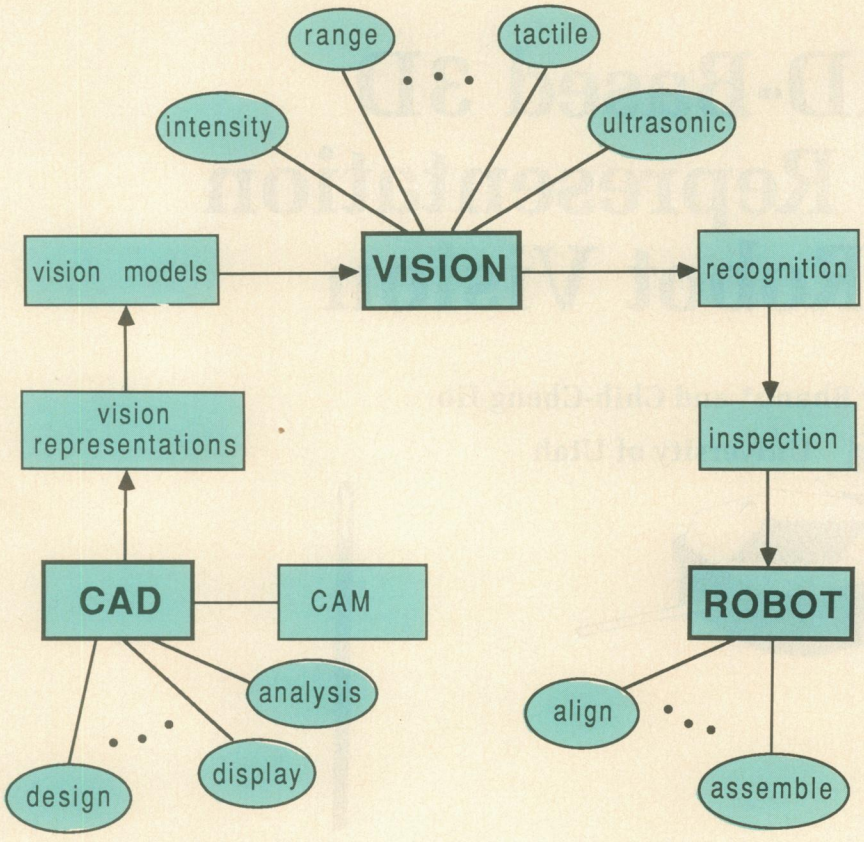


Figure 1. An automated manufacturing system.

modeling system[2] developed at the University of Utah. It utilizes spline-based boundary representation. We present the following six CAD-based representations:

(1) surface points and normals,
(2) surface curvatures,
(3) generalized sweep,
(4) polyhedral,
(5) extended Gaussian image, and
(6) object decomposition and hierarchical representation.

The construction of vision models that organize and use these representations efficiently requires consideration of both image feature-extraction algorithms and feature-matching techniques. We do not include them here. These details can be found in a recent survey on model-based robot vision by Chin and Dyer.[3]

3D object representations

A representation for shape as defined by Marr[4] "is a formal scheme for describing shape or some aspects of shape together with rules that specify how the scheme is applied to any particular shape," and "the results of using a representation to describe a given shape is called a description of the shape in that representation." It is always possible to describe a given shape using different representations. The choice of a representation to obtain an efficient description depends not only on the kind of object to be described, but also on how the description is to be used.

Three general classes of 3D solid object representations used in computer graphics, CAD, and computer vision are

(1) volume,
(2) sweep, and
(3) surface or boundary.

Besides these, other 3D object representations are used in computer vision. A useful representation is the extended Gaussian image (EGI), in which each face of an object is mapped onto a unit sphere, called the *Gaussian sphere*, according to its orientation and area. Here we do not consider ambiguous CAD schemes, such as the wireframe representation, which can be perceived as more than one object from a given description.[5]

Volume representation. An intuitive way to represent a solid object is to describe the space occupied by that object. Instead of enumerating a huge number of spatial points, we can use a combination of primitives, possibly with different shapes and sizes. Each of the primitives is then described by its geometrical parameters. Increasing generality of primitives and their combinations lead to different schemes:

(1) pure primitive instancing,
(2) spatial occupancy enumeration,
(3) cell decompositions, and
(4) constructive solid geometry (CSG).

CSG is a super-set of the other three volume representations. It is commonly used in CAD geometric modeling systems.

CSG represents an object as a binary tree (see Figure 2) where each leaf represents an instance of a primitive and each node represents an operation of its descendent(s).

Primitives such as blocks, spheres, cylinders, and cones are first transformed (translation, rotation, and scaling) and then combined (union, intersection, or difference) from the bottom to the top of the tree. Regularized set operators[5] help ensure the regularity and the validity of the combined object so that there are no dangling edges or faces.

CSG representation is sufficient to cover most conventional, unsculptured objects. However, it cannot describe a large class of sculptured, freeform shapes precisely, even using a fairly large number of primitives. For unsculptured objects that contain primitives as their subparts, CSG can give concise descriptions. Similar objects may have the same subtrees, and different configurations of primitives may yield different objects. This representation is unambiguous but not unique, i.e. it can have different decompositions of an object using the same set of primitives.

The greatest drawback of using CSG in robot vision applications is that surface evaluation is always required because we can only see the object's surface. The surface evaluation is computationally intensive. Moreover, since the primitives are not equal to the subparts of an object in general, we may not be able to perceive or derive any of the primitives from the object's surface. It makes the reconstruction of the CSG tree from a given scene

and the use of its topological information almost impossible.

Sweep representation. A generalized sweep, or generalized cylinder (GC) or cone, is defined as the volume swept by a set of cross sections along an axis under some sweeping rule. It was first introduced by Binford[6] in computer vision for the recognition of 3D curved objects. According to the formal definition of GC given by Shafer,[7] a GC consists of four parts:

(1) There is a space curve, called the axis of the shape.

(2) At each point on the axis, at some fixed angle to the tangent to the axis, there is a cross-section plane.

(3) On each such cross-section plane, there is a planar curve which constitutes the cross-section of the object on that plane.

(4) There is a transformation rule that specifies the transformation of the cross-section as the cross-section plane is swept along the axis. This rule always imposes (at least) the constraint that the cross-section changes smoothly.

The surface of the object is the union of these cross sections and the volume swept by the closed cross-sectional curves is the GC. With different restrictions on the axis and the cross sections together with their intersections and the manner in which cross sections blend, there is an exhaustive taxonomy of different classes of GCs.

Sweep representation is well suited for many manmade objects that have an axis of symmetry. It is very concise. Similar objects have similar axes and cross sections. A small difference between similar objects will be reflected by a small difference in the axes or cross sections.

Sweep representation is not unique in general. Although it is unambiguous under a given blending rule, there is no efficient way to describe the shapes at both ends for an arbitrary GC. We cannot precisely estimate the axis and cross sections without seeing the whole surface. However, we can use the axis alone in object recognition. By using hierarchical grouping of GCs, it is possible to describe more complex shapes. But to get a unique and reasonable decomposition of an object is a difficult problem.

GCs have been used only to a very limited extent in CAD.

Boundary representation. To represent an object by its enclosing surface or boundary is the most commonly used scheme in both computer graphics and computer vision. A solid is represented by segmenting its boundary into a finite number of bounded subsets, usually called *faces* or *patches*.

If we use planar patches only, each face is represented by its edges and vertices, resulting in polyhedral or polyhedron-approximated objects. In order to describe curved surfaces efficiently, splines are used in many CAD systems. However, some geometrical computations are expensive in spline representation. For example, finding the intersection points of a line and a surface or the intersection curves between two surfaces is not easy in general. Polygonal approximations are still used in most applications.

Boundary representations (B-rep) can be derived from 3D range data directly. In fact, the data itself is a description of the object's shape in the form of surface points. (For an example of the advanced 3D range sensing technology available today, see the accompanying sidebar.) B-rep is unambiguous but the validity is not guaranteed and it is also not unique. Different tessellations of the surface and different polygonal approximations of each patch can still give the same object. Spline-embedded surface representation gives a concise and optimal approximation of the true shape.[8] However, finding high-level features is not easy and there are no spline forms for some computations, such as surface curvatures.

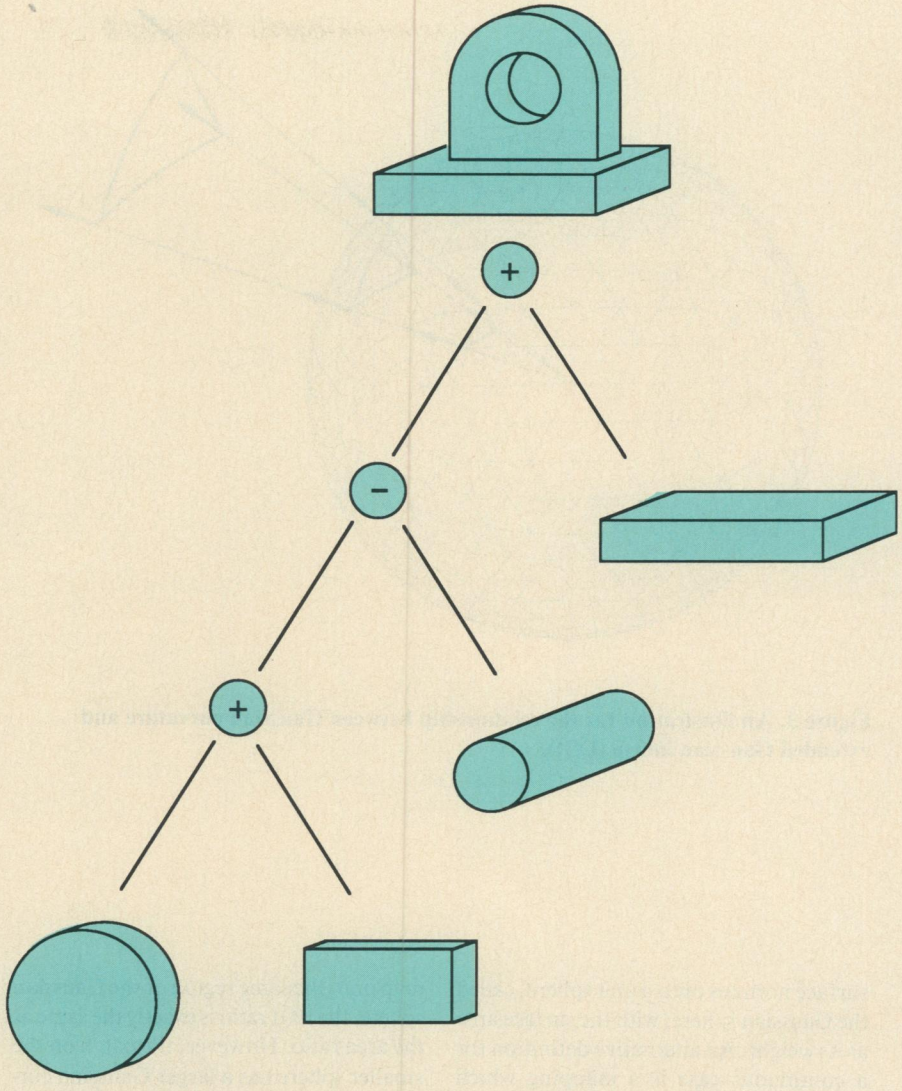


Figure 2. An example of constructive solid geometry (CSG) representation.

