COMPUTER VISION, GRAPHICS, AND IMAGE PROCESSING **36**, 229–255 (1986)

Determination of the Identity, Position and Orientation of the Topmost Object in a Pile*

H. S. YANG AND A. C. KAK

Robot Vision Lab, School of Electrical Engineering, Purdue University, West Lafayette, Indiana 47907

Received April 1, 1986; accepted July 7, 1986

In this paper, we first propose schemes for segmenting out the visible part of the topmost object from a pile of planar and curved objects. We then describe our work on using *B*-splines for the characterization of the topmost object surface when it is curved; the *B*-splines are used for deriving operators that yield the Gaussian and mean curvatures. This is followed by a description of our identification strategies which depend upon whether the topmost object is judged to be planar or curved. The identification strategy for planar objects revolves around the EGI representation of their visible surfaces and is a function of whether the number of visible planar surfaces is one, two, three, or more. In case the number of sufficiently visible planar surfaces is less than or equal to two, we have incorporated surface boundary information with angular relation between adjoining surfaces to improve the identification process. For curved objects, the identification strategy depends upon the signs of the Gaussian and the mean curvatures and the EGI. While these identification strategies are not guaranteed to work in every case, we expect them to be practically useful for a wide range of industrial objects.

1. INTRODUCTION

Seemingly simple operations like picking up the topmost object from a pile of objects can be exceedingly difficult for even a robot that is endowed with sensory capabilities for the purpose of dealing with a random environment. For many years, it was hoped by the computer vision community that we would succeed in implementing on a computer the laws of Gestalt organization and stereo perception that allow us humans to perform such tasks so effortlessly. More recently, the attitude has been that while research must continue in simulating these human abilities, which appear to be guided by knowledge and driven by expectation, we must in the meantime also look for purely engineering solutions to the sensory feedback problems of robots.

During the past few years, many engineering solutions have been shown to be feasible. For example, if the aim is to simply pick objects from a bin and if the object surfaces are smooth it is possible to use a vacuum gripper [14]. If the aim is a bit more sophisticated, such as sorting a pile of objects on the basis of shape, one must now use the sensory data (in most cases, vision) to not only locate the least occluded object—usually topmost—but also recognize its shape in two or three dimensions; in addition, it may be necessary to compute optimum holdsites.

With vision sensing, because of difficulties with the segmentation and grouping of photometric information, it is now generally believed that when the objects involved are inherently 3D in nature (as opposed to being flattish), one must resort to range

*This work was supported by the National Science Foundation and Purdue ERC (Engineering Research Center for Intelligent Manufacturing).

maps for scene analysis. Of course, the ideal would be to use the human-like stereo vision capability for generating the range maps with two or more cameras [13]; but, in practice, for reliable and sufficiently dense data, a more engineering solution must be used—which is either laser based or structured-light based.

The main purpose of this paper is to describe our strategies for analyzing structured-light range maps for determining the identity, position, and orientation of the topmost object in a pile. The extent to which the topmost object must be free of occlusion for our strategies to work depends, in general, upon how many different types of objects there are in the pile. Currently, if the topmost object lacks sufficient visibility and/or cannot be matched with any of the known descriptions, it will be declared unidentifiable. In a future system, at this point the robot would presumably disturb the pile and the scene would be reëxamined.

The problem of locating and identifying the topmost object in a pile could be considered to be a specialized case of scene analysis with 3D vision. Limiting our citations to those that deal specifically with multi-object scenes: Oshima and Shirai [16] have segmented scenes consisting of polyhedral and curved objects by using a region growing technique; the overall scene was then described in terms of properties of regions and relations between them. Using information generated by photometric stereo, Horn, and Ikeuchi [10] have used the *extended Gaussian image* to determine the identity and orientation of an object that is a part of a small pile. Bolles [4] has used structured light range data to quickly and reliably locate cylinders of a specified diameter in a pile of cylinders. We have described in [5] a procedure for pile analysis that correctly segments a structured light generated edge-vertex description of a scene consisting of convex polyhedral objects. Dessimoz *et al.* [7] have used a matched filter type of implementation in which one first takes note of the geometry and the mechanics of how the robot end-effector picks up objects; this consideration then leads to a set of features that can be invoked to identify the object if it is successfully grasped by the robot.

The approach presented in this paper first extracts local surface definitions from structured-light range maps. In the vicinity of the topmost point of the scene, as discussed in Section 2, surface definition is tested for planarity; subsequent processing depends upon the outcome of this test. Depending upon whether the topmost object is planar or curved, we invoke different heuristics for the segmentation of its visible patch; these heuristics are described in Sections 3 and 4. After the visible part of a curved object is isolated, special operators must be used to estimate its characteristics—of course, we want these characteristics to possess the property of visible-invariance [3]. In Section 5, we have shown how B-splines can be used for computing curvature properties, which possess this feature of visible-invariance, for curved surfaces. As discussed in Sections 6 and 7, depending upon whether the topmost object is judged to be planar or curved, different procedures are used for object identification—to the extent such identification is possible from the characteristics of the visible patch.

2. TESTING THE TOPMOST VISIBLE SURFACE FOR PLANARITY

Before we can test the topmost surface for planarity, it must first be located. Fortunately, with 3D vision data that is trivially accomplished by seeking out the pixel with the largest z coordinate, which corresponds to the maximum height above the work table. (If there exist many pixels whose heights are maximal and identical —this can happen when a straight edge of the topmost object is positioned parallel to the work table—it is possible to choose any one of them as the topmost pixel.) An $N \times N$ set of pixels containing the topmost pixel is then defined as the topmost patch; and a 3×3 set of patches, containing the topmost patch at the center, is then tested for the planarity of the topmost surface. The size of the patches ($N \times N$) is heuristically determined; if it is too small, the planarity test may not be reliable, and if it is too large, the implicit assumption that all continuous surfaces, even curved ones, are locally plane will be violated. We have used $N = 8$.[1]

The planarity test measures the variation in surface normals over this 3×3 set of patches; each surface normal being calculated by performing a least-mean-squares fit of a plane to the corresponding 8×8 cluster of pixels. To give greater credibility to the planarity test, we can also examine the Gaussian and mean curvatures, since over regions that are declared to be planar, both should be zero. (For details about the estimation of local Gaussian and mean curvatures, the reader is referred to Sect. 5.) If with these tests it is determined that the surface region under examination is indeed planar, the object (or, at least its topmost surface) is declared planar; otherwise, it is assumed to be curved. In some cases it is possible that the 8×8 patch containing the topmost pixel might straddle an edge; clearly we do not wish such a patch to contribute to planarity decisions. These can be discarded from further consideration by putting a threshold on the error between the fitted plane and the pixel positions, as was done first in [16].

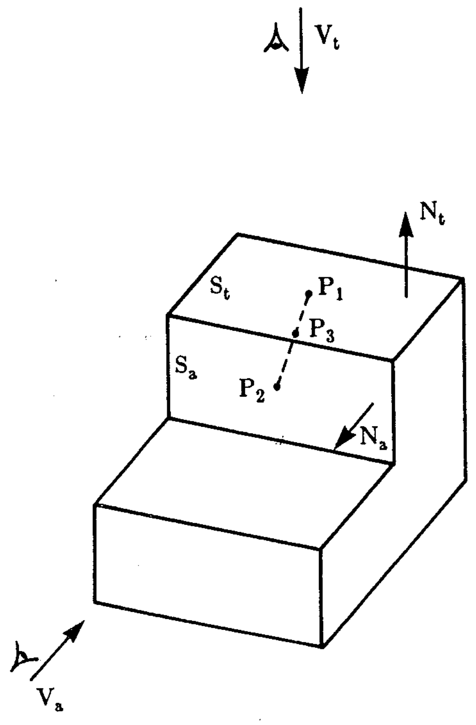
3. SEGMENTATION OF THE VISIBLE PART OF A TOPMOST PLANAR OBJECT

If the topmost visible surface is declared planar by the preceding procedure, the rest of the surface is extracted by a region growing algorithm utilizing the simple heuristic that all the surface normals of the patches of the topmost surface must point in essentially the same direction (surface normal constraint) and that the patches be adjacent (adjacency constraint). A region is allowed to grow until either there is nothing more to merge, or until we have exceeded some specified distance from the starting patch; this distance reflecting our knowledge of the maximum expected size of the objects in the scene.

The process of merging can be made somewhat faster if at the beginning we also estimate the direction toward which most of the topmost surface lies from the vantage point of the starting patch. For example, if we knew that the starting patch was at the top right corner of the surface, then the growing process could be confined to the left of and below the starting patch. Determination of where the starting patch lies in relation to the rest of surface can in some cases be made by examining a neighborhood set of patches.

After the topmost surface is extracted, the rest of the visible portion of the topmost object must be segmented out; this means extracting the visible portions of adjoining surfaces. In a sense, this is a continuation of the region growing process

[1]All processing is done over an array of numbers each column of which corresponds to range values along one illuminated light stripe in the scene. If the reader wants to know how much of an object surface area corresponds to an 8×8 patch in our illustrations, one of the linear dimensions of this area is equal to the length spanned by eight light stripes. The other dimension depends upon the sampling rate along a stripe; in our case that corresponds to the raster lines generated by the camera, which was 512.



S_t:	topmost surface
S_a:	adjoining surface
N_t:	surface normal of S_t
N_a:	surface normal of S_a

FIG. 1. Illustrated here is a concave polyhedral object in which the topmost surface (S_t) and its adjoining surface (S_a) satisfy the convexity constraint.

mentioned above, and a similar set of heuristics is now called for; the ones that we use are:

1. *Adjacency constraint.* To act as a starting patch for an adjoint surface, a patch must be a neighbor of one of the patches of the extracted topmost surface.

2. *Surface normal constraints.* Patches on an adjoining surface must have the same surface normals as the starting patch for that surface. In some cases, based on the knowledge available about the objects, constraints may be imposed on the permissible angles for the patch surface normals on adjoining surfaces. For example, if the objects are known to be rectangular, we may not accept a patch as a seed for an adjoining surface unless its surface normal is perpendicular to the normal associated with the topmost surface.

