FROM SHAPE FROM SHADING TO OBJECT RECOGNITION

ANA ORTEGA and MUBARAK SHAH
Computer Vision Laboratory, Computer Science Department
University of Central Florida, Orlando, Florida 32816, USA
E-mail: {*ortega,shah*}*@cs.ucf.edu*

Recognition of objects is one of the main goals of computer vision. Several approaches have been proposed to solve this problem using 3-D shapes. In most of them it is assumed that the 3-D shape (depth map) is available. Several object recognition systems use range images to extract the 3-D shape.

We present a method that uses a shape from shading algorithm to perform 3-D object recognition for simple objects. This method extracts the 3-D information from a single intensity image, then segments the object into regions. After computing the properties of the regions, it compares the input object with the model objects in the database. To test our method, several images with slightly different viewing angles of single objects are matched against five models in the database.

Keywords: Shape from shading, object recognition, integration of modules.

1. INTRODUCTION

Recognition of objects is one of the main goals of computer vision. Several approaches have been proposed to solve this problem using 3-D shapes. In most of them it is assumed that the 3-D shape (depth map) is available. Several object recognition systems use range images to extract the 3-D shape. Since correct depth information depends only on geometry and is independent of illumination and reflectivity, intensity image problems with shadows and surface markigs do not occur in depth maps. Therefore, the process of recognizing objects by their shape should be less difficult in range images than in intensity images.[2]

There are a number of methods to derive 3-D shapes from intensity images. These methods include shape from stereo, shading, motion, texture, etc. So far there is no object recognition approach to our knowledge that obtains the object's depth information employing a shape from shading algorithm. The purpose of this paper is to propose an object recognition method that gets the depth information from an intensity image using a shape from shading algorithm and uses it for 3-D object recognition of real objects.

One advantage of this method is that we need only one intensity image as input. Methods such as shape from stereo and shape from motion require two or more intensity images to estimate the depth map. An additional advantage of our method is that we do not need laser scanners to acquire depth information. Finally, using 3-D data enables us to compute properties, such as surface types (e.g. ellipsoid, elliptic paraboloid), that could not be computed using only 2-D information.

2. THE METHOD

We consider an object as a solid mass composed of convex parts and Lambertian surfaces. A **region** is defined as a subset of the surface of an object. This method has four main phases: extraction of 3-D information, object segmentation, region properties computation, and matching.

The first step is to estimate the depth information using a shape from shading algorithm. In this approach, we use a linear shape from shading method proposed by Tsai and Shah.[7] The second step is to segment the input image. The segmentation is performed by an edge-based approach. The region boundaries are detected using the negative curvature extrema in the surface curvature. The negative curvature extrema form boundaries which we connect to obtain the object regions. In the third stage, the region properties (surface type, centroid, area and perimeter) to be used in the matching are computed. The fourth stage, matching, has two phases. In the first phase, for each model, an interpretation tree is used along with unary and binary constraints to obtain a set of possible matches.[4] In the second phase, the best match is selected based on the surface type similarities. The next step would be to obtain the pose estimation and verification, but the step is not addressed in this paper.

3. EXTRACTION OF 3-D DATA: SHAPE FROM SHADING ALGORITHM

The input to the system is a real intensity image. We smooth the image once to remove noise. Smoothing is done using two one-dimensional Gaussian masks.[1] Next, we estimate the light source direction from the smoothed intensity image using Lee and Rosenfeld's algorithm.[5] Once we have the light source direction, we apply the linear shape from shading algorithm by Tsai and Shah.[7] This method uses the discrete approximations for p and q in terms of Z in the reflectance function, and then linearizes it in $Z(x,y)$. The reflectance equation is given by $E(x,y) = R(p,q)$ where $E(x,y)$ is the gray level at pixel (x,y) and R is given by

$$R(p,q) = \frac{1 + pp_s + qq_s}{\sqrt{1+p^2+q^2}\sqrt{1+p_s^2+q_s^2}}\,. \tag{1}$$

Using the following discrete approximations for p and q

$$p = \frac{\partial Z}{\partial x} = Z(x,y) - Z(x-1,y)\,, \tag{2}$$

$$q = \frac{\partial Z}{\partial y} = Z(x,y) - Z(x,y-1)\,, \tag{3}$$

the reflectance equation can now be rewritten as:

$$f(Z(x,y)) = E(x,y) - R(Z(x,y) - Z(x-1,y), Z(x,y) - Z(x,y-1)) = 0\,. \tag{4}$$

By taking the Taylor series expansion of this function f about $Z(x,y) = Z^{n-1}(x,y)$, where $Z^{n-1}(x,y)$ is the depth at the $n-1$ iteration, up through the first-order terms, one has

$$\begin{aligned} 0 &= f(Z(x,y)) \\ &\approx f(Z^{n-1}(x,y)) + (Z(x,y) - Z^{n-1}(x,y))\,\frac{df}{dZ(x,y)}\,(Z^{n-1}(x,y))\,. \end{aligned} \tag{5}$$