Extended Gaussian image. Extended Gaussian image[9] (EGI) is a mapping from

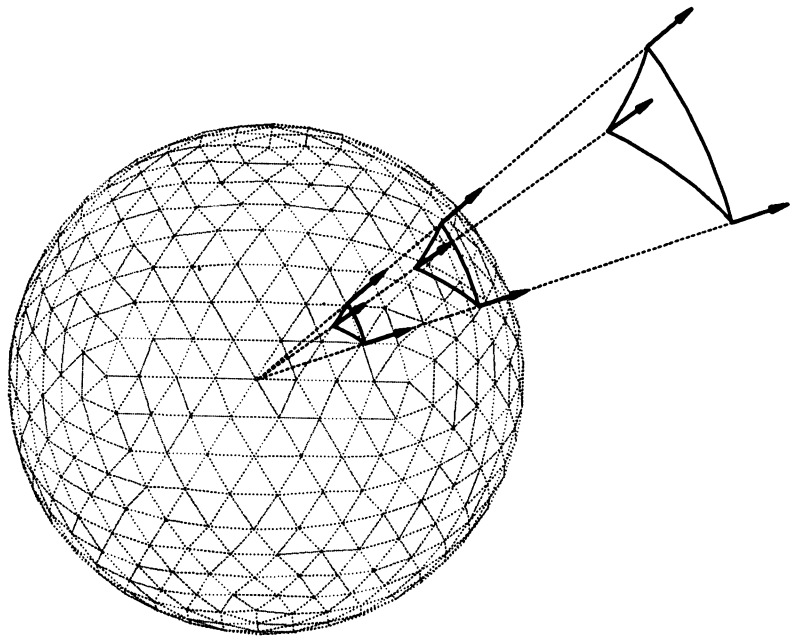


Figure 3. An illustration for the relationship between Gaussian curvature and extended Gaussian image (EGI).

surface normals onto a unit sphere, called the Gaussian sphere, with the surface area as its weight. An alternative definition for a continuous case is a mapping which associates the inverse of the Gaussian curvature at a point on the surface of the object with the corresponding point on the Gaussian sphere. In discrete cases such as polyhedra, EGI can also be obtained by placing a weight at each point equal to the sum of the surface area of the faces in the corresponding orientation. It can be computed easily and is used to find the object's orientation.

The inverse relationship between EGI and Gaussian curvature can be understood using the concept of Gaussian curvature, as the spread of surface normals, that is equal to the area on the Gaussian sphere mapped from a unit area on the object's surface. Conversely, EGI is the total area on the object's surface mapped onto a unit area on the Gaussian surface.

A simple example is the EGI and Gaussian curvature on spheres of different radii. Shown in Figure 3 are three concentric spherical triangles with radii of 0.5, 1, and 2. The ratio of the areas of these triangles is 0.25 : 1 : 4. Since all the triangles map onto the same region of the Gaussian sphere, the EGI ratio is exactly the same as the area ratio. However, the patch on the smaller sphere has a larger Gaussian curvature. It is not hard to figure out that for a sphere of radius r, the Gaussian curvature is $1/r^2$ and the EGI is r^2. A full mathematical proof of the inverse relationship is given by Horn.[9]

EGI is used in the analysis, but not the synthesis, of surface shapes. It has the following properties:

(1) The center of mass of the extended Gaussian image is at the origin of the Gaussian sphere.

(2) The total mass of the EGI equals the total surface area of the object.

(3) It is unique for any convex object.

(4) For a convex object, the weight of each point of the EGI is equal to the inverse of the Gaussian curvature at the corresponding point on the original surface.

(5) EGI does not depend on the position of the object. It allows us to determine the object's orientation before knowing its position. Since it represents objects explicitly by their orientation, the degree of freedom is reduced, from 6 (position plus orientation) to 3 (orientation only).

(6) The rotation of the object does not affect the relative weight distribution of its EGI.

For concave objects, different faces may have the same orientation and are mapped onto the same cell on the Gaussian sphere. Therefore, EGI representation is not unique. Ikeuchi[10] used a global approach for concave objects. Instead of mapping the whole object onto one EGI, he selected 60 different viewer directions and mapped the visible surfaces from each of these views onto an EGI. This is not only inefficient in storage (60 EGIs for one object), but also it cannot deal with occluded objects. In general, a local approach in which we decompose the boundary into patches and use one EGI for each patch seems better. However, subdividing the object to get a unique EGI for each part is a difficult problem.

Evaluation of 3D object representations for recognition requirements. The problem of 3D object recognition can be defined as follows[11]:

Given

(a) digitized sensory data corresponding to one particular, but arbitrary, view of the real world (may be in a certain, known environment) and

(b) knowledge, or models of a set of distinguishable objects.

Then find the following for each object in the set:

(1) Does the object appear in the digitized sensory data? If so, how many times does the object occur?

(2) For each occurrence of the object determine its location and orientation with respect to a known coordinate system.

Marr[4] has given a set of criteria for the representation of a shape, used in object recognition:

Scope: What kind of objects can it represent?

Accessibility: Can it be obtained inexpensively from the image?

Conciseness: Can it describe objects efficiently?

Uniqueness: Can a unique description of an object be obtained under different conditions?

Stability: Can it reflect the similarity between different objects?

Sensitivity: Can it reflect the differences between similar objects?

A summary of the 3D object representations using Marr's criteria is given in Table 1.

3D range sensor-phase shift detection

Robert E. Sampson
Environmental Research Institute of Michigan

Three-dimensional imaging sensors and their informational content have been under investigation for many years, with two main sensor techniques, *triangulation* and *time of flight*, recognized as the fundamental technical approaches to the problem. The triangulation method can be implemented in a variety of ways, including stereographic, structured light, etc., all of which present a number of complex problems that need to be overcome. A preferred concept that greatly simplifies and enhances range imaging uses a modulated laser diode to optically directly measure the range to each point in the scene. In original envisioned forms of such a system, a pulsed laser is directed at the scene and the time until a returned pulse is received is determined, which is proportional to the range to the object.

In a practical adaptation of this concept recently developed at the Environmental Research Institute of Michigan (ERIM) shown schematically in Figure 1, the difference in phase between the transmitted and received signals is determined rather than the time of flight. The major components of the sensor in Figure 1 include the laser diode and modulation source, a scanning mechanism (either polygon or nodding mirror), a photo detector, and the phase shift detection electronics. The difference in phase between the transmitted and received signals is determined and the phase shift is directly related to range. This sensor yields both a normal reflectance image and an image in which each pixel value is directly proportional to the distance to the pixel area.

This method of obtaining 3D images, often referred to as optical radar, has a wide variety of applications, including navigation, robot guidance, and inspection. Although these tasks may seem similar at first glance, the requirements differ greatly.

At ERIM, the phase-detection method of 3D imaging has been employed in the development of a family of sensors. One of these sensors was developed for autonomous vehicle navigation. This navigation sensor has a field of view (FOV) of plus or minus 40 degrees in the horizontal plane and covers a vertical FOV at depression angles of 15 to 45 degrees. It has a range resolution of 8 centimeters.

Another, more advanced, navigation sensor uses multiple lasers that operate at various frequencies in the visible, near infrared, and shortwave infrared wavelengths. These multiple wavelengths allow not only determination of the range to the terrain, but also the reflectance values of the materials in the scene. This sensor has a FOV of 60 × 80 degrees and a

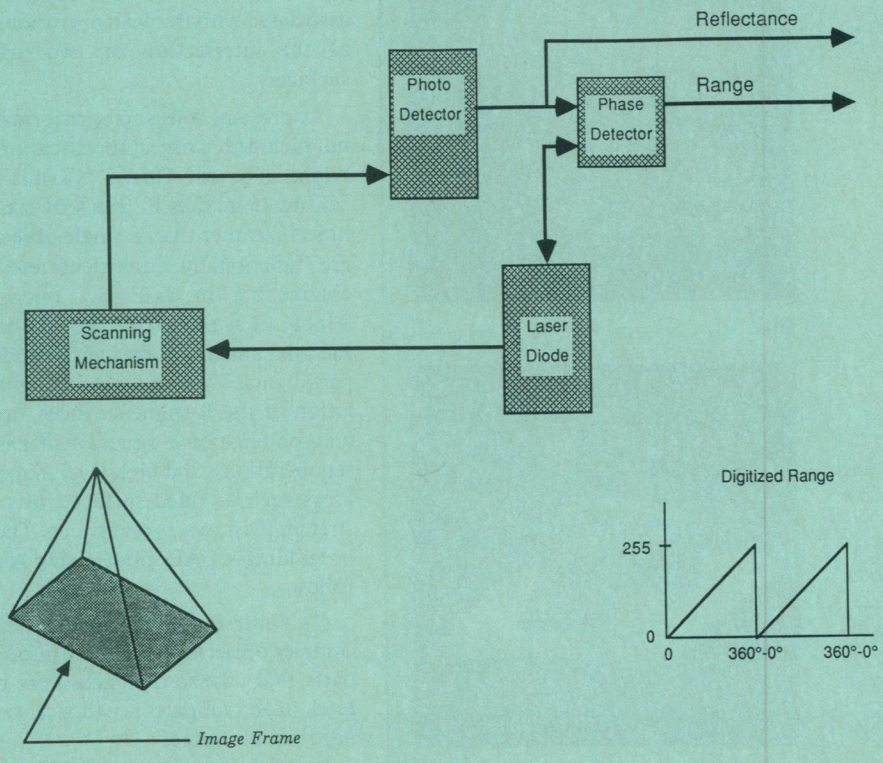


Figure 1. 3D laser scanner simplified block diagram.

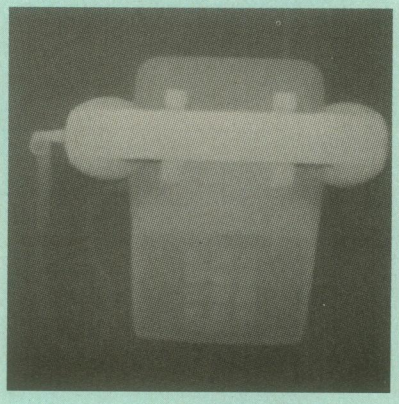
range resolution of 2 centimeters.