3. *Object convexity constraint.* All surfaces that adjoin the topmost surface must also satisfy the convexity condition. Which is the same as saying that points that lie on a straight line connecting any location on the topmost surface and a point in the adjoining surface must all lie behind both surfaces when those points are viewed along the direction which is the inverse of surface normal of either the topmost surface or its adjoining surface. As shown in Fig. 1, P_3 which is on a line connecting P_1 and P_2 is behind S_t when it is viewed along V_t, which is the inverse of the surface normal of the topmost surface; and behind S_a when it is viewed along V_a. The object convexity constraint obviously limits us to objects that are convex. For most

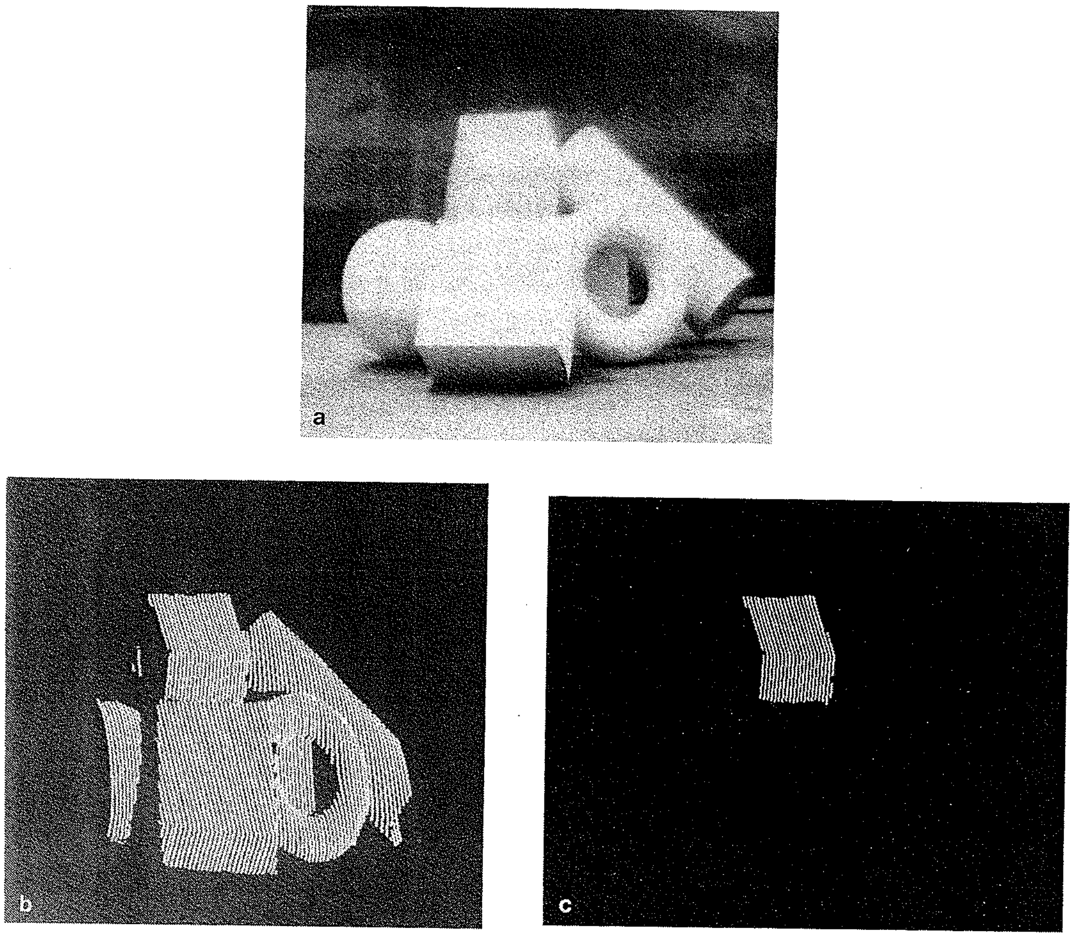


FIG. 2. (a) Another scene consisting of a pile of objects with a cube at the top. (b) Light stripe image of the scene. (c) Illustrated here is the visible part of the cube as extracted by the segmentation algorithm.

complex planar objects, via this constraint we will only be able to extract the convex part of what is visible for the topmost object.

Figure 2a illustrates a scene consisting of a pile of objects with a cube at the top; Fig. 2b shows the light stripe image of the scene, and Fig. 2c shows the topmost planar object (a cube) segmented out by the region growing procedure.

The reader might notice a certain lack of registration between (a) and (b) in Fig. 2—in the sense that the relative orientations of some of the edges in the two displays are not the same. The reason for that is experimental: the reflectance image of (a) is taken from a particular point-of-view, which bears no relationship with the point-of-view in (b). In fact, it is not even possible to associate a single point-of-view with the stripe image in (b) because it is taken with a robot-mounted scanner with both the light projector and the camera undergoing linear motions during the scan. The reader is referred to [19] for further details on this approach to structured-light 3D vision.

4. SEGMENTATION OF THE VISIBLE PART OF A TOPMOST CURVED OBJECT

Region growing is not the best strategy for extracting the visible portion of a topmost curved object; the difficulty being caused by the variations in the surface normals over such surfaces.

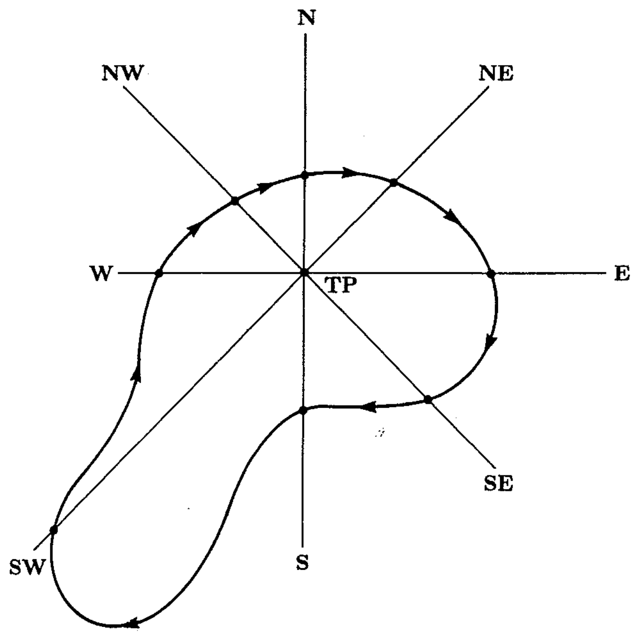


FIG. 3. TP is the topmost point. The continuous line with arrows represents the outer boundary of a hypothetical topmost curved object. The posts, on the eight cardinal directions from the topmost point, used for boundary tracking are shown also.

A better strategy for curved objects is to locate the outer boundary of the topmost curved surface by finding the defining range discontinuities. This is accomplished in the following manner. From the topmost point, we sequentially proceed outwards in eight directions, E, W, N, S, NE, NW, SE, SW, until along each direction we reach a point of a significant range discontinuity (Fig. 3); this pursuit being abandoned if a discontinuity is not found within a certain distance, which depends upon our a priori knowledge of the maximum dimension of the objects in the scene. After the range discontinuity points are detected in the eight or fewer directions, we select one of those as a starting point for outer boundary tracking. The rest are designated as "posts" on which the extracted outer boundary must "hang." Suppose, after tracking out from the starting point for a certain predetermined distance, a post is not encountered, we abandon that track, and select one of the other posts as a starting point. Tracking proceeds from post to post. After one complete attempt at going around all the posts in one direction, we go back to those post pairs where tracking proved unsuccessful; for these the tracking is then attempted in the opposite direction. If that also fails, we either connect the two posts by a straight line; or if greater precision is demanded, we might set up another post between the offending pair and retry the procedure.

However, for an object like a torus, whose range map from many viewpoints contains a hole inside the outer boundary, some posts may not reside on the outer boundary at all. As shown in Fig. 4, for the viewpoint shown there, post 5 is located on the inner boundary of the torus, therefore the tracking process from post 4 to post 5, and from post 5 to post 6, will fail even if we subdivide the sectors between post pairs. As a result, we could lose almost half of the topmost object surface if we simply connected posts 4 and 6. To overcome this flaw, whenever there exist post

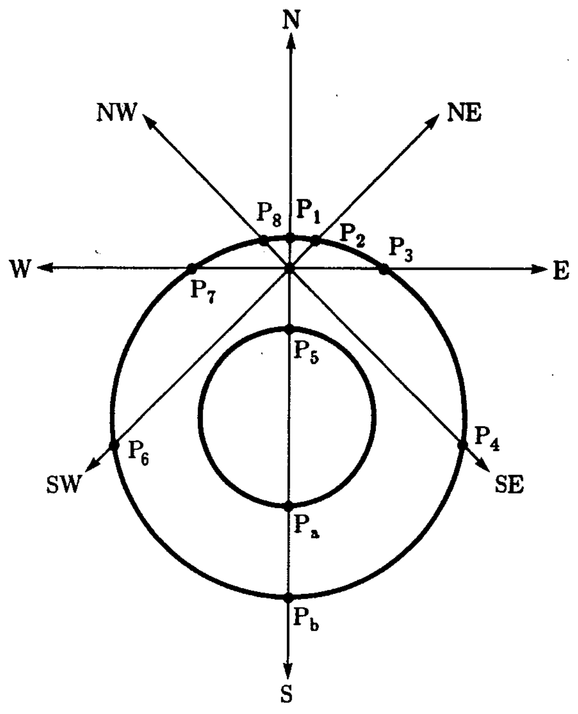


FIG. 4. An example of a torus, where the first range discontinuity along the south post, P_5, does not reside on the outer boundary of the torus. If points around P_5 have negative Gaussian curvature, the third range discontinuity to the south, located at P_b, is taken to be on the outer boundary of the torus.

pairs between which tracking fails, one must meticulously examine regions around those posts to find out if there really exists a hole there, implying a torus like object. This examination can be carried out by checking the polarity of the Gaussian curvature of the points around those posts. (Negative Gaussian curvature indicates that the region around the point is locally saddle shaped.) If it is verified that a post, such as P_5, is located on the rim of a hole, we must then try to locate the third range discontinuity along that direction (P_b in Fig. 4) and use it as a new post for the purpose of further tracking.

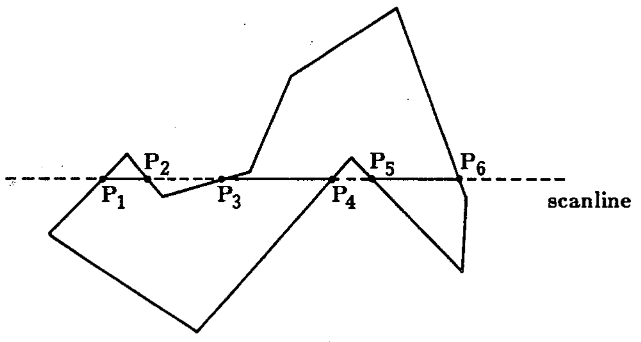


FIG. 5. Illustrated here are the intersection points of a scan line with the object boundary approximated by a polygon. Points that lie on the chords between P_1 and P_2, P_3 and P_4, P_5 and P_6 belong to the interior of the boundary.

After the outer boundary of the topmost curved object is extracted, we are faced with the not so simple task of extracting the interior points.

This is accomplished by first extracting a minimum bounding rectangle containing the outer boundary points; although this operation is trivial, it does reduce the amount of data for further processing. To extract from this rectangle those points that are within the object boundary, we have devised a procedure which uses the polygon filling technique of computer graphics.

The procedure is based on the principle that when a straight line intersects a closed boundary at, say, $P1, P2, P3, \ldots, Pn$, then all the points that lie on a chord between $P1$ and $P2$ must belong to the interior of the boundary; those that are on a chord joining $P2$ and $P3$ must be exterior to the boundary; those that are on a chord joining $P3$ and $P4$ must again be interior to the boundary; and so on (Fig. 5).

The determination of the intersections of the scan lines with the boundary is aided by the fact that the computer representation of the boundary is that of a polygon. The points of intersection of a scan line with each side of the polygon can be found by the following simple routine.