Then for $Z(x,y) = Z^n(x,y)$, the depth map at the nth iteration, can be solved directly as follows:

$$Z^n(x,y) = Z^{n-1}(x,y) + \frac{-f(Z^{n-1}(x,y))}{\frac{df}{dZ}(Z^{n-1}(x,y))} \tag{6}$$

where

$$\begin{aligned} &\frac{df(Z^{n-1}(x,y))}{(dZ)} \\ &= -1 * \left(\frac{(p_s + q_s)}{\sqrt{p^2+q^2+1}\sqrt{p_s^2+q_s^2+1}} - \frac{(p+q)(pp_s+qq_s+1)}{\sqrt{(p^2+q^2+1)^3}\sqrt{p_s^2+q_s^2+1}} \right). \end{aligned} \tag{7}$$

Now, assuming the initial estimate of $Z^0(x,y) = 0$ for all pixels, the depth map can be iteratively refined using Eq. (6).

This method results in a extremely simple iterative scheme for computing depth. It is a linear method and requires only two or three iterations to get a good relative depth map. This shape from shading method assumes that the surface is Lambertian and its albedo is constant. Figure 1 shows the intensity images of the objects used in this work and the corresponding plots of reconstructed 3-D shape.

4. OBJECT SEGMENTATION

Once we have obtained the relative depth map, we apply a segmentation approach similar to the one applied by Fan.[3] Using the object surface's directional curvature values, we obtain the curvature negative extrema. The negative extrema correspond to the object's region boundaries. Next, those boundaries are linked to form the boundaries of initial regions which will be merged to obtain the final object segmentation.

The detailed steps in object segmentation are as follows: first, the depth map, $Z(x,y)$, is smoothed with three σ values: 0.5, 1.0 and 1.5. Then, for each smoothed depth map and for the original depth map, the directional curvatures are computed in four different directions (0°, 45°, 90°, 135°). This process creates a total of sixteen directional curvature sets, each one with a different σ and different angle. The directional curvature, K_θ, in direction θ is given by:

$$K_\theta = -\frac{Z_{\theta\theta}}{(1+Z_\theta^2)^{3/2}} \sqrt{\frac{1+(Z_x\cos\theta - Z_y\sin\theta)^2}{1+Z_x^2+Z_y^2}}\,, \tag{8}$$

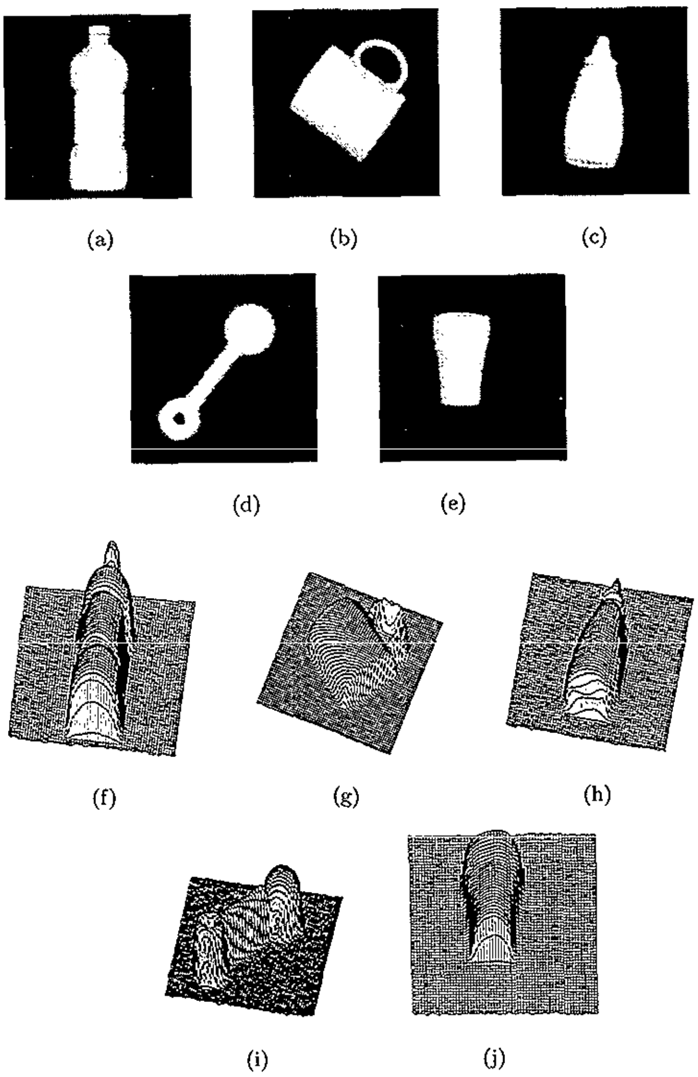

Fig. 1. Intensity images for (a) Bottle, (b) Cup, (c) Mustard, (d) Toy and (e) Glass. Estimated depth maps using shape from shading method for (f) Bottle, (g) Cup, (h) Mustard (i) Toy and (j) Glass.