A third sensor developed by ERIM has been employed to perform bin picking and other robot guidance tasks. This sensor has an adjustable FOV ranging from 1.6 × 1.6 degrees to 35 × 35 degrees and can operate over distances from 15 centimeters to over 90 centimeters. In the robot guidance mode the sensor has a resolution of 0.08 centimeters. It can also operate in an inspection mode, where it has a resolution of 0.003 centimeters. This sensor consists of three modules: the optics head, power supply, and electronic modules. The optics head can be monitored on a robot arm and connected via remote cables to the rack-mounted electronics and power supply.

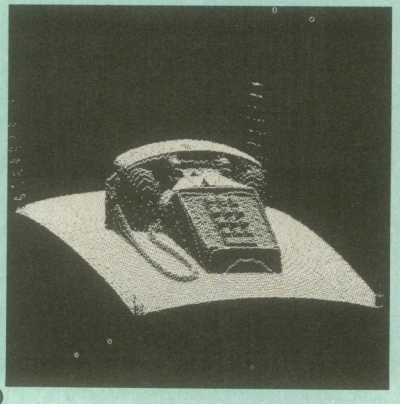
This sensor can provide range data at 100,000 measurements per second taken in a programmable scan pattern (any image size from one pixel to 1000 × 1000 pixels can be programmed). Image collection takes less than a second to over 10 seconds, depending upon image size. While image acquisition is slower than normal visible cameras, for range imagery the system is faster overall because we need not perform complex computational analysis to determine range—it is a direct output from the sensor.

Figure 2 includes an example of range imaging from the robot guidance sensor. Figure 2a is a normal visible image of a telephone. Figure 2b is a range image from the same view. In Figure 2b, the value of each pixel is directly proportional to range (light tone is closer, dark tone further away) rather than a measure of the intensity of reflections as in Figure 2a. In Figure 2c, the telephone image is processed into a perspective view using an ERIM-developed "cytocomputer" that provides high-speed range-image processing capability for the sensor.

(a)



(b)



(c)

Figure 2. Visible image of a telephone (a), approximately 30 × 30 centimeters taken with a normal photographic camera. Range image of a telephone (b), 300 pixels by 300 pixels by 8 bits—25 × 25 × 20 centimeters deep—taken in 10 seconds with the range sensor. Perspective view from the range image (c), with the range image of (b) rotated by cytocomputer to give a different perspective of the scene; processing time, 0.4 seconds.

Model building using the Alpha_1 CAD system

We use the CAD system Alpha_1, an advanced experimental solid modeling system[2] developed at the University of Utah. Alpha_1 models the geometry of solid objects by representing their boundaries as discrete B-splines. B-splines are an ideal design tool, simple yet powerful. Many common shapes can be represented exactly using rational B-splines. For example, all of the common primitive shapes (spheres, cylinders, ellipsoids, etc.) used in CSG systems fall into this category. Other advantages include good computational and representational properties of the spline approximation: the variation diminishing property, the convex hull property, and the local interpolation property.

The single underlying mathematical formulation of Alpha_1 simplifies implementation, but it is sufficiently powerful to represent a very broad class of shapes. It is able to create images of the designed objects, to perform certain analysis functions on them, and to produce information for numerically controlled machining. It uses the Oslo algorithm,[12] for computing discrete B-splines. Subdivision, effected by the Oslo algorithm, supports various capabilities including the computation associated with Boolean operations, such as the intersection of two arbitrary surfaces.

At present, tolerancing information is not included in object specification in the Alpha_1 system. Once it is available, we can make models in terms of classes of objects (rather than a single object) that are functionally equivalent and interchangeable in assembly operations. Alpha_1 has powerful shape description facilities and it supports several modeling paradigms. These include direct manipulation of the B-spline surfaces, creation and combination of primitive shapes using set operations, and high-level shape operators such as ruled surface, loft, bend, stretch, twist, warp, and sweep. The steps in building a CAD model using Alpha_1 follow:

(1) *Analysis of the object.* Usually a complex object is decomposed into simpler parts that can be designed more easily. Each of the subparts is called a *shell* and need not be closed.

(2) *Design of parts and measurement of*

parameters. Geometric operators are used to design each subpart. Sometimes one shape can be designed using different paradigms, which may require a different set of parameters. For example, an arc can be specified by its center point and two end points. It can also be specified by two tangent lines and its radius. We want to use those parameters that can be measured easily and precisely.

(3) *Validation of designs.* When designing the surface patches of each part, ensure that they have the correct orientation and that the adjacency information between patches is correct. Otherwise an invalid object may have been created and the combiner (used in the next step) will be unable to manipulate it. At present, the designer makes explicit the correctness of orientation and adjacency. We want the system to be able to generate such information automatically.

(4) *Application of the combiner.* The last step is to put all the parts of the object in the correct positions and orientations by performing appropriate transformations, then use the combiner to perform the required set operations on them. This results in the design of the complete object.

Two examples demonstrate some of the modeling paradigms of the Alpha_1 system.

Example 1: green piece. To design simple objects such as the "green piece" in Figure 4a, which has many local features, we build the complete object in a stepwise manner. First, we design the plate and all the holes, then the dent part and the scratches shown in Figure 4b. To design these parts, we first design curves using B-splines, then use various high-level operators for surface construction, such as revolving a curve about an axis, extruding a curve in some direction, and filling the surface between two curves.

There are seven holes with threads in the green piece. We design each of these by filling two surfaces between two twisted curves. Figure 4c shows the line drawing and shaded display of the completed CAD model.

Example 2: Renault piece. For objects like the automobile "Renault piece" in Figure 5a, which contains sculptured surfaces, it is still possible to divide it into a set of simpler parts, although the decomposition may not be obvious. Here we divide it into five subparts in Figure 5b: small right head (upper left), base plate (upper right), left head (lower left), back bump (lower center), and neck (lower right). For the right head and the left head, we find all sharp edges and then construct the surfaces from them as we did in designing the green piece. For the base plate, the neck, and the back bump, first we design some pseudo edges, which are the intersection of the surface planes. Then we construct these surfaces but leave small gaps between them where cubic patches are used to produce the rounded edges. Figure 5c shows the intersection curves of these parts, which are computed to obtain the complete object using set operations performed by the combiner. Figure 5d shows the completed CAD model of the Renault piece.

Note that Alpha_1 can be used to model a large class of sculptured mechanical parts that are not representable by either a CSG or a sweep model. Although the use of nonuniform rational B-splines allows significant flexibility in the geometric modeling, spline representation does not explicitly exhibit important features used in most recognition techniques. Thus we want to construct descriptions based on other vision representations from this CAD model.

Table 1. Evaluation of 3D object representation.

Criterion	CSG	Sweep	B-rep	EGI
Scope	Fair	Fair	Good	Good
Accessibility	Poor	Poor	Good	Good
Conciseness	Fair	Good	Fair	Poor
Uniqueness	Poor	Poor	Fair	Poor
Stability	Fair	Good	Good	Fair
Sensitivity	Fair	Fair	Good	Fair

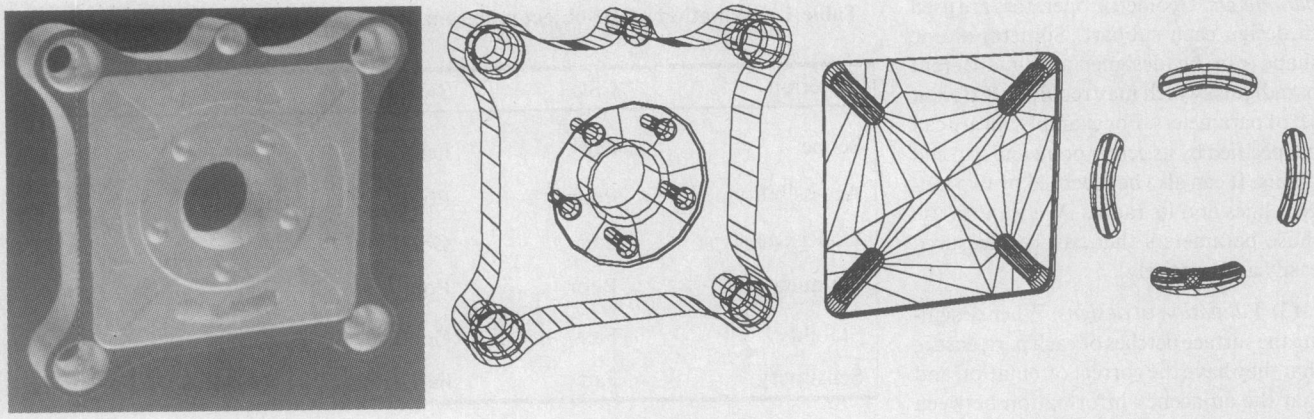
CAD-based 3D object representations

In this section we derive vision representations from the Alpha_1 CAD models. These representations can then be integrated into a vision model employing multiple representations. Appropriate models are used based on the results from different recognition tests.[1,13]

Surface points and normals representation. Surface points representation provides a universal discrete description of the object's boundaries. However, it requires a large amount of data to describe a given surface. Note that in a spline-based system, the surface normal is a byproduct of the surface (point) evaluation procedures, called knots insertion, spline refinement, or subdivision.

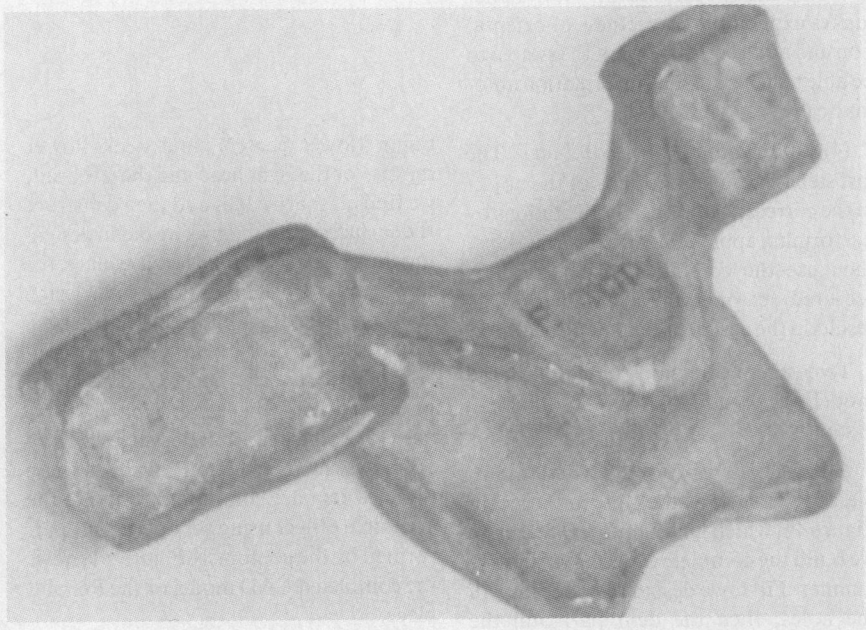
Although this representation does not carry more information than the original CAD model, it provides the ability to communicate with other vision modules that create higher level descriptions based on data in this format. For example, region growing and edge detection algorithms are commonly used with 3D range data and matching is done on the extracted symbolic features. [14,15] Moreover, this representation generates synthetic data for arbitrary shapes as well as for regular objects (cubes, spheres, cylinders, etc.). For an example of the use of surface points to parts localization problem in manufacturing, refer to the article by Gunnarsson and Prinz[16] in this issue of *Computer*.