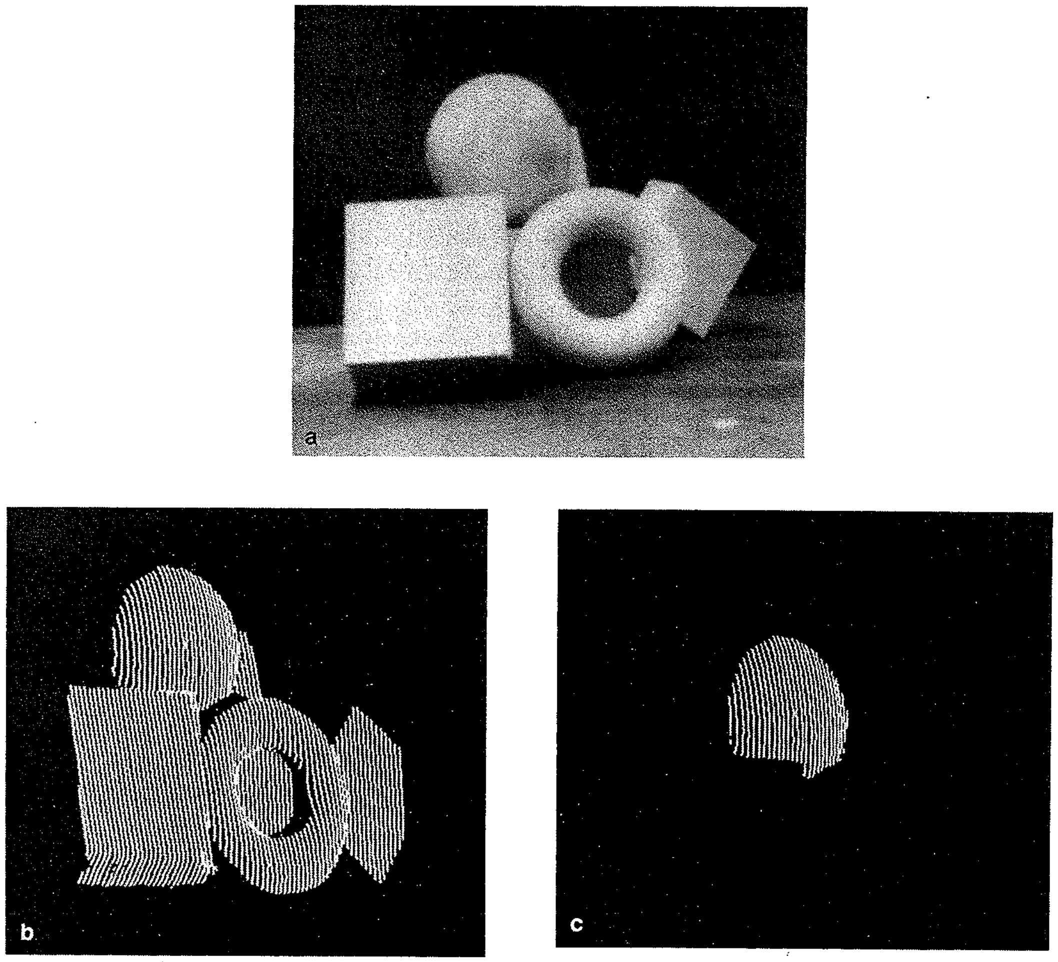


FIG. 6. (a) A scene consisting of pile of objects with a sphere at the top. (b) Light stripe image of this scene. (c) Shown here is the visible part of the sphere as extracted by the segmentation algorithm.

Let (I_1, J_1), and (I_2, J_2) be the two vertices of a particular side of the polygon. We then calculate

$$r = \frac{I_{\text{scan}} - I_1}{I_2 - I_1}. \tag{1}$$

If $r < 0.0$ or $r > 1.0$, then the scan line does not touch the line segment; if $r = 0.0$ or 1.0, then the scanline intersects the start or ending vertices of the line segment, respectively. On the other hand, if $0.0 < r < 1.0$, then the scanline intersects the interior of the line segment and the coordinates of the intersection point are given by

$$\begin{aligned} I_i &= I_{\text{scan}} \\ J_i &= J_1 + r * (J_2 - J_1). \end{aligned} \tag{2}$$

For each scanline, the intersection points with all sides of the polygon are collected, sorted from left to right in the order of increasing horizontal coordinate; and then the points between each pair are considered as the interior points of object region provided, of course, the range data is available at such points.

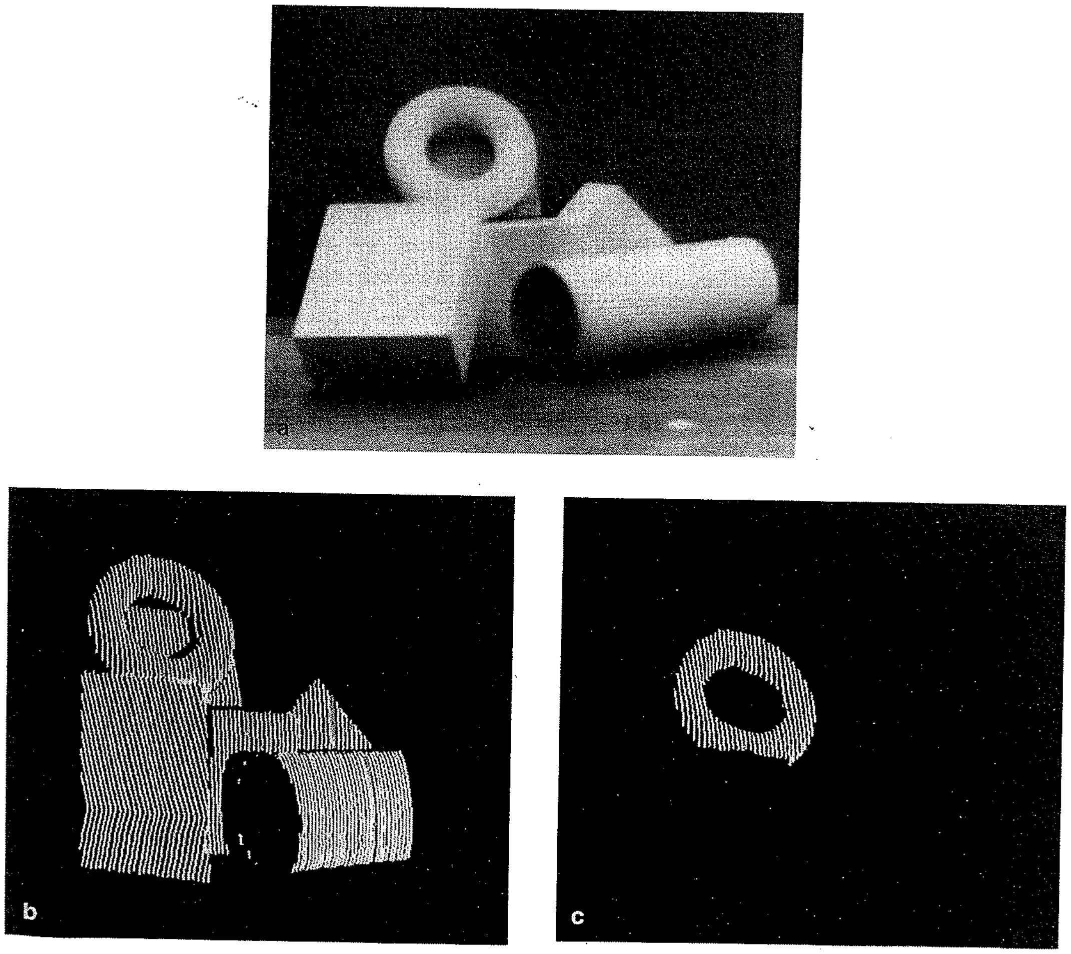


FIG. 7. (a) A scene consisting of a pile of objects with a torus at the top. (b) Light stripe image of this scene. Stripes reflected from the box beneath the torus are visible through the hole. (c) Visible part of the torus as extracted by the segmentation algorithm.

Figure 6a illustrates a piled scene in which a sphere is located at the top of the heap; Fig. 6b depicts a light stripe image of Fig. 6a; Fig. 6c describes the visible part of the topmost object as segmented by the procedure described here. The comments made at the end of Section 3 also apply here with regard to any perceived differences in registration between (a) and (b) in Fig. 6.

This algorithm can run into trouble with torus-like objects when other objects are showing through the hole, since in such cases not all the visible points within the outer boundary belong to the object. Fig. 7a shows a scene consisting of a pile of objects with a torus at the top; Fig. 7b shows its light stripe image. As is evident from the latter image, light stripes reflected from a box beneath the torus are visible through the hole. The algorithm described above for segmenting out a curved object would include this "false" region as a part of the torus. However, it is possible to eliminate such extraneous regions by using the range discontinuity information; a successful result is shown in (c).

5. SURFACE CURVATURE ESTIMATION WITH *B*-SPLINES

After the visible surface of the topmost curved object is isolated, we need to represent its shape by a set of descriptors, which could then be used for object identification and orientation determination, the reliability of such computations being determined by the extent of the visible surface. As Besl and Jain [3] have eloquently pointed out, the mean curvature and the Gaussian curvature are the local second-order surface characteristics that possess several desirable invariance properties and can be used to classify surface shapes. As long as a surface region is visible, these two curvatures are invariant to changes in surface parameterization and to translations and rotations of object surfaces. A most noteworthy feature of these two curvatures is that their signs can be used to classify a surface region into one of eight basic types; these being *flat*, *peak*, *pit*, *minimal*, *ridge*, *saddle-ridge*, *valley*, and *saddle-valley*. The reader is referred to [3] for a pictorial depiction of the elemental shapes.

In what follows, we will first very briefly present the expressions for the first and the second fundamental forms that uniquely characterize and quantify a general smooth surface and then, using the coefficients in these forms, we will show how the Gaussian and mean curvatures are related to the first and the second partial derivatives of a range map. Although all these expressions can be found in [3] and almost any book on differential geometry [8, 15] we show them here again for two reasons: they give a sense of completeness to the presentation in this section and because they provide us with a framework for introducing *B*-splines that we use for deriving operators for curvature estimation.

Note that an explicit parametric representation of a surface in E^3 is described as

$$\mathbf{x}(u, v) = (x(u, v), y(u, v), z(u, v)). \tag{3}$$

where $\mathbf{x}$ is a vector from the origin of E^3 to a point on the object surface. The first fundamental form is given by

$$\begin{aligned} \mathrm{I}(du, dv) &= d\mathbf{x} \cdot d\mathbf{x} \\ &= (\mathbf{x}_u\, du + \mathbf{x}_v\, dv) \cdot (\mathbf{x}_u\, du + \mathbf{x}_v\, dv) \\ &= E\, du^2 + 2F\, du\, dv + G\, dv^2 \end{aligned} \tag{4}$$

where $\mathbf{x}_u$ and $\mathbf{x}_v$ are partial derivatives of $\mathbf{x}$ with respect to the parametrization variables, u and v, and where

$$\begin{aligned} E &= \mathbf{x}_u \cdot \mathbf{x}_u \\ F &= \mathbf{x}_u \cdot \mathbf{x}_v \\ G &= \mathbf{x}_v \cdot \mathbf{x}_v. \end{aligned} \tag{5}$$

E, F, and G are called the first fundamental coefficients. The utility of the first fundamental form should be evident from the definition $\mathbf{I}(du, dv) = d\mathbf{x} \cdot d\mathbf{x}$, that is it allows one to compute simple metrics on the surface, such as the lengths of curves and areas of regions.