where Z_x and Z_y are the first derivatives of the depth map, $Z(x, y)$, in the x and y directions, Z_θ and $Z_{\theta\theta}$ are the first and second derivatives in the θ direction. Next, we detect the negative extrema in each of the sixteen directional curvature sets and represent them in sixteen binary images, the black pixels being the extrema. In Fig. 2, we show the result for the negative curvature extrema for the object Bottle in the 90° direction. We can see that the smaller the σ value, the greater the amount of detail. Consequently for the lowest σ most of the details will be detected, including the noisy features. On the other hand, the highest σ value will contain less noisy features, but their localization will be imprecise.

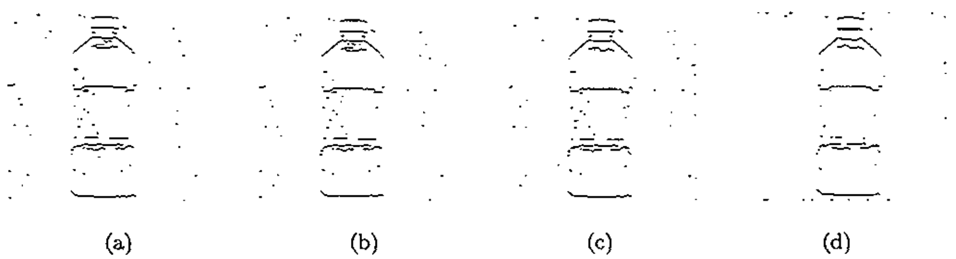

Fig. 2. Negative curvature extrema for Bottle in 90° for (a) $\sigma = 0.0$, (b) $\sigma = 0.5$, (c) $\sigma = 1.0$, (d) $\sigma = 1.5$.

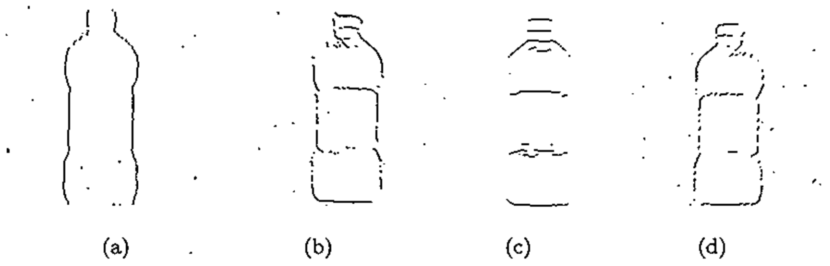

Fig. 3. After scale space tracking for Bottle in degrees (a) 0°, (b) 45°, (c) 90° and (d) 135°.

A scale-space tracking algorithm is applied next on the negative extrema obtained in the same direction to combine the results obtained at different scales. In this case, we are combining the negative curvature extrema obtained at different σ values in the same direction. The detection of the curvature negative extremum is done at the coarsest level (*highest* σ) and the localization is done at the most fine level ($\sigma = 0.0$). For each curvature negative extremum at $\sigma = 1.5$, we look for its corresponding negative extremum in the image at $\sigma = 1.0$ within a 2-pixel distance in θ direction. If found, we continue looking for its corresponding negative extrema in the extrema image with the next lower σ value ($\sigma = 0.5$). Finally, the position of the corresponding negative extremum at $\sigma = 0.0$ is marked, if found. If at any level a corresponding negative extremum is not found, that location will not be marked. The scale space tracking will result in four binary images, one for each direction. See Fig. 3 for results of scale space tracking. The four resulting binary images are merged using the logical OR operator.

We observed that noisy contours were attached to the actual region boundaries. Those contours must be removed before connecting these boundaries because they may produce false boundary connections. The noisy contours are removed as follows: first, a line thinning algorithm is applied. Second, every line end point is traced until a fork is reached. If the number of points traced is less than four, the points are removed. See Fig. 4 for results after removing noisy contours.

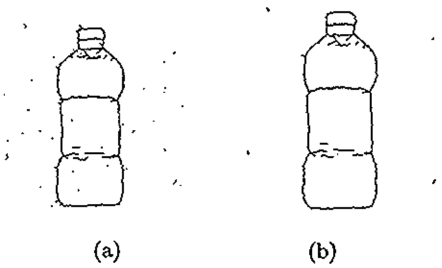

Fig. 4. (a) Result after the logical OR operation for Bottle. (b) Result after removing noisy contours.