Surface points extraction. A simple technique to extract surface points from B-spline patches uses the subdivision method.[1] In this technique a B-spline patch is first subdivided into smaller pieces that are within the given resolution, then the centroid of each of these small patches is computed. The points extracted by this

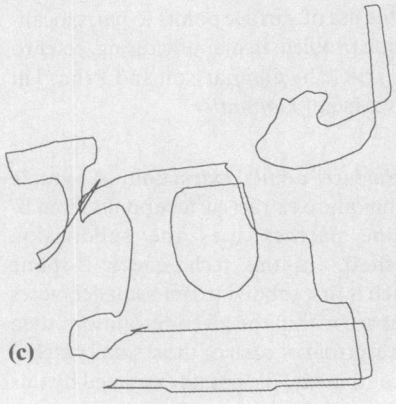


(a)

(b)



(a)



(c)

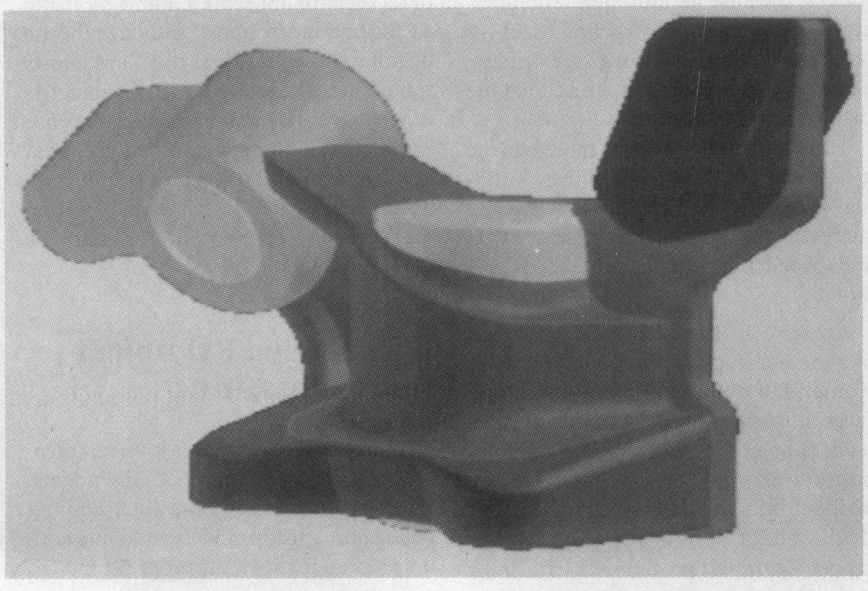
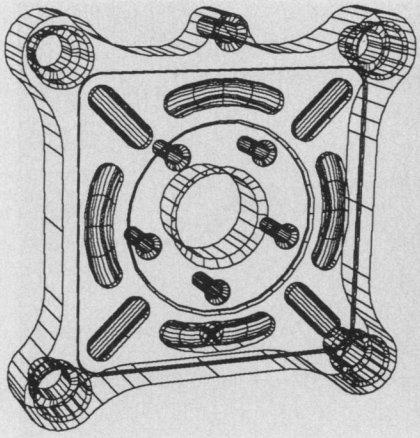
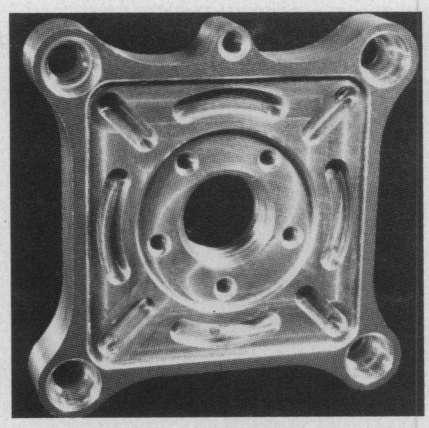


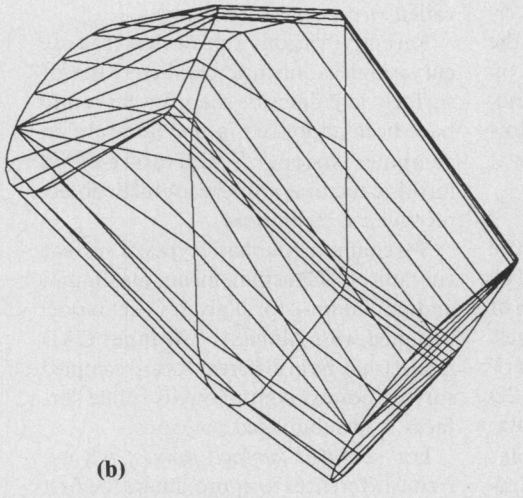
Figure 5. Design of the Renault piece using the Alpha_1 CAD system. (a) shows the Renault piece object, (b) shows the subparts of the Renault piece CAD model, (c) shows intersection curves of the subparts of the Renault piece, and (d) shows the designed CAD model of the Renault piece.



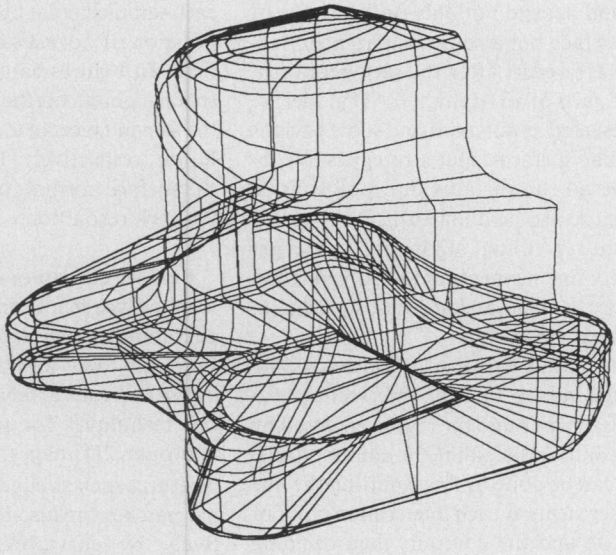


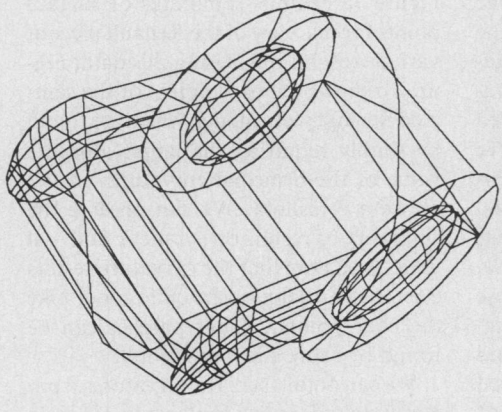
(c)

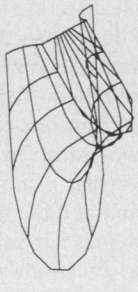
Figure 4. Design of the green piece using the Alpha_1 CAD system. (a) shows the green piece object, (b) shows the subparts of the green piece CAD model, and (c) shows the designed CAD model for the green piece.

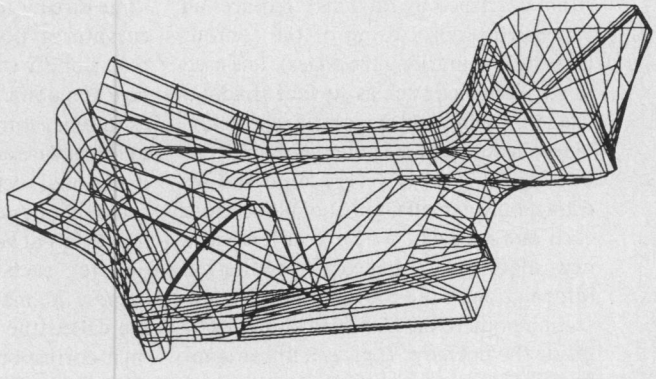


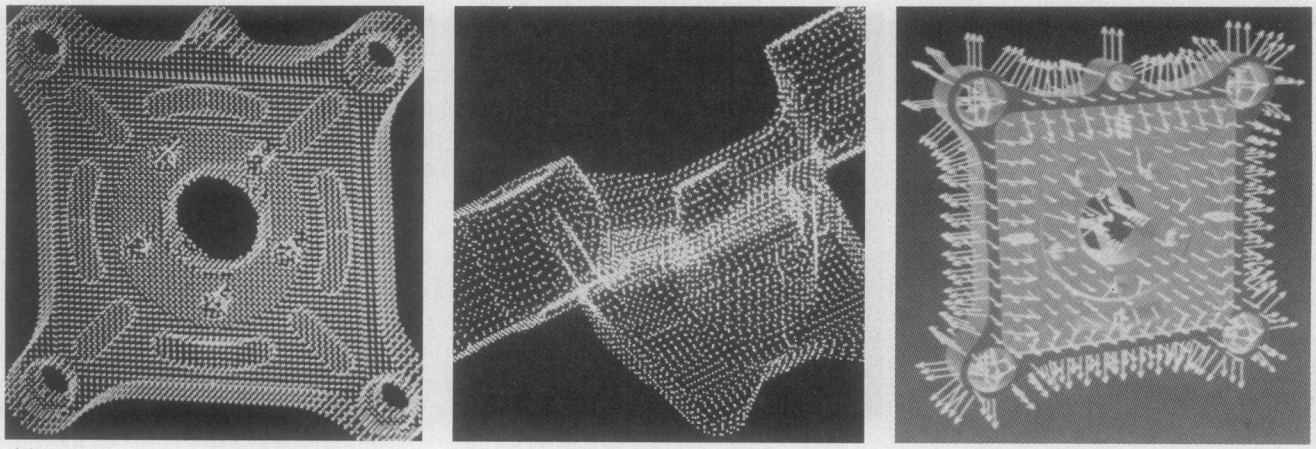
(b)











(a) (b) (c)

Figure 6. Surface points and normals representation. (a) shows surface points on the green piece (0.1-inch resolution), (b) shows the surface points on the Renault piece (0.2-inch resolution), and (c) shows the surface normals on the green piece (0.4-inch resolution).

method depend not only on the shape of the surface but also on its parameterization. However, after the set operations, some parts of an Alpha_1 CAD model are represented as polygons and some parts as B-spline surfaces. Our strategy is to subdivide all the surfaces into polygons to make the problem uniform. By applying a contour-filling algorithm,[17] we get interior line segments of the polygons and then extract points along these segments at a desired resolution.