While the first fundamental form does not contain partial derivatives of $\mathbf{x}$ higher than the first, the second fundamental form depends upon the second-order partial derivatives of the surface and is given by

$$\begin{aligned} \mathbf{II}(du, dv) &= -d\mathbf{x} \cdot d\mathbf{N} \\ &= -(\mathbf{x}_u\, du + \mathbf{x}_v\, dv) \cdot (\mathbf{N}_u\, du + \mathbf{N}_v\, dv) \\ &= L\, du^2 + 2M\, du\, dv + N\, dv^2 \end{aligned} \tag{6}$$

where

$$\begin{aligned} L &= \mathbf{x}_{uu} \cdot \mathbf{N} \\ M &= \mathbf{x}_{uv} \cdot \mathbf{N} \\ N &= \mathbf{x}_{vv} \cdot \mathbf{N} \end{aligned} \tag{7}$$

with

$$\mathbf{N} = \frac{\mathbf{x}_u \times \mathbf{x}_v}{|\mathbf{x}_u \times \mathbf{x}_v|} \tag{8}$$

being the surface normal at position $\mathbf{x}(u, v)$ on the object surface. L, M, and N are called the second fundamental coefficients.

It is clear that the functions E, F, G, L, M, and N completely determine the two fundamental forms. We will now present formulas for the Gaussian and the mean curvatures in terms of these functions. The Gaussian curvature, denoted by K, is given by

$$K = \frac{LN - M^2}{EG - F^2} \tag{9}$$

and the mean curvature, denoted by H, is given by

$$H = \frac{EN + GL - 2FM}{2(EG - F^2)}. \tag{10}$$

The reason why H is called the mean curvature is that in terms of the two principal curvatures, k_1 and k_2 at a point, H is given by

$$H = \frac{k_1 + k_2}{2}. \tag{11}$$

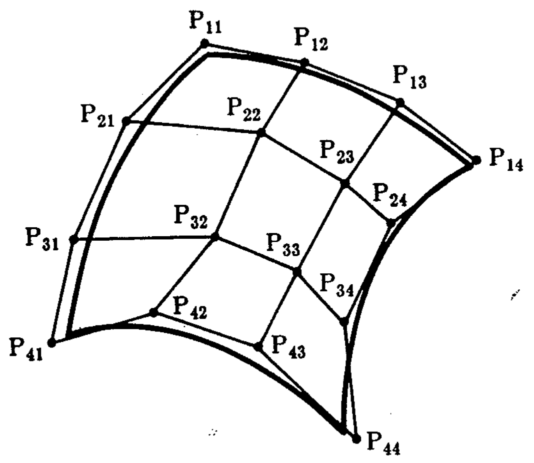


FIG. 8. Depicted here is a *B*-spline surface fitted to the 16 control vertices.

To also show the dependence of the Gaussian curvature on the principal curvatures, it is given by

$$K = k_1 k_2. \tag{12}$$

So, while H is the arithmetic mean of the principal curvatures, K is the square of the geometric mean.

It is clear that the numerical computation of K and H requires that we first calculate the first, second, and mixed derivatives of $\mathbf{x}(u, v)$.

Assuming $\mathbf{x}(u, v)$ to be of form $(u, v, z(u, v))$, a form that corresponds to what is known as a graph surface (which is also known as a Monge patch), Besl and Jain [3] have used a procedure, originally proposed by Beaudet [2] for the computation of the required derivatives of $z(u, v)$ by first fitting a quadratic function to the range data and then computing analytically the derivatives of this fitted surface.[2] They showed how all these computations can be combined into the form of a separable window convolution operation that possesses a fast implementation.

The approach used here will consist of fitting *B*-splines to $\mathbf{x}(u, v)$; we will now assume that the arguments u and v are discrete and that $\mathbf{x}(u, v)$ corresponds to an experimentally recorded range map, with the range values $\mathbf{x}$ recorded for a discrete set of (u, v). The required partial derivatives can then be obtained analytically.

For the 1-dimensional case, it is known that a parametric form must have at least a cubic dependence on the parameter in order to guarantee the continuity of positions and slopes at points where curve segments meet. There exist many ways to define such cubic forms; examples include Hermite, Bezier, *B*-spline, β-spline polynomials. However, among all these the *B*-spline form usually yields the smoothest

[2]As has been pointed out in [3] all experimentally recorded range maps are sampled graph surfaces (or Monge patches), implying that the parametric dependence of any range map must be expressible by $(u, v, z(u, v))$.

fit, since it also guarantees the continuity of the second-order derivatives at the data points; the data points used for this purpose are also called control points. The data set that is required for a *B*-spline fit must contain at least four control points; and if we do use only four control points at a time, the *B*-spline curve thus generated is called a cubic segment. In some applications, it might be desirable to fit overlapping cubic segments to a data stream. It is noteworthy that when we fit a cubic segment to four control points, control points themselves may not lie on the segment. As with *B*-spline segments for the 1-dimensional case, a *B*-spline surface patch is formed by using 16 control points—which may not lie on the fitted patch (Fig. 8). In analogy with the cubic segment for the 1D case, such surface patches are also called bicubic.

For the 1-dimensional case, a *B*-spline fit to a set of four equispaced data points is given by [1, 9]

$$x(u) = UM_bG_b \tag{13}$$

$$\mathbf{x}_u:\ \frac{1}{12}\begin{array}{|c|c|c|}\hline -1 & -4 & -1\\\hline 0 & 0 & 0\\\hline 1 & 4 & 1\\\hline\end{array} \qquad \mathbf{x}_v:\ \frac{1}{12}\begin{array}{|c|c|c|}\hline -1 & 0 & 1\\\hline -4 & 0 & 4\\\hline -1 & 0 & 1\\\hline\end{array}$$

$$\mathbf{x}_{uu}:\ \frac{1}{6}\begin{array}{|c|c|c|}\hline 1 & 4 & 1\\\hline -2 & -8 & -2\\\hline 1 & 4 & 1\\\hline\end{array} \qquad \mathbf{x}_{vv}:\ \frac{1}{6}\begin{array}{|c|c|c|}\hline 1 & -2 & 1\\\hline 4 & -8 & 4\\\hline 1 & -2 & 1\\\hline\end{array}$$

$$\mathbf{x}_{uv}:\ \frac{1}{4}\begin{array}{|c|c|c|}\hline 1 & 0 & -1\\\hline 0 & 0 & 0\\\hline -1 & 0 & 1\\\hline\end{array} \qquad \begin{array}{|c|c|c|}\hline P_{11} & P_{12} & P_{13}\\\hline P_{21} & P_{22} & P_{23}\\\hline P_{31} & P_{32} & P_{33}\\\hline\end{array}$$

FIG. 9. Operators of size 3 × 3 for computing the derivatives of a range map.

where

$$U = (u^3 u^2 u 1)$$

$$M_b = \frac{1}{6}\begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 0 & 3 & 0 \\ 1 & 4 & 1 & 0 \end{bmatrix} \tag{14}$$

$$G_b = (P_1, P_2, P_3, P_4)^t$$

and P_i, $i = 1, 2, 3, 4$ are the ordinate values at the four data points. The continuous function $x(u)$ as given by the above formula is good only for the interval between the data points P_2 and P_3, and can be mapped over this interval by allowing u to vary from 0 to 1. If the continuous function was desired over, say, the interval P_3 to P_4, the four data element computation window will have to slide to the right by one element.

Extending the above formulation to the 2-dimensional case, a B-spline surface patch is given by

$$\mathbf{x}(u, v) = UM_b\mathbf{P}M_b^tV^t \tag{15}$$

$\mathbf{x}_u$: $\frac{1}{384}$

1	-23	-23	-1
-5	-115	-115	-5
5	115	115	5
1	23	23	1

$\mathbf{x}_v$: $\frac{1}{384}$

-1	-5	5	1
-23	-115	115	23
-23	-115	115	23
-1	-5	5	1

$\mathbf{x}_{uu}$: $\frac{1}{96}$

1	23	23	1
-1	-23	-23	-1
-1	-23	-23	-1
1	23	23	1

$\mathbf{x}_{vv}$: $\frac{1}{96}$

1	-1	-1	1
23	-23	-23	23
23	-23	-23	23
1	-1	-1	1

$\mathbf{x}_{uv}$: $\frac{1}{64}$

1	5	-5	-1
5	25	-25	-5
-5	-25	25	5
-1	-5	5	1

P_{11}	P_{12}	P_{13}	P_{14}
P_{21}	P_{22}	P_{23}	P_{24}
P_{31}	P_{32}	P_{33}	P_{34}
P_{41}	P_{42}	P_{43}	P_{44}

FIG. 10. Operators of size 4 × 4 for computing the derivatives of a range map.

where

$$\mathbf{P} = \left[\mathbf{P}_{ij}\right],\ i, j = 1,2,3,4, \qquad \text{and} \qquad V = (v^3\, v^2\, v 1). \tag{16}$$

Note that $\mathbf{P}_{ij}$'s represent the 4×4 matrix of data points to which the bicubic surface patch is fitted; each data point is a vector in $\mathbf{E}^3$. In keeping with the 1-dimensional case, the analytic form given by the above formula applies only over the "rectangular" cell whose vertices are given by P_{22}, P_{23}, P_{32}, and P_{33} under the assumption that the 2D region defined by $0 \le u \le 1$, $0 \le v \le 1$ maps onto this cell

$\frac{1}{24}$

1	2	1
2	12	2
1	2	1

(a)

$\mathbf{x}_u$: $\frac{1}{288}$

-1	-6	-10	-6	-1
-2	-20	-52	-20	-2
0	0	0	0	0
2	20	52	20	2
1	6	10	6	1

$\mathbf{x}_v$: $\frac{1}{288}$

-1	-2	0	2	1
-6	-20	0	20	6
-10	-52	0	52	10
-6	-20	0	20	6
-1	-2	0	2	1

$\mathbf{x}_{uu}$: $\frac{1}{144}$

1	6	10	6	1
0	8	32	8	0
-2	-28	-84	-28	-2
0	8	32	8	0
1	6	10	6	1

$\mathbf{x}_{vv}$: $\frac{1}{144}$

1	0	-2	0	1
6	8	-28	8	6
10	32	-84	32	10
6	8	-28	8	6
1	0	-2	0	1

$\mathbf{x}_{uv}$: $\frac{1}{96}$

1	2	0	-2	-1
2	12	0	-12	-2
0	0	0	0	0
-2	-12	0	12	2
-1	-2	0	2	1

(b)

FIG. 11. (a) Smoothing operator (3×3) with Gaussian weights. (b) Operators of size 5×5 for computing the derivatives of a range map.

(see Fig. 8). To obtain this analytical form for other cells, the 4×4 window will have to be shifted over accordingly.

Given this analytic description of the fitted surface patch, the partial derivatives required for the computation of curvatures at any point within this patch are given by

$$\begin{aligned}
\mathbf{x}_u &= U'M_b\mathbf{P}M_b^tV^t \\
\mathbf{x}_v &= UM_b\mathbf{P}M_t^tV^{t'} \\
\mathbf{x}_{uu} &= U''M_b\mathbf{P}M_b^tV^t \\
\mathbf{x}_{vv} &= UM_b\mathbf{P}M_b^tV^{t''} \\
\mathbf{x}_{uv} &= U'M_b\mathbf{P}M_b^tV^{t'}.
\end{aligned} \tag{17}$$

Since $u = v = 0$ corresponds to the point $\mathbf{P}_{22}$, the values of the partial derivatives at

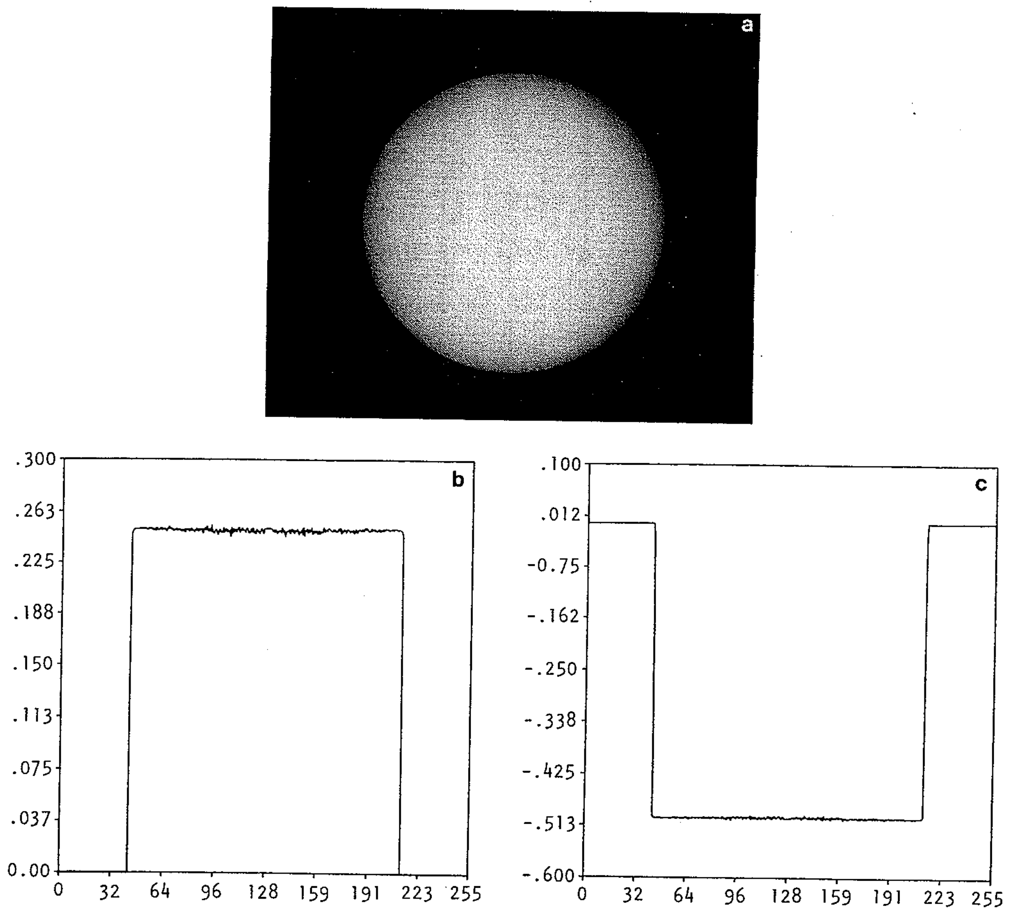


FIG. 12. (a) Illustrated here is a 256×256 synthetic range map of a sphere of radius 2 units (1 unit in the continuous domain equals 43 samples in the range map). (b) Gaussian curvatures computed along the middle horizontal line through the range map. (c) Mean curvature computed along the middle horizontal line through the range map.

this reference point are given by

$$
\begin{aligned}
\mathbf{x}_u(0,0) &= \tfrac{1}{12}(-\mathbf{P}_{11} - 4\mathbf{P}_{12} - \mathbf{P}_{13} + \mathbf{P}_{31} + 4\mathbf{P}_{32} + \mathbf{P}_{33})\\
\mathbf{x}_v(0,0) &= \tfrac{1}{12}(-\mathbf{P}_{11} - 4\mathbf{P}_{21} - \mathbf{P}_{31} + \mathbf{P}_{13} + 4\mathbf{P}_{23} + \mathbf{P}_{33})\\
\mathbf{x}_{uu}(0,0) &= \tfrac{1}{6}(\mathbf{P}_{11} + 4\mathbf{P}_{12} + \mathbf{P}_{13} - 2\mathbf{P}_{21} - 8\mathbf{P}_{22} - 2\mathbf{P}_{23} + \mathbf{P}_{31} + 4\mathbf{P}_{32} + \mathbf{P}_{33}) \quad (18)\\
\mathbf{x}_{vv}(0,0) &= \tfrac{1}{6}(\mathbf{P}_{11} - 2\mathbf{P}_{12} + \mathbf{P}_{31} + 4\mathbf{P}_{21} - 8\mathbf{P}_{22} + 4\mathbf{P}_{23} + \mathbf{P}_{13} - 2\mathbf{P}_{23} + \mathbf{P}_{33})\\
\mathbf{x}_{uv}(0,0) &= \tfrac{1}{4}(\mathbf{P}_{11} - \mathbf{P}_{31} - \mathbf{P}_{13} + \mathbf{P}_{33}).
\end{aligned}
$$

In Fig. 9 we show the finite difference operators generated by the equations above. Each of these operators yields the corresponding partial derivative of the range map at the center of the 3×3 array.

If instead of computing the partial derivatives at $u = v = 0$, we compute them at, say, $u = v = 0.5$, we obtain 4×4 operators; these are shown in Fig. 10.

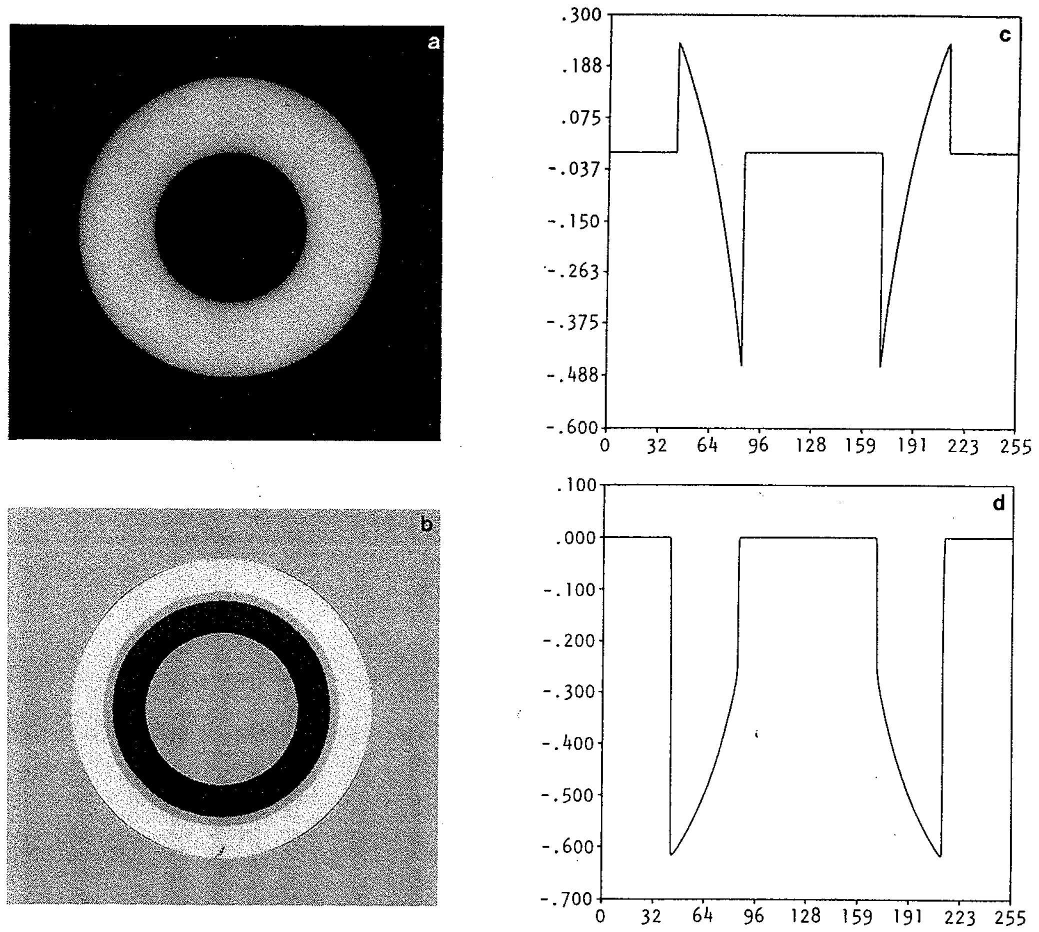


FIG. 13. Depicted here is a 256×256 synthetic range map of a torus whose minor axis is of length 1 unit and major axis 3 units. (b) A rendition of the sign of the calculated Gaussian curvature: with whites representing positive values, blacks negative, and gray the zero value. (c) Gaussian curvatures computed along the middle horizontal line through the range map. (d) Mean curvatures computed along the middle horizontal line through the range map.

Since in most cases an experimentally measured range map may be noisy, 3×3 operators may be too small for computing reliable derivatives. In such cases, one may use the 4×4 operators, or even larger sized operators that can be constructed by the following procedure. Operators of size 5×5 can be constructed by convolving our 3×3 operators of Fig. 9 with the 3×3 window (Fig. 11a) used by Brady *et al.* [6] for smoothing range maps; the resulting operators are shown in Fig. 11b. It is also possible to construct derivative operators of size larger than 5×5 by repeatedly convolving the 3×3 operators of Fig. 9 with the 3×3 smoothing window of Fig. 11a. As pointed out the Brady *et al.* [6], an n times repeated average with this smoothing window corresponds approximately to filtering the range map with a Gaussian whose standard deviation is proportional to $\sqrt{n}$.

In all the results we have shown on experimental data, the derivatives of range maps were computed by using the 5×5 operators of Fig. 11b; from these derivatives, in each case, we subsequently compute the Gaussian and mean curvatures. Note the weight distribution in the operators of Fig. 11b: the points closer to the center—where the derivative information is being computed—have larger values. This means there is a greater emphasis on range values closer to the point where the derivative is actually being computed.