The main element of a contour-filling algorithm is to find the intersection segments of a line and the region enclosed by that contour, including the contour itself. This can be done by first splitting the line into segments at each intersection point of the line and the contour, then deciding which of these segments lie inside the region.

An edge-based contour-filling algorithm described by Pavlidis[18] requires an expensive preprocessing of the contour (sorting and marking the edges). It is used in applications, such as surface shading, where the same contour is used repeatedly. Since, in our case, the number of polygons in a model is usually very large and we extract only a small number of points from each one of them, we have developed a new algorithm[17] that uses topological information of the contour at each intersection point to decide which segments lie inside the polygon. It gives a linear computational complexity in the average case. We obtain the surface normal vector at each sampled point by using bilinear interpolation of normals at the adjacent vertices. In Figures 6a and 6b we show the surface points on the green piece and on the Renault piece at 0.1- and 0.2-inch resolution, respectively. In Fig. 6c, we show the surface normals on the green piece at 0.4-inch resolution.

Surface curvature representation. The local surface shape can be characterized by curvatures, which combine information of both the first and second derivatives. These derivatives have been used in various techniques for segmentation of 2D contours, 2D images, and 3D range data. Features such as edges, corners, and planar patches can also be defined quantitatively by curvatures. In differential geometry,[8] the curvature of a 2D contour is defined as the change in the tangent vector per unit length. If this change has the same direction as the normal vector, the curvature is positive, otherwise it is negative. For 3D surfaces, the normal curvature at a point in one direction is defined as the curvature of the intersection curve of the surface and the plane containing this directional vector and the surface normal vector at this point. Therefore, each point has different values of normal curvature, one for each direction. Among these values, the maximum and the minimum are called the *principal curvatures* and their corresponding directions are called the *principal directions*. The product of the principal curvatures is called *Gaussian curvature* and their arithmetic average is called *mean curvature*.

Recently, various approaches that use curvatures as intrinsic characteristics of surfaces and describe shape by curvature have been addressed in the literature on computer vision.[11,13] Curvature-based intrinsic features are very useful in object recognition techniques.

We compute four basic types of surface curvatures—Gaussian, mean, maximum, and minimum—for a given CAD model designed with Alpha_1. The input CAD models may be in different forms: sampled surface points, continuous B-spline surfaces, and subdivided polygons.

For *sampled surface points*, we use finite differences to approximate the first and second partial derivatives. Then we apply standard equations to find the Gaussian, mean, and principal curvatures.[8] Figure 7a exhibits samplings of surface points for one view of the Renault piece at various resolutions. Using this data, Figure 7b shows the edge points of the sampled Renault piece model, which are found by simply requiring the larger absolute value of the principal curvatures to fall above a threshold. We can observe the similarity of results on synthetic and real data. Also note that the curvature results even at low resolution are quite good. Like the edge points, planar patches can be found by using a low pass filter.

We can obtain surface curvatures from *continuous B-spline surfaces* in a simple manner. The basic surface type in the

Alpha_1 system is a tensor product B-spline patch. A convenient way to think of the tensor product surface is to think of the rows or columns of the matrix of spatial points (control mesh) as a set of individual B-spline "control curves," with one knot vector and one order associated with each of them. The other knot vector and its order then describe how these curves will be blended to form the surface. The derivative of this B-spline surface is another tensor product B-spline surface with a lower order formed by the differentiation of these control curves. Higher order derivatives are found by successive differentiations. The unit normal vector is found from the cross product of the first partial derivatives (tangent vectors). The rest of the computations of curvatures are the same as before. Since it requires rational computations in vector normalization, there is no *closed form* of surface curvatures in B-spline.

For models in the form of *subdivided polygons*, we use an alternative definition of Gaussian curvature used in EGI. The Gaussian curvature of a small polygon can be approximated by the ratio of the area of regions enclosed by normals of vertices on the Gaussian sphere to the actual area of the polygon.

Interpretation of curvature results. Figure 8 shows the results of the computation of curvatures for a Coons patch. In the two views of a Coons patch in Figure 8a, the surfaces contain four interesting parts: peak, pit, and two saddles. Figure 8b shows the four surface curvatures. In Figure 8b the black lines are isoparametric lines of curvatures (not lines of curvature) and the white lines are the zero crossings. Figure 9 shows the Gaussian curvature and the extrema of principal curvatures of a tea pot. Images in Figure 8b and 9 are generated by mapping the curvature values onto its B-spline control mesh approximately. From Figures 8 and 9 we can make the following observations:

(1) The zero crossings of Gaussian curvature do not necessarily correspond to step edges. They are just surface inflection points, a kind of critical point.

(2) Segmentation of range data based on the zero crossings of Gaussian curvature gives a meaningful decomposition of surface patches. They are clearly separated by the zero crossings of Gaussian curvature.

(3) The sign of Gaussian and mean curvatures provides a useful symbolic descrip-

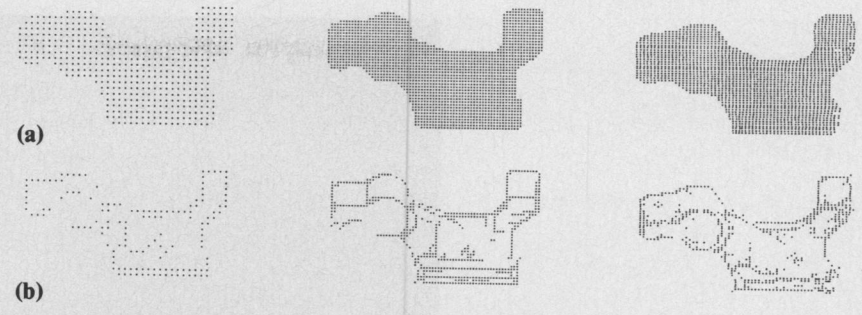
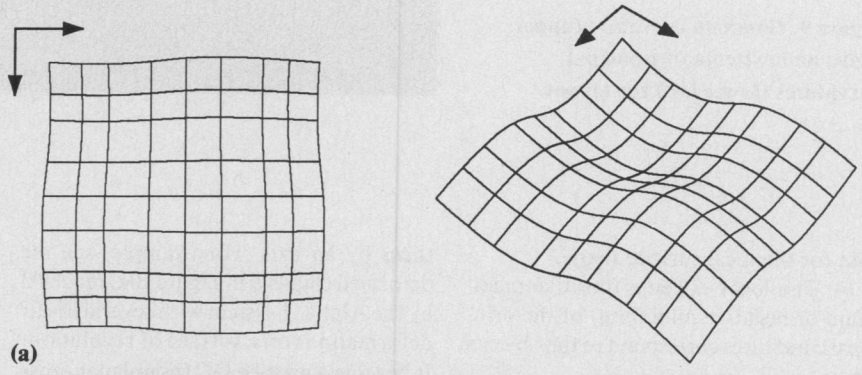


Figure 7. Surface points and the extrema of principal curvatures. (a) shows a sampling of surface points at two resolutions (0.2-inch and 0.1-inch spacing) and the real range data taken with a laser range finder (0.12-inch resolution in the *x* direction and 0.08-inch resolution in the *y* direction). (b) shows edge points as the extrema of principal curvatures for the figures in (a).



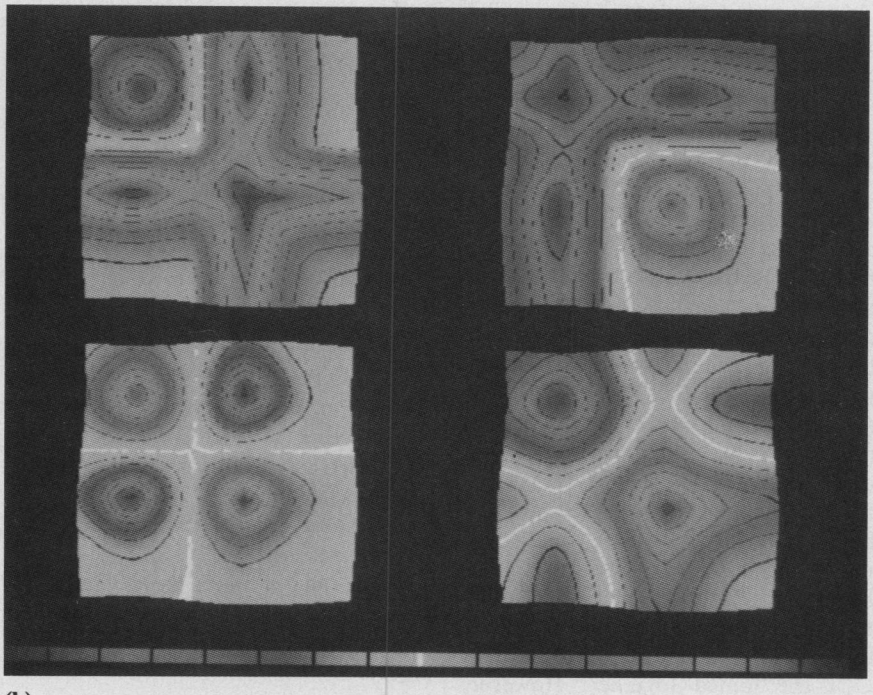


Figure 8. Four basic types of surface curvatures of a Coons patch. (a) shows two different views of a Coons patch and (b) shows surface curvatures: maximum principal curvature (upper left), minimum principal curvature (upper right), Gaussian curvature (lower left), and mean curvature (lower right).