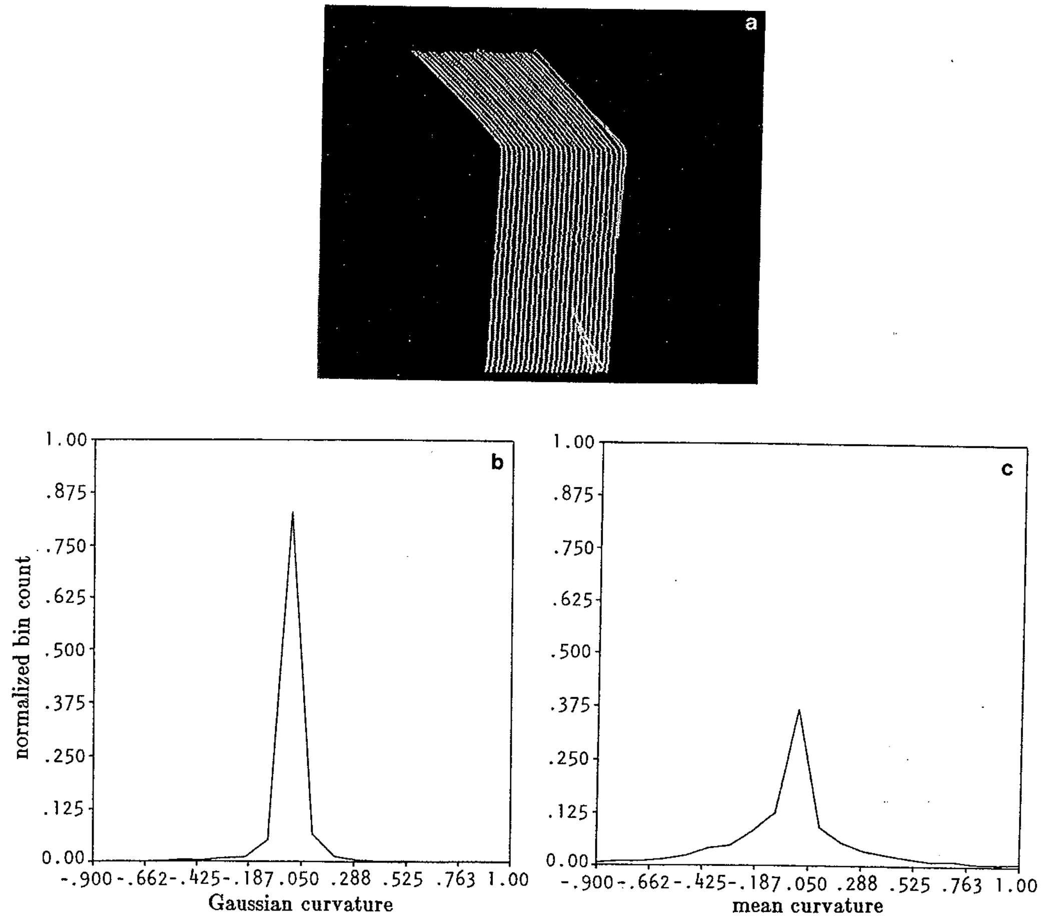


FIG. 14. (a) A light stripe image of a box. (b) Gaussian curvature histogram obtained from the range map. (c) Mean curvature histogram obtained from the range map.

Measured range maps can almost always be represented by the graph surface from (also called the Monge patch): $\mathbf{x}(u, v) = (u, v, z(u, v))$. In this case, the required partial derivatives can be obtained by applying the operators shown in Fig. 9, 10, or 11b to $z(u, v)$, the relationships between these derivatives and those of $\mathbf{x}$ being given by

$$\begin{aligned}
\mathbf{x}_u &= (1, 0, z_u)\\
\mathbf{x}_v &= (0, 1, z_v)\\
\mathbf{x}_{uu} &= (0, 0, z_{uu})\\
\mathbf{x}_{vv} &= (0, 0, z_{vv})\\
\mathbf{x}_{uv} &= (0, 0, z_{uv}).
\end{aligned}$$

When the derivatives, as calculated above, are plugged into Eqs. (5)–(10), we obtain values for the local Gaussian and mean curvature. We have tested the accuracy of this computational procedure by using synthetic range maps. Figure 12a illustrates a 256 × 256 synthetic range map of a sphere of radius 2 units (1 unit in the continuous domain equals 43 samples in the range map). Figures 12b and c

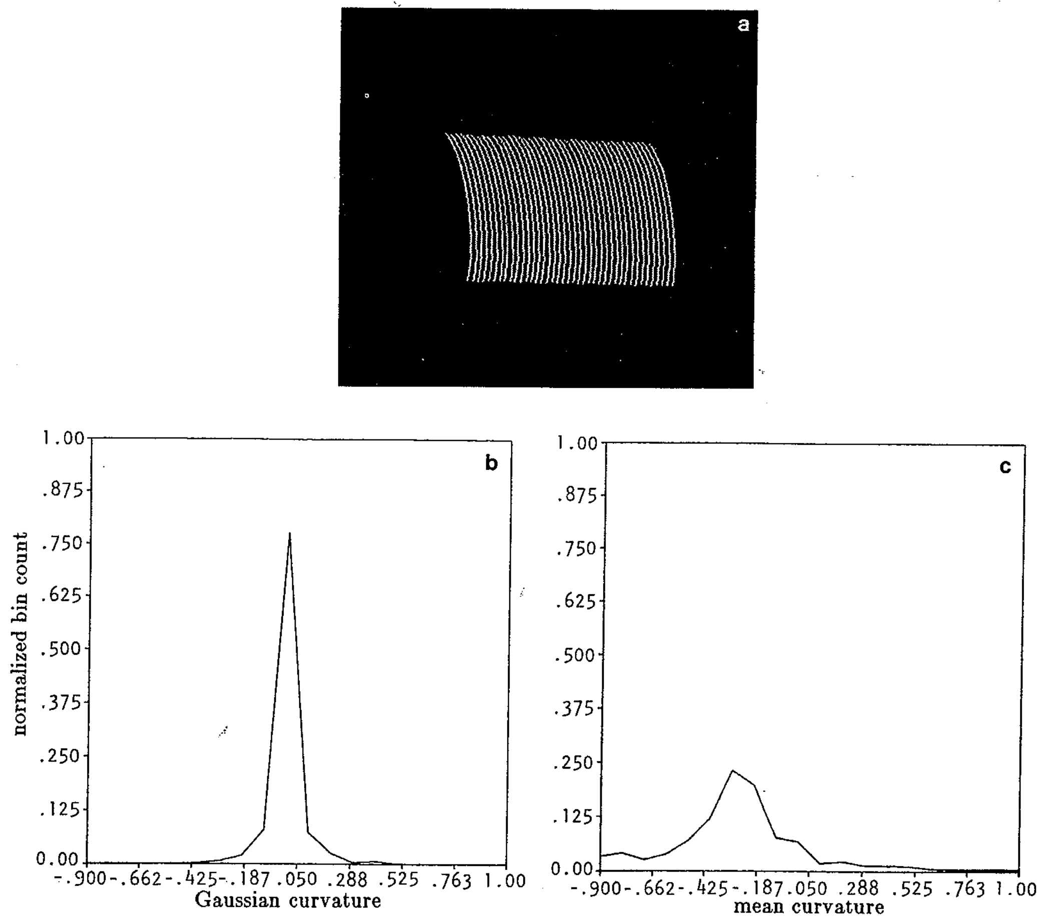


FIG. 15. (a) A light stripe image of a cylinder. (b) Gaussian curvature histogram. (c) Mean curvature histogram.

show the Gaussian and the mean curvatures, respectively, computed along the middle horizontal line through the range map of the sphere. The synthetic range data consisted of floating point numbers and did not suffer from quantization errors. The two principal curvatures for this sphere are both equal to -0.5; therefore the Gaussian curvature must equal 0.25, and the mean curvature -0.5.

Figure 13a shows a 256×256 synthetic range map of a torus whose minor axis is of length 1 unit (1 unit in the continuous domain equals 21 samples in the range map) and major axis of length 3 units. Again, the range data consists of floating point numbers and incorporates no quantization effects. Figure 13b depicts the sign of the calculated Gaussian curvature, with whites representing positive values, blacks negative, and gray and zero value. Figures 13c and d show, respectively, the Gaussian and the mean curvatures computed along the middle horizontal line through the range map in Fig. 13a.

We have also tested the computational procedure on experimentally recorded range maps obtained with structured light. In Fig. 14a is shown the light stripe image projected by a structured light sensor on a box; Figs. 14b and c show the Gaussian and mean curvature histograms constructed from the range data. We can see a peak at zero for both curvature histograms, which is what we should expect for an object consisting only of planar surfaces.

We would also like to show some results with experimental data for an object containing a cylindrical surface. Figure 15a illustrates the light stripe image for such an object. Figures 15b and c show again the curvature histograms. We can see a peak at zero in the Gaussian curvature histogram, while there is a peak at -0.32 in the mean curvature histogram. Since the radius of the cylinder used in the experiment was 1.375 units, the peak must occur over a cell corresponding to a value of -0.36. From the peak in our curvature histogram, the radius of the cylinder can be estimated as 1.563.

6. DETERMINATION OF THE IDENTITY AND ORIENTATION OF THE TOPMOST PLANAR OBJECT

Our identification scheme for topmost planar objects is based on the EGI (extended Gaussian image, also known as the orientation histogram) representation of visible surfaces. We will not elaborate on how we generate the EGI representation, that subject is treated in some detail in [17] and a bit more briefly in [12].

To make the EGI representation invariant to object size, we have constructed normalized EGI's. This can be done by normalizing the value of each cell in the orientation histogram by the total count in all the cells. Note that, in general, a normalized EGI will remain invariant to object size only under certain constraints. For example, the normalized EGI of a rectangular box remains invariant only if the length, width, and height change by the same scale; for both a cylinder and a cone, only if the ratio of the height to the radius remains invariant; for an ellipsoid, only if the ratio of the major to the minor axes remains invariant; for a torus, only if the product of the lengths of the major and minor axes remains invariant; etc.

In Fig. 16 we have laid out the identification strategy that is invoked when the topmost object is declared to be planar. As is clear from the figure, the strategy depends upon the number of visible planar surfaces as given by the number of peaks in the EGI representation.

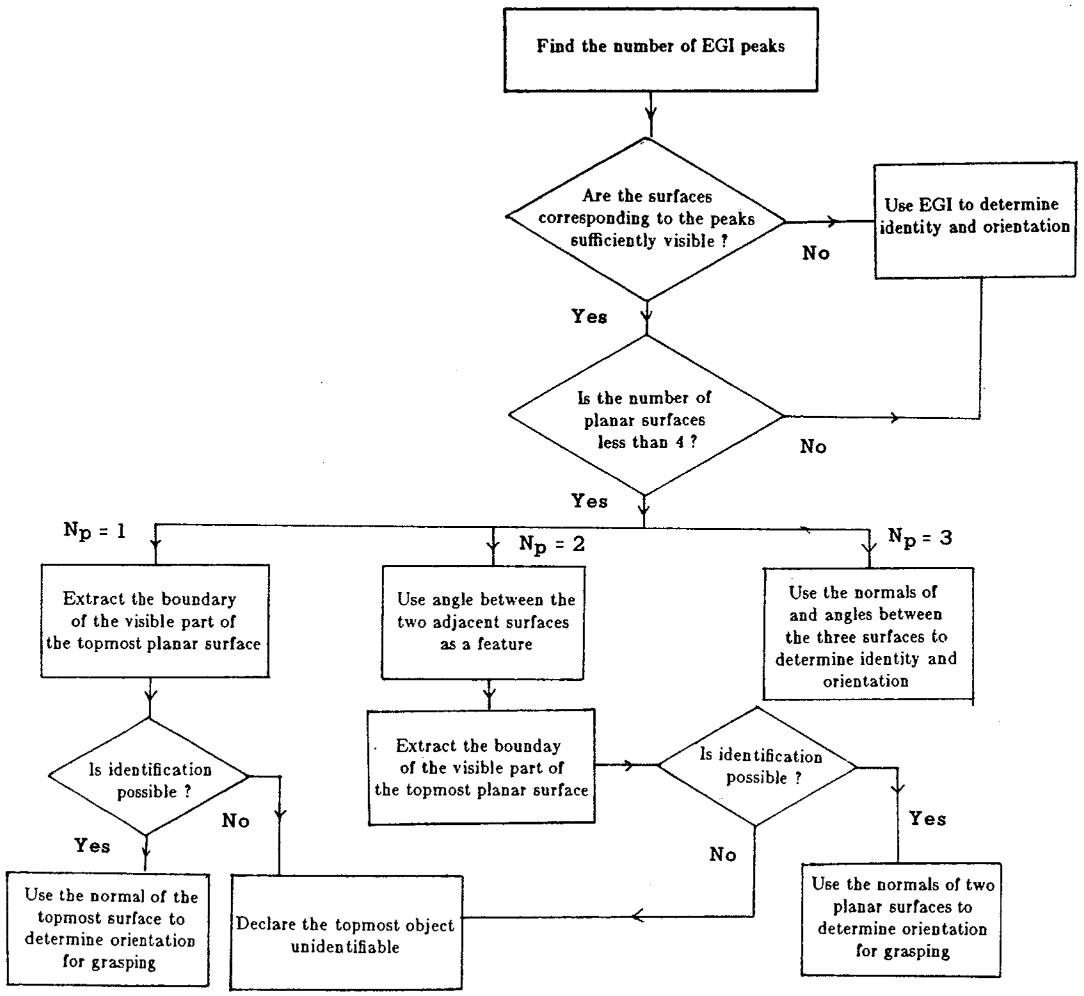

Find the number of EGI peaks
Are the surfaces corresponding to the peaks sufficiently visible ?
No
Use EGI to determine identity and orientation
Yes
Is the number of planar surfaces less than 4 ?
No
Yes
Np = 1
Np = 2
Np = 3
Extract the boundary of the visible part of the topmost planar surface
Use angle between the two adjacent surfaces as a feature
Use the normals of and angles between the three surfaces to determine identity and orientation
Extract the bounday of the visible part of the topmost planar surface
Is identification possible ?
No
Yes
Is identification possible ?
Yes
No
Use the normal of the topmost surface to determine orientation for grasping
Declare the topmost object unidentifiable
Use the normals of two planar surfaces to determine orientation for grasping

FIG. 16. Illustrated here is the strategy for determining the identity and orientation of the topmost object if it is found to be planar. N_p denotes the number of EGI peaks corresponding to the sufficiently visible planar surfaces.

The most natural thing to do with an experimentally obtained EGI representation is to match it against the EGI's of candidate objects; the matching process being carried out over all allowable rotations between the unknown object-part and the candidate object. If the matching succeeds, it also yields the orientation of the object part.

As Fig. 16 shows, in general we only perform matching when the number of major visible surfaces (as given by the number of distinguished peaks in the EGI) exceeds three. Whether or not an EGI peak is distinguished is determined simply by thresholding the EGI—a threshold of 0.2 has worked well for a number of scenes we have examined.[3]

When the number of visible planar surfaces on the topmost object is equal to or less than three, we try to avoid identifying the object by EGI matching; the reason

[3]A peak acceptance threshold of 0.2 means that we want at least one fifth of all range cells that contribute to the EGI to correspond to only one planar surface. Clearly, a threshold of 0.2 means that when we detect major surfaces, we do not expect to find more than five of them in a single object.

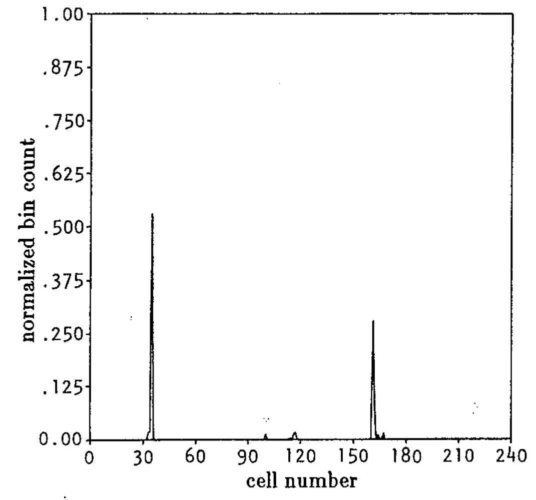


FIG. 17. Shown here is the EGI of the topmost planar object (a cube) isolated from the scene of Fig. 2a.

being that for such cases there is not enough "peak structure" in the experimentally obtained EGI and we might as well depend directly on the angles between the adjoining surfaces. When the number of major visible surfaces is only one or two, clearly this angular information would not be enough, so we also include the shape of the boundary of the topmost planar surface (from among all the planar surfaces that are visible for the topmost object). Of course, depending upon what part of the topmost surface of the topmost object is visible, this strategy does not always work.

Figure 17 shows (in a linear format) the EGI of the topmost planar object (a cube) isolated from the scene of Fig. 2a. We detect two distinguished peaks from its EGI. The center coordinates of the EGI cell for the first peak are $(-0.6388, -0.1036, 0.7623)$, and the value of the EGI at this cell is 0.53, meaning that 53% of the visible area for the topmost object is taken up by the surface corresponding to this peak. The other cell is at center coordinates $(-0.6875, -0.1563, -0.7031)$ with an EGI value of 0.28—this again qualifies as a distinguished peak. The angle between two surfaces is measured to be 94.6°. Since we only have two major surfaces, in such a case the boundary shape would be included in the identification process.

7. DETERMINATION OF THE IDENTITY AND ORIENTATION OF THE TOPMOST CURVED OBJECT

For curved objects, after extracting the visible part we compute the local surface normals and surface curvatures. We then construct the Gaussian and mean curvature histograms; in the rest of this discussion we will refer to the Gaussian curvature histogram as the K-histogram and the mean curvature histogram as the H-histogram. From the surface normals, we again construct an EGI.

In our identification strategy, we first constrain the object type by using the characteristics of the K-histogram and the H-histogram. For this purpose, a curvature histogram is classified into one of the following six categories:

1. *Positive only with peak.* A histogram falls in this class if the curvature values are practically all positive, and there exists a peak value over these values. If the histogram had zero counts in all cells except for where the peak occurs, that would mean all points on the curved surface have the same curvature.

2. *Positive only without peak.* This is the same as the previous class, except that the positive half of the histogram is not peaked anymore. This situation corresponds to a surface which has positive, but varying, curvature everywhere it is visible.

3. *Negative only with peak.* The same as for case 1, except that the histogram is confined to the negative half.

4. *Negative only without peak.* The same as for case 2, except that the histogram is confined to the negative half.

5. *Peak at zero.* A curvature histogram falls in this class if it has only one peak and that occurs over the bin corresponding to zero curvature.

6. *Positive and negative without peak.* This corresponds to surfaces with continuously varying curvatures.

We also have a category called *unclassified*, which is used when a histogram, on account of the thresholds used for peak detection, cannot be classified into one of the six categories listed above. Table 1 shows the characteristics of the curvature histograms for different curved surfaces.

When dealing with noisy experimental data, one has to be careful with characterizing curvature histograms that have no peaks; the reason being that in particular cases the thresholds used for peak detection might simply be too high. When we run into such histograms, we treat them with a bit of suspicion and supplement the evidence with the EGI representation.

Figures 18a and b depict the K and H histograms constructed from the visible portion of the topmost object (a sphere) in the scene of Fig. 6a. The K-histogram falls into the class *positive only with peak* and the H-histogram into the class *negative only without peak.*[4] From Table 1, we therefore constrain the object type to be either spherical or elliptical.

With the object type thus constrained, we proceed to use the prototype EGIs for a sphere and an ellipse for final identification. Figure 18c shows the observed EGI of the topmost object (a sphere). In this case, the process of matching this EGI with the prototypes for a sphere and an ellipsoid, yielded a smaller sum with the sphere case.

Figures 19a and b show the K- and H-histograms constructed from the topmost object, in this case a torus, for the scene of Fig. 7a. Figure 19c is the observed EGI. The K-histogram falls into the class *positive and negative without peak*, whereas the H-histogram falls into the *unclassified* class; the reason being that although the

[4] For all K-histograms, we have used 0.5 for peak detection, meaning that 50% of the recorded surface points must fall in the bin corresponding to the peak—in other words, 50% of the points on the visible surface must have the same Gaussian curvature, whose value corresponds to where the peak occurs. For all H-histograms, we have used 0.25 for peak detection, meaning that 25% of the points on the visible surface must have a mean curvature corresponding to the peak in the histogram.

TABLE 1

Characteristics of Curvature Histograms for Different Curved Surfaces

Curvature histogram / Surface	K-histogram	H-histogram
Cylindrical	Peak at zero	Negative only with peak
Conical	Peak at zero	Negative only without peak
Spherical	Positive only with peak	Negative only with peak
Elliptical	Positive only without peak	Negative only without peak
Toroidal	Positive and negative without peak	Negative only without peak or positive and negative without peak

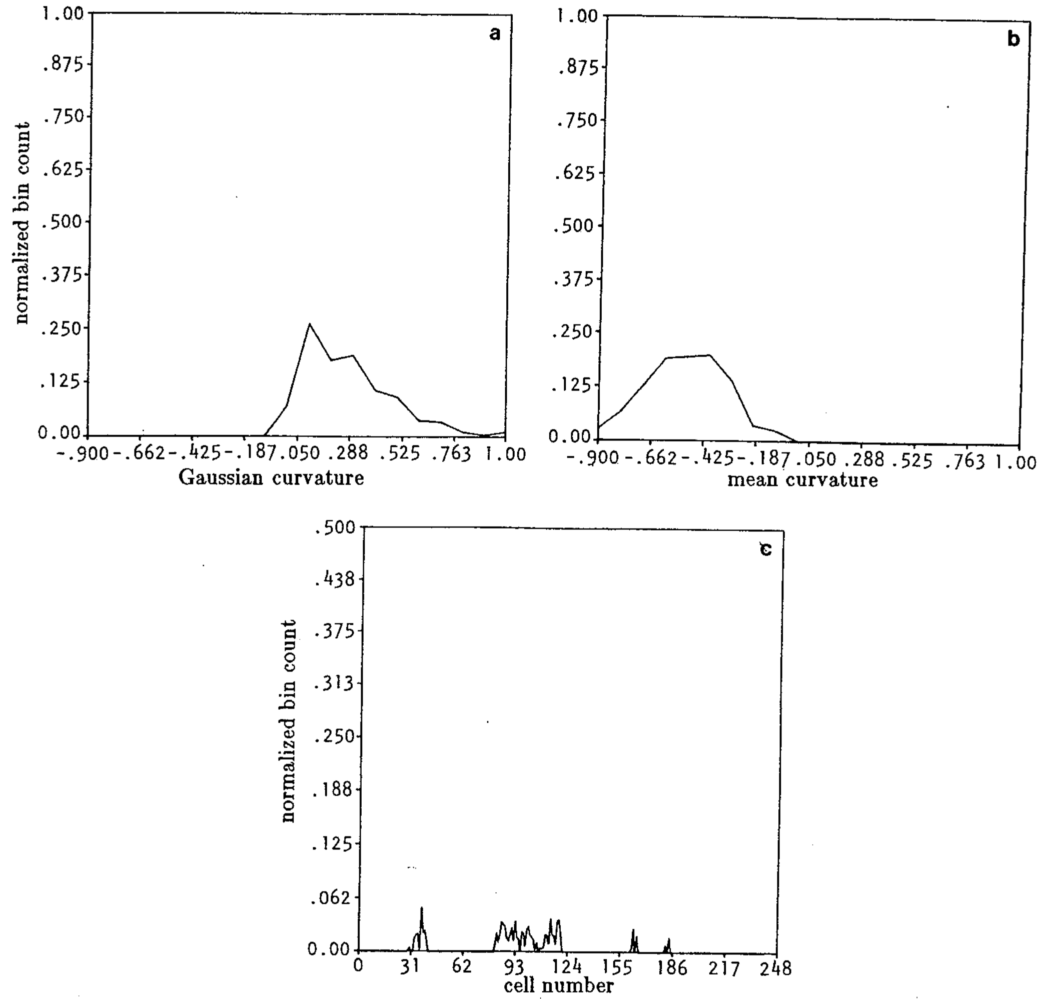
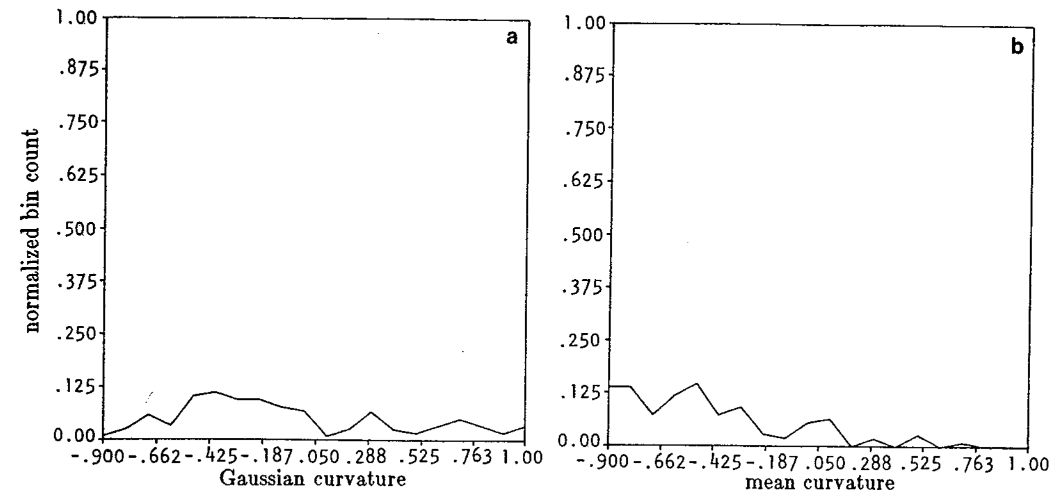