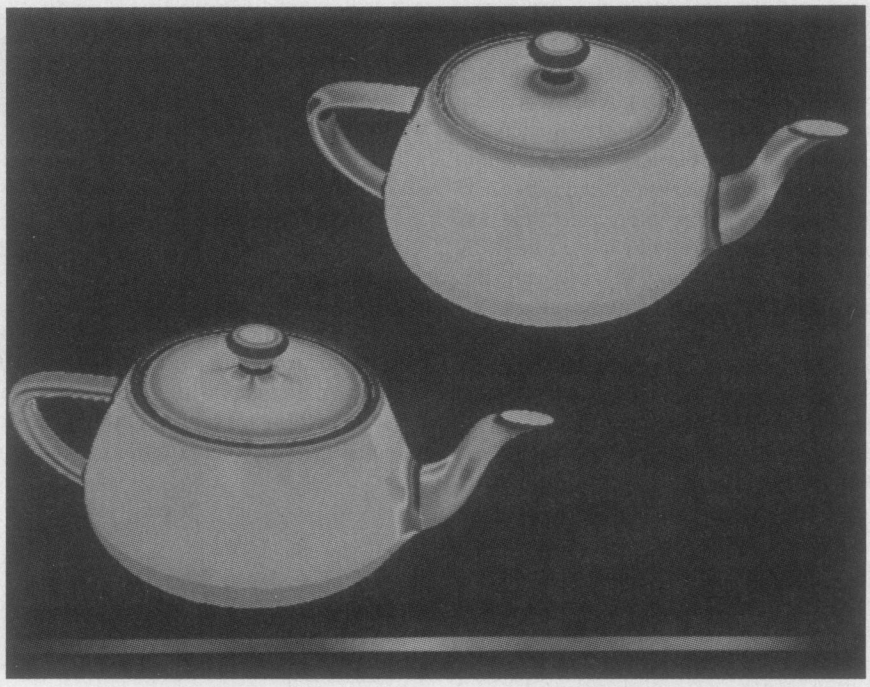


Figure 9. Gaussian curvature (upper right) and extrema of principal curvatures (lower left) for teapot.

tion for the local surface shape.

(4) The local extrema (positive maximum or negative minimum) of the principal curvatures correspond to the object's edges.

(5) The local maxima of the smaller absolute values of principal curvatures correspond to the object's corners.

(6) Conic surfaces (sphere, cylinder, and cone) and planes can be specified by the value of principal curvatures.

Generalized sweep representation. We can extract this representation directly from the CAD design procedures if the object is designed in this way. However, to design the axis, a set of cross sections and their profile functions for a given shape is not straightforward. Sometimes, without special design tools, it even becomes impossible for some fairly complex objects. One possible solution is to design the generalized sweep representation by other powerful construction operators, then extract the approximate cross sections and axis from the designed model's surfaces. For more complex objects, we can use hierarchical structures where different GCs are joined together. However, sophisticated decompositions are required in that case.

To generate GCs for simple objects, or simple subparts of a complex object, we have to find some cross sections and link them by an axis. For example, see the deformed ellipsoid in Figure 10a, modeled by the Alpha_1 system with several linear deformations on a surface of revolution. It became a warped GC (nonplanar cross sections) after these deformations. One possible way to generate a GC description for this object is:

(1) Slice it in some direction to find all the cross sections.

(2) Find the centroid of each cross section. This can be done by applying Green's theorem[8] to the curve, as long as it is closed.

(3) Link all centroids to construct the axis.

Figure 10b shows a result of the above procedure. It has the following properties:

(1) All cross-section planes are parallel to each other.

(2) All cross sections are planar but not necessarily circular.

(3) The axis passes through the centroid of each cross section.

(4) The angle between the axis and each cross section plane is not necessarily 90 degrees, and each one has a different angle.

This parallel slicing method is very simple and the result in Figure 10b looks good. However, using this technique we will have an infinite number of descriptions for a single object. For example, in Figure 10c, we used a different slicing direction and got a totally different result—a different axis and different cross sections. This kind of representation obtained from the parallel slicing method is useless in object recognition. More constraints are needed to get a unique description.

The initial slicing direction is important in order to get the canonical axis that is invariant to rotation and translation. A splitting scheme given below is similar to the one used in curve approximation.[19] It uses the axis of inertia and generates GCs such that all cross sections are closed, cross sections do not intersect each other, the axis is orthogonal to each cross section, and the axis passes through the centroid of each cross section.

Iterative splitting method for generalized cylinder approximation. This algorithm has the following steps:

(1) Find the major axis of inertia, the one having minimum moments of inertia, and the extrema of the object along this axis.

(2) Find the cross sections near these extrema that are perpendicular to the major axis of inertia and connect their centroids as the first approximation to the axis.

(3) Find the cross section that passes through the midpoint of the approximated axis and is perpendicular to it. Connect its

centroid to the two endpoints of the previous axis and split the axis and the object into two pieces.

(4) Repeat Step 3 on each of the subpieces recursively until the desired resolution is obtained, or the new cross section is not closed, i.e. it intersects other cross sections.

(5) Adjust the axis and cross sections recursively such that all cross sections are found at critical points of the axis and are perpendicular to the axis at their centroids.

This approach provides several advantages:

- It uses the axis of inertia in the initialization procedure to obtain the canonical axis of an object.
- The axis is perpendicular to the cross sections and passes through their centroid.
- The cross sections are closed planar curves and do not intersect each other.
- Since the whole surface is split during the recursion, its time complexity is improved from $O(mn)$ to $O(m\log_2 n)$ where m is the total number of polygons in the CAD model of a component and n is the number of cross sections.

The algorithm minimizes the number of possible GC representations for one object to achieve the unique property of a vision model, and does not restrict the shapes of axis and cross sections to represent a larger scope of objects. Figure 11a shows the results of the axis and cross sections extraction on the object shown in Figure 10a, after one, three, and five iterations. Figure 11b shows the axis and cross section for the helicopter shown in Figure 12b. It is a first-order approximation. A refined GC representation is obtained by including an angle test along the axis and a similarity test by moments on adjacent cross sections.

Polyhedral representation. Polyhedral representation is widely used in computer vision because of its simplicity and good support of both geometrical and topological information. A polyhedral model can be constructed from vertices, edges and faces where vertices are 3D spatial points, edges that are straight-line segments between vertices, and faces that are planar polygons enclosed by an ordered list of edges or their corresponding vertices. A set of geometrical and topological conditions, as given by Requicha,[5] must be met for any valid polyhedral model.

In this work, the geometrical conditions are assumed from the validity of the given CAD models. Most of the topological con-

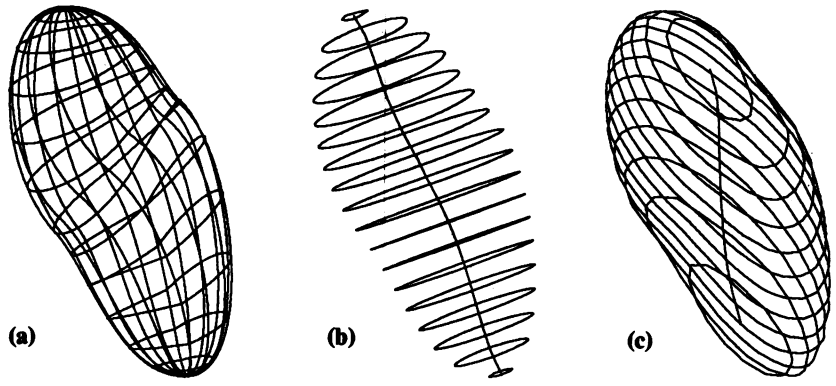


Figure 10. Generalized cylinder representation for a simple object showing (a) a deformed ellipsoid, (b) a generalized cylinder approximation of (a), and (c) another generalized cylinder approximation of (a).

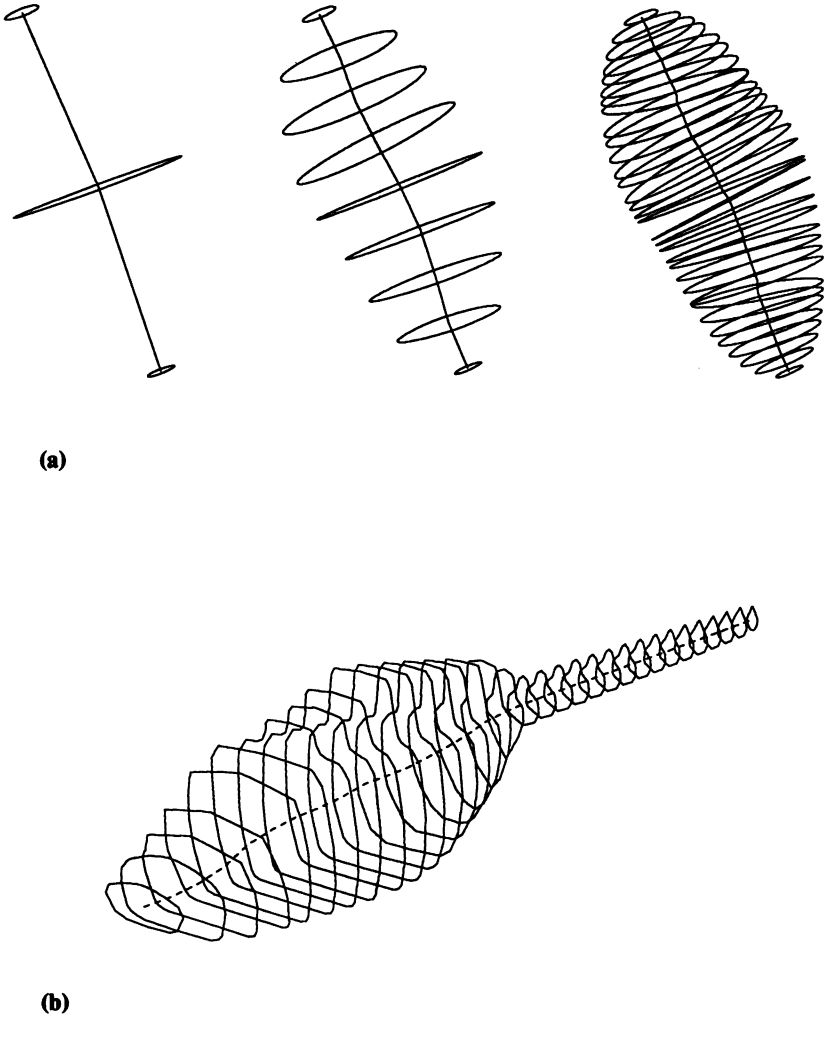


Figure 11. Generalized cylinder approximation showing (a) results of the iterative splitting algorithm after one, three, and five iterations for the object shown in Figure 10a, and (b) generalized cylinder approximation for the helicopter shown in Figure 12b.