FIG. 18. (a) Gaussian curvature histogram constructed from the visible portion of the topmost object (a sphere) in the scene of Fig. 6a. (b) Mean curvature histogram constructed from the same region. (c) The EGI calculated from the range map data for the topmost object.



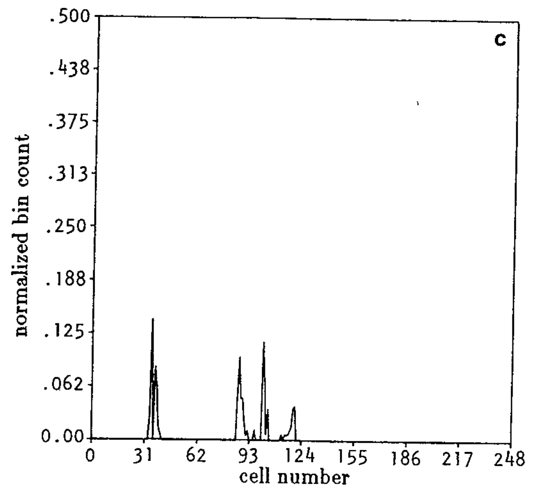


FIG. 19. (a) Gaussian curvature histogram constructed from the topmost object (a torus) of the scene in Fig. 8a. (b) Mean curvature histogram constructed from the same region. (c) The EGI calculated from the range map data for the topmost object.

distribution is approximately *negative only without peak*, however, the percentage of the negative curvature values is below the acceptance threshold of 0.85.[5]

From Table 1, the classification for the K-histogram tells us that the topmost object might be a torus. A confirmation of this fact could then be made by matching the EGI for the visible surface against the prototype EGI of a torus. However, for the example shown that did not work because of the small size of the visible surface of the torus in relation to the total surface of the object.

Note that if both the K and the H histograms had been declared unclassified, at this point we would have given up attempting to determine the identity and attitude of the topmost object; this kind of situation can only happen when either the range data is full of errors or the topmost object is not sufficiently visible—both of these factors would cause the EGI to also become unreliable. In such cases, the robot can disturb the scene and reexamine it.

[5]For a histogram to be characterized by one of the six classes, the total count in all the bins corresponding to the distinguishing feature of that class must exceed a threshold—we have used 0.85. So, if a histogram is to be called *negative only with peak*, at least 85% of the points on the visible curved surface must have negative curvature.

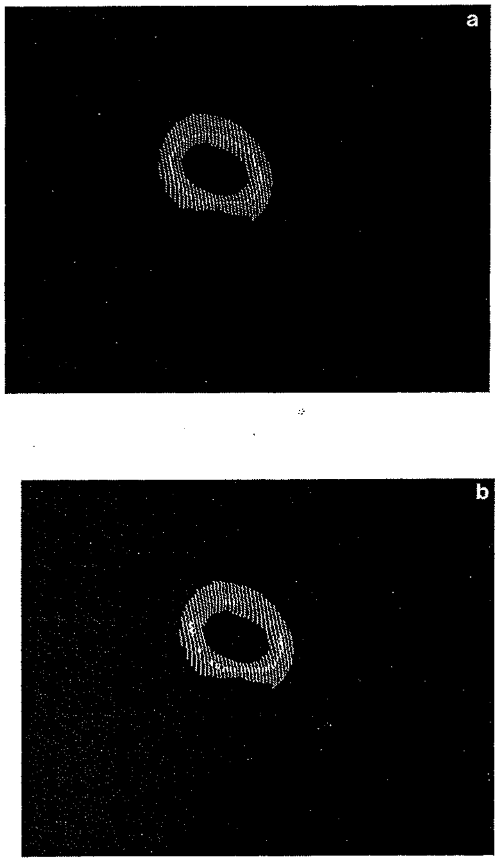


FIG. 20. (a) Shown here is a skeleton of the visible part of the topmost object (a torus) of the scene of Fig. 7a. (b) Needle diagram illustrating the surface normals computed at the points on the skeleton; only the points on the skeleton where Gaussian curvatures are approximately zero have been used.

For a torus, positive identification can sometimes be obtained by skeletonizing the extracted surface; this procedure would clearly not work if the torus is heavily occluded. Such skeletons can also be used for attitude determination if we use the heuristics that the skeleton points are characterized by zero Gaussian curvature and the surface normals of the points on the torus at which Gaussian curvatures are zero are parallel to the axis of revolution of a torus.

In Fig. 20a is shown a skeleton of the visible surface extracted in Fig. 7d. Figure 20b shows a needle diagram illustrating the surface normals computed at the points on the skeleton. (Note that only the points on the skeleton whose Gaussian curvatures are close to zero have been used.)

8. CONCLUDING REMARKS

The research reported in this paper was motivated by our interest in how to best use vision feedback for robot tasks such as bin picking and heap sorting. The techniques we have presented should work for a large number of industrial type objects. Do our techniques represent the best in terms of speed of execution and performance—we do not yet have an answer to that question. Processing 3D vision data is inherently expensive from a computational standpoint, but then one does gain in reliability in relation to photometric data. It remains to be seen whether algorithms, such as those reported here, will prove viable for the intelligent robotics of the near and distant future.

A justifiable criticism of this report would be that we have not conducted any statistical studies on the reliability and robustness of our algorithms—in that regard we have not departed from the long, but nevertheless unfortunate, tradition in much of archival literature in computer vision. Our future research will focus on such reliability and robustness issues.

ACKNOWLEDGMENTS

The authors thank all members of the Robot Vision Lab for providing a stimulating research environment; we are especially grateful to Robert Safranek and Kirk Smith for the robot manipulation software and the robot-computer communication software. We are also thankful to Myung S. Kim of the Department of Computer Science at Purdue University for many fruitful discussions on computational models for generating surfaces of 3D objects.

REFERENCES

1. B. A. Barsky, A description and evaluation of various 3D models, *IEEE Comput. Graphics Appl.*, January 1984, 38–52.
2. P. R. Beaudet, Rotationally invariant image operators, in *Proc. Int. Joint Conf. Pattern Recognit.*, November 1978, pp. 579–583.
3. P. J. Besl and R. C. Jain, Invariant surface characteristics for 3D object recognition in range images, *Comput. Vision Graphics Image Process.* **33**, 1986, 33–80.
4. R. Bolles, A ransac-based approach to model fitting and its application to finding cylinders in range data, in *Proc. 7th Int. Joint Conf. Artif. Intell.*, 1981, Vol. 2, pp. 637–643.
5. K. L. Boyer, H. S. Yang, and A. C. Kak, *3-D Vision for Pile Analysis*, Purdue University Technical Report TR-EE-84-17, 1984.
6. M. Brady, J. Ponce, A. Yuille, and H. Asada, Describing surfaces, in *Proc. 2nd Int. Sympos. on Robotics Res.*, MIT Press, Cambridge, Mass., 1985.
7. J. Dessimoz, J. R. Birk, R. B. Kelley, H. A. S. Martins, and Chi Lin I, Matched filters for bin picking, *IEEE PAMI*, *Trans. Pattern Anal. Mach. Intell.* **PAMI-6**, No. 6, 1984, 686–697.
8. Manfredo P. Do Carmo, *Differential Geometry of Curves and Surfaces*, Prentice–Hall, Englewood Cliffs, N.J., 1976.
9. J. D. Foley and A. V. Dam, *Fundamentals of Interactive Computer Graphics*, Addison–Wesley, Reading, Mass., 1982.
10. B. K. P. Horn and K. Ikeuchi, The mechanical manipulation of randomly oriented parts, *Sci. Amer.*, Aug. 1984, 100–111.
11. K. Ikeuchi, Recognition of 3-D objects using the extended Gaussian images, in *Proc. 7th Int. Joint Conf. Artif. Intell.*, 1981, pp. 595–600.
12. A. C. Kak, K. L. Boyer, C. H. Chen, R. J. Safranek, and H. S. Yang, A knowledge-based robotic assembly cell, *IEEE EXPERT*, Spring 1986, pp. 63–83.
13. A. C. Kak, Depth perception for robots, in *Handbook of industrial Robotics* (S. Nof, Ed.), pp. 272–319, Wiley, New York, 1985.
14. R. B. Kelley, H. A. S. Martins, J. R. Birk, and J. Dessimoz, Three vision algorithms for acquiring workpieces from bins, *IEEE Proc.* **71**, No. 7, 1983, 803–820.
15. B. O'Neill, *Elementary Differential Geometry*, Academic Press, New York/London, 1966.
16. M. Oshima and Y. Shirai, Object recognition using three dimensional information, *IEEE Pattern Anal. Mach. Intell.* **PAMI-5**, 1983, 353–361.
17. H. S. Yang and A. C. Kak, Determination of the identity, position and orientation of the topmost object in a pile, in *Proc. Third Workshop on Computer Vision, Representation and Control*, Oct. 1985, pp. 38–48.
18. H. S. Yang and A. C. Kak, Determination of the identity, position and orientation of the topmost object in a pile: Some further experiments, in *Proc. IEEE Conf. on Robotics and Automation*, April 1986, pp. 293–298.
19. H. S. Yang and A. C. Kak, Edge extraction and labeling from structured light 3D vision data, submitted for publication.