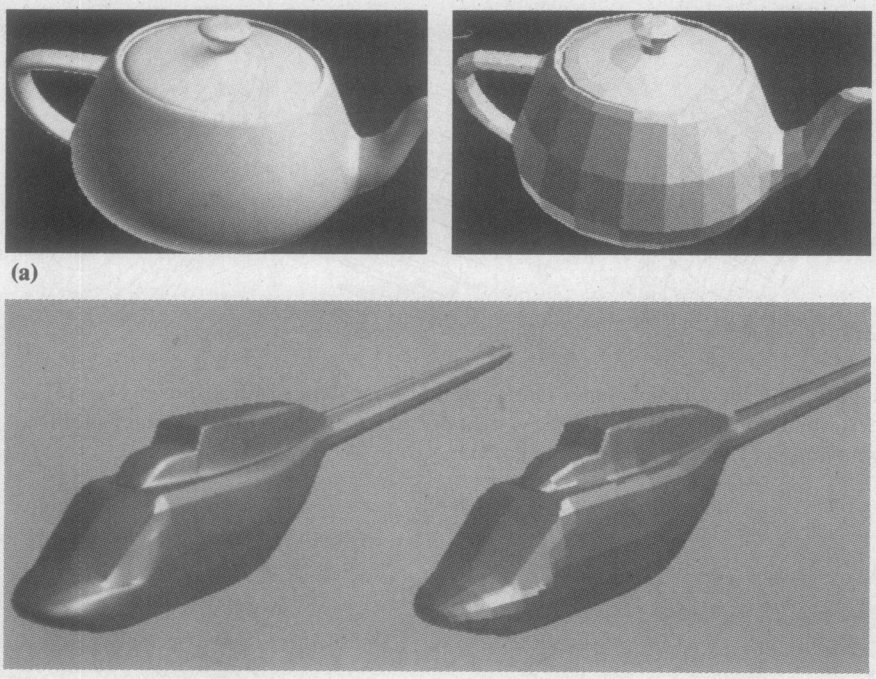


Figure 12. Polyhedral representation for a teapot and a helicopter. (a) shows a B-spline model for a teapot and its polyhedral approximation. (b) shows a B-spline model for a helicopter and its polyhedral approximation.

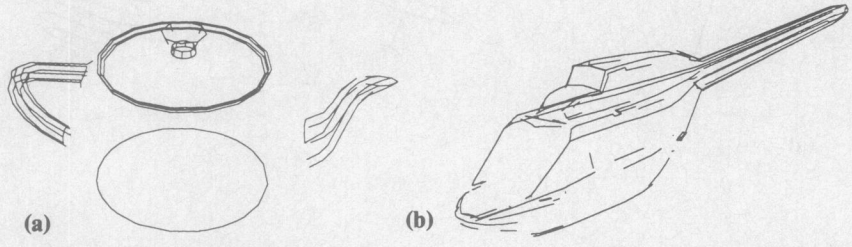


Figure 13. Extraction of features from the polyhedral representation showing (a) edge detection on a teapot and (b) edge detection on a helicopter.

ditions are also ensured by the embedded linked-list data structure. However, due to the flexibility of nonuniform B-splines used in the Alpha_1 system, different combinations of order, knot vector, and control points may result in different curves having identical geometry. For modeling of complex shapes it is very likely that adjacent patches will not have the same knot vector and control points along their common boundary, and that some patches will be adjacent to more than one patch on one side.

Currently Alpha_1 does subdivision and polygonal approximation locally, which leaves some gaps along adjacent patches because of the above situations. In order to get a valid polyhedral model such that every edge is shared by two and only two polygons, we use a global approach in this research. Adjacency information on surface patches contains not only the common sides but also the ranges in which they are matched. For partially adjacent patches and patches that are differently parameterized along the common boundary, we insert information on more than one adjacency into each side. This information is propagated proportionally to the subdivided patches whenever a subdivision occurs. In this global approach we first perform all the required subdivisions, then build polygons for each small subpatch which contain not only the subpatch's four corners but also the adjacent corners of all neighboring patches. Therefore, the adjacency information on the subpatch can be mapped onto each of the approximated polygons and still maintain the topological validity of the resulting polyhedron.

Figure 12a shows the B-spline model for a teapot and its polyhedral approximation. Figure 12b shows a similar example of a helicopter. Features can also be extracted from this polyhedral representation. For example Figure 13 shows the results of edge detection on the teapot and the helicopter model by thresholding the changes in the surface normal vector along adjacent faces.

Extended Gaussian image representation. Although there is a continuous expression[9] for EGI for some objects, such as the solid of revolution, a uniform approach that allows both smooth surfaces and polyhedral objects to have the same EGI structure first approximates smooth surfaces by polygons and then maps each polygon onto the Gaussian sphere. To gain the advantages of EGI, such as its invariant mapping under rotation, a tessellation of the Gaussian sphere should have cells such that

(1) there is no overlap and gap between cells,
(2) they have the same area,
(3) they have the same shape,
(4) they occur in a regular rounded pattern, and
(5) there exists a formal scheme to obtain finer resolution that still has the above properties.

Unfortunately, these criteria cannot be satisfied simultaneously.

A simple tessellation by divisions of meridians and parallels has a higher density of cells on both north and south poles. Although we can overcome this by having fewer strips at higher altitudes, this tessellation does not have a linear relationship of rotation between the object and its EGI mapping unless the rotational axis is vertical. Better tessellations result from projecting regular polyhedra onto a concentric unit sphere. These tessellations have proved to be the optimal sampling of

a sphere on the corresponding number of samples.

However, there are only five regular polyhedra and the maximum number of faces for an icosahedron is 20. Further subdivision on each face of these regular polyhedra is required to obtain finer resolution. A well-known method is the geodesic divisions. For a tessellation based on the icosahedron, each triangle of the icosahedron is subdivided into four equal-sized right triangles. After projecting these subdivided triangles onto a Gaussian sphere, we obtain an 80-face tessellation.

By repeating the subdivision/projection procedure, a multiple-resolution tessellation of the Gaussian sphere is constructed hierarchically. This structure consists of a set of concentric spherical shells. The outermost shell has the highest resolution of the geodesic tessellation and the innermost one is the basic icosahedron. On each face of the icosahedron is an inverted triangular pyramid that links the corresponding triangles at different resolution levels. This resembles the pyramidal image structure used in 2D computer vision and has similar properties and advantages. For example, the EGI weight of one cell is equal to the sum of the weights of its four descendents at the next level.

To construct the multiple-resolution EGI from a given CAD model, each B-spline patch is first subdivided into flat polygons within a given tolerance. The area of each of the polygons is then accumulated in the corresponding cell of each level. The procedure to access the corresponding cell in one resolution from the polygon's normal vector follows:

(1) Determine into which of the triangles of the icosahedron the given normal falls.

(2) Determine into which of its four descendents the given normal falls.

(3) Repeat Step 2 until you reach the required resolution level.

The total number of tests needed to access one cell at level n, assuming the icosahedron is level 0, is $4n + 20$ in the worst case. In fact, we find not only the cell at level n, but also all the corresponding cells from level 0 to level $n - 1$ simultaneously.

Figure 14 shows the results of the geodesic tessellation based on an icosahedron and its EGI mapping for a cylinder. Figure 14a shows the testing cylinder, whose length is twice the diameter. Its EGI mappings are also shown at each resolution level. The darker triangle has higher weight. Figure 14b is the icosahedron at level 0. Figures 14c to 14e are levels 1, 2, and 3 and have 80, 320, and 1280 faces, respectively. The orientation of the cylinder is clearly reflected on its EGIs. The two black triangles in Figure 14e are images of the top and bottom circular faces of the cylinder. The side faces of the cylinder are mapped onto a circular strip on the Gaussian sphere.

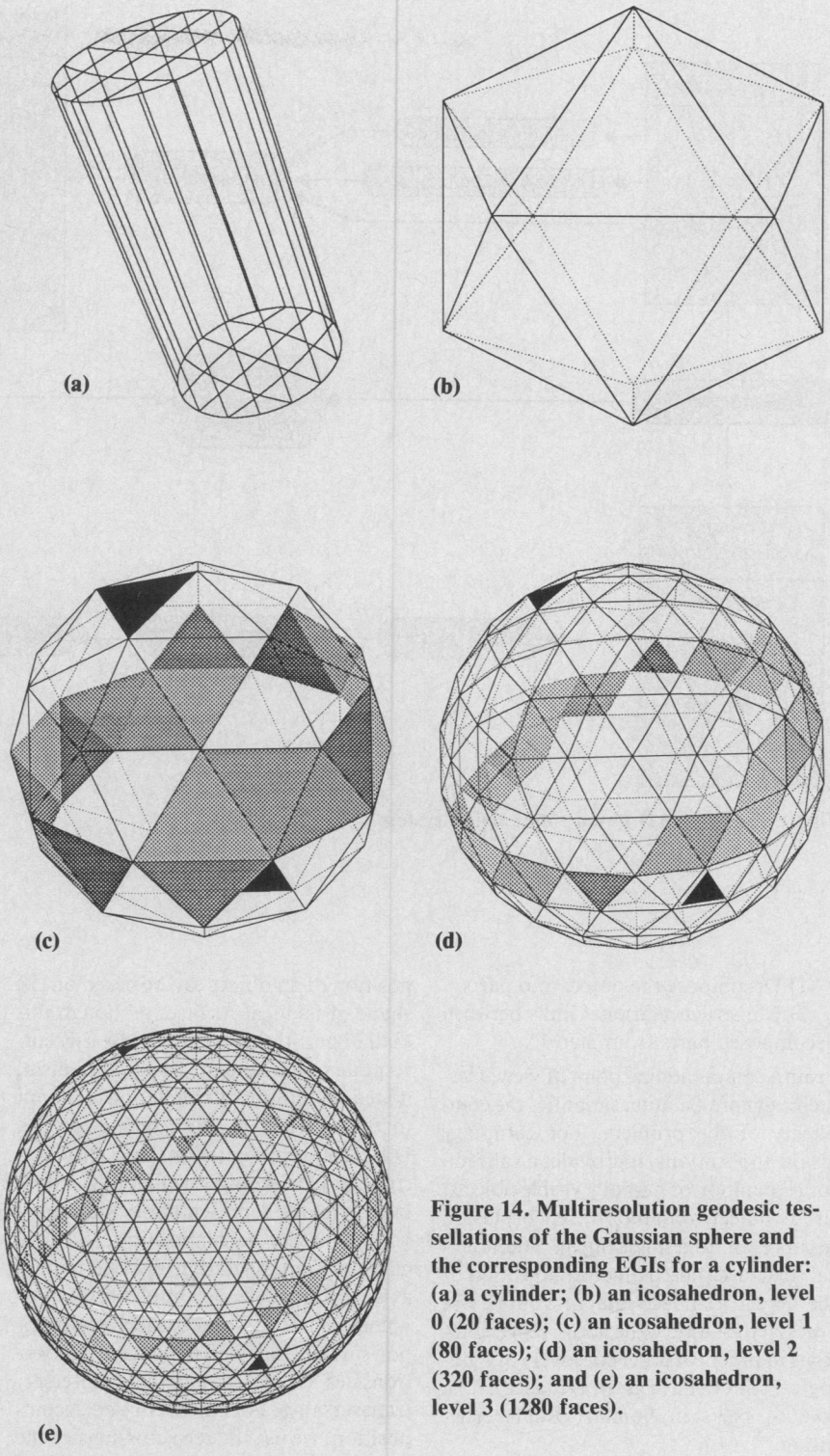


Figure 14. Multiresolution geodesic tessellations of the Gaussian sphere and the corresponding EGIs for a cylinder: (a) a cylinder; (b) an icosahedron, level 0 (20 faces); (c) an icosahedron, level 1 (80 faces); (d) an icosahedron, level 2 (320 faces); and (e) an icosahedron, level 3 (1280 faces).

For a concave object, we decompose the object's surface and build an EGI for each patch.

Object decomposition and hierarchical representation. Hierarchical representation has been commonly used in different domains.[4,20,21] An approach to this representation requires two steps:

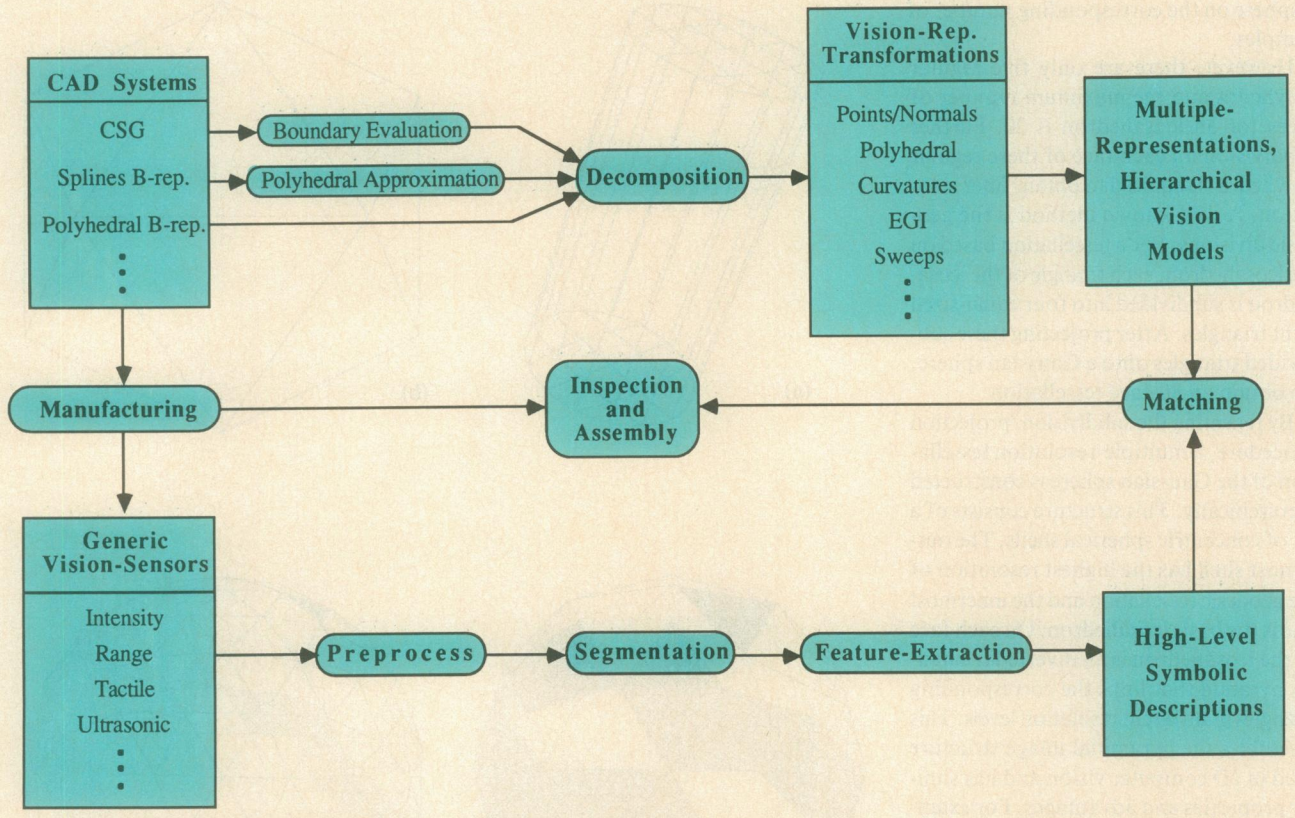


Figure 15. The CAD-based robot vision system.

(1) Decompose the object into parts.

(2) Construct relational links between decomposed parts from Step 1.

From a computational point of view, hierarchical representation simplifies the complexity of the problem. For computer vision applications, it provides a solution for recognition of partially visible objects. In 3D object recognition, self occlusion occurs even for a single object. Hierarchical structures based on decomposition of the object's surface and/or volume are necessary for any of the above representations in practical use. Moreover, psychological studies have given evidence of the role of parts in human visual recognition.[22]

Dividing objects into regular primitives (spheres, cubes, etc.) has been common in CSG systems. It is useful in CAD/CAM applications because of the analogy between set operations and mechanical manufacturing. However, this decomposition contains primitives that may not exist in the sensed data. Thus, it does not suit computer vision applications.

As described in the section "Surface curvature representation" above, decomposition of an object can be based on the shape of its local surface. Although this kind of partition does not necessarily correspond to the human visual mechanism, it is computationally simple and invariant to the viewer's position and direction. Different representations use different strategies to decompose objects. Generalized cylinder representation requires a volume-based decomposition. Surface curvature representation and EGI require a surface-based decomposition. For volume-based decomposition, we partition surfaces based on the generic intersections of surfaces (surfaces intersect transversally). For surface-based decomposition, we use the zero crossings of the Gaussian curvature and the extrema of the principal curvatures. In Figure 9 note that Gaussian curvature and extrema of principal curvatures provide good decomposition.

The CAD-based approach presented here allows the construction of vision models employing multiple representations for most of the objects found in industrial environments. It differs from using CAD tools to design features that can be visually measured. As summarized in Figure 15, the CAD-based vision model preparation procedure reveals a strong analogy to the image-understanding procedure. It needs some preprocessing of the input CAD designs. Decompositions or 3D segmentations are then performed on the model's shape, the physical surface. Finally, we extract features for different representations from each subpart and integrate them into the hierarchical multiple-representations vision models.

Our approach connects the relationship between the object's image in the real world and the sensory data and its image in the designer's mind, the CAD models. It also provides an automatic and systematic approach to building models using multiple representations on different parts of the same object. These multiple representations and the multiple matching techniques based on these representations are required in a flexible automated environment, where robots equipped with multiple sensors operate. □

Acknowledgments

This work was supported in part by National Science Foundation grants DCR-8506393, DMC-8502115, ECS-8307483, and MCS-8221750. We would like to thank the Alpha_1 group at the University of Utah for their cooperation on this project.

References

1. B. Bhanu and T.C. Henderson, "CAGD-Based 3D Vision," *Proc. IEEE Int'l Conf. Robotics and Automation*, Mar. 1985, pp. 411-417.
2. Alpha_1 Research Group, *Alpha_1 Users Manual*, Dept. Computer Science, University of Utah, Jan. 1986.
3. R.T. Chin and C.R. Dyer, "Model-Based Recognition in Robot Vision," *Computing Surveys*, Mar. 1986, pp. 67-108.
4. D. Marr, *Vision*, W.H. Freeman and Co., New York, 1982.
5. A.A.G. Requicha, "Representation for Rigid Solids: Theory, Methods, and Systems," *ACM Computing Surveys*, Dec. 1980, pp. 437-464.
6. T.O. Binford, "Visual Perception by Computer," *Proc. IEEE Conf. Systems and Control*, Dec. 1971.
7. S.A. Shafer, *Shadows and Silhouettes in Computer Vision*, Kluwer Academic Publishers, 1985.
8. I.D. Faux and M.J. Pratt, *Computational Geometry for Design and Manufacture*, John Wiley & Sons, New York, 1979.
9. B.K.P. Horn, "Extended Gaussian Images," *Proc. IEEE*, Dec. 1984, pp. 1671-1686.
10. K. Ikeuchi, "Generating an Interpretation Tree from a CAD Model to Represent Object Configurations for Bin-Picking Tasks," Tech. Report CMU-CS-86-144, Dept. Computer Science, Carnegie Mellon Univ., Aug. 1986.
11. P.J. Besl and R.C. Jain, "Invariant Surface Characteristics for 3D Object Recognition in Range Images," *Computer Vision, Graphics, and Image Processing*, Jan. 1986, pp. 33-80.
12. E. Cohen, T. Lyche, and R.F. Riesenfeld, "Discrete B-splines and Subdivision Techniques in Computer-Aided Geometric Design and Computer Graphics," *Computer Graphics and Image Processing*, Oct. 1980, pp. 87-111.
13. T.C. Henderson and B. Bhanu, "Intrinsic Characteristics as the Interface Between CAD and Machine Vision Systems," *Pattern Recognition Letters*, Vol. 3, 1985, pp. 425-430.
14. B. Bhanu, "Representation and Shape Matching of 3D Objects," *IEEE Trans. Pattern Analysis and Machine Intelligence*, May 1984, pp. 340-351.
15. B. Bhanu et al., "Range Data Processing: Representation of Surfaces by Edges," *Proc. 8th Int'l Conf. Pattern Recognition*, Oct. 1986, pp. 236-238.
16. K.T. Gunnarsson and F.B. Prinz, "CAD Model-Based Localization of Parts in Manufacturing," *Computer*, Aug. 1987, this issue.
17. B. Bhanu, C.C. Ho, and T. Henderson, "3D Model Building for Computer Vision," *Pattern Recognition Letters*, May 1987, pp. 349-356.
18. T. Pavlidis, *Algorithms for Graphics and Image Processing*, Computer Science Press, 1982.
19. D.H. Ballard and C.M. Brown, *Computer Vision*, Prentice Hall, New York, 1982.
20. R. Nevatia and T.O. Binford, "Description and Recognition of Curved Objects," *Artificial Intelligence*, Vol. 8, 1977, pp. 77-98.
21. T. Phillips, R. Cannon, and A. Rosenfield, "Decomposition and Approximation of Three-Dimensional Solids," *Computer Vision, Graphics, and Image Processing*, Mar. 1986, pp. 307-317.
22. I. Biederman, "Human Image Understanding: Recent Research and a Theory," *Computer Vision, Graphics, and Image Processing*, Vol. 32, 1985, pp. 29-73.

Bir Bhanu is the guest editor of this special issue on CAD-based robot vision. His photo and biography appear following the Guest Editor's Introduction in this issue.

Chih-Cheng Ho is a graduate student in the Department of Computer Science at the University of Utah. His research interests include computer vision, computer-aided geometric design, and computer graphics. He is also interested in Unix/C programming and small computer systems.

Ho received his MS degree in computer science from the University of Utah and BS degree in engineering from National Taiwan University.