An edge linker algorithm is applied to close boundaries and get the surface regions. The edge linker finds open boudaries and tries to close them, following the direction of the end point up to a distance of 10 pixels. To separate the object regions from the background, only the regions whose average depth is above the background level are taken into consideration.

Next, those regions connected by *weak boundaries* are merged. A boundary is weak if more than 60% of the pixels forming it were added during the edge liking. The merging is done in a recursive fashion. In Fig. 5, we show the results for the final segmentation for five objects.

5. REGION PROPERTIES

Four region properties are used in this method: 3-D surface area, centroid, perimeter and surface type of each significant region. A region is **significant** if its area is greater than 500. These properties are to be used later in the matching phase.

The **area**, A, of each 3-D surface is computed as follows:

$$A = \iint \sqrt{Z_x^2 + Z_y^2 + 1}\, dx\, dy\,.$$

Let n be the total number of points in a region. The coordinates of a point, i, in the region are denoted by (x_i, y_i, z_i). The **centroid**, c, of that region is given by $(\bar{x},\ \bar{y},\ \bar{z})$ where

$$\bar{x} = \frac{\sum_{i=1}^{n} x_i}{n}, \qquad \bar{y} = \frac{\sum_{i=1}^{n} y_i}{n}, \qquad \bar{z} = \frac{\sum_{i=1}^{n} z_i}{n}.$$

The **perimeter**, P, of a region is defined as the total number of pixels that are on the boundary of the region.

The **surface type** is classified as one of the eight possible types: ellipsoid, hyperboloid of two sheets, hyperboloid of one sheet, cone, elliptic paraboloid, hyperboloid paraboloid, invalid, point or conic section.

The surface type is obtained by fitting the quadratic polynomial to each region:

$$f(x,y,z) = Ax^2 + By^2 + Cz^2 + Dxy + Exz + Fyz + Gx + Hy + Iz + J = 0\,. \quad (9)$$

The coefficients A, B, C, D, E, F, G, H, I and J are determined using the least square fit method.

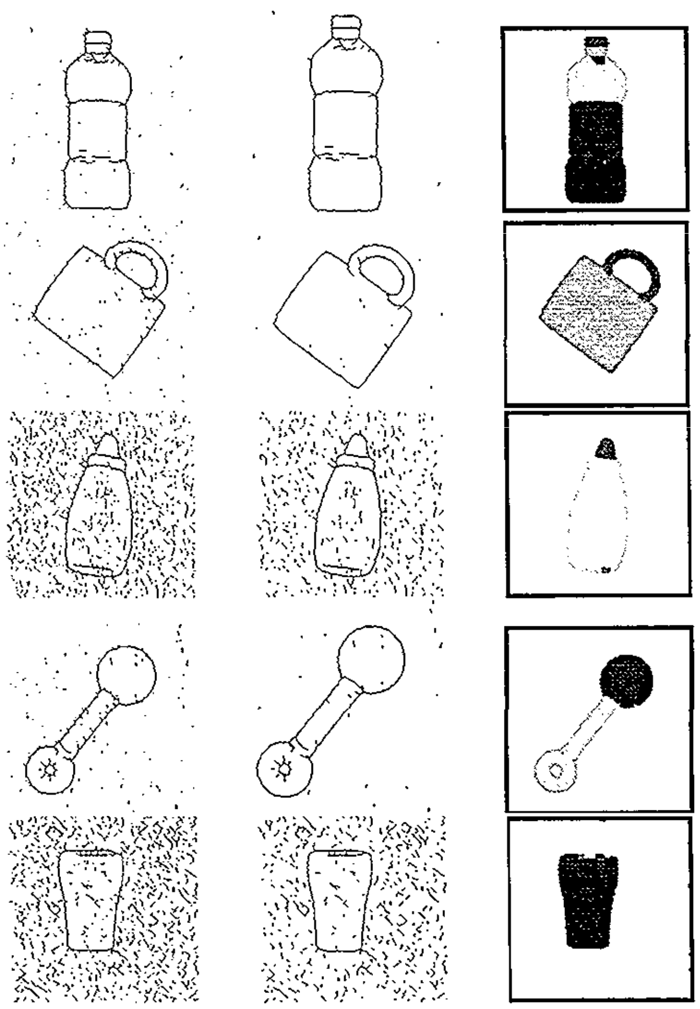

Fig. 5. (a) Results of scale space tracking. (b) Results after removal of noisy contours. (c) Final segmentation. Different gray levels indicate different regions: Bottle (6 regions), Cup (2 regions), Mustard (4 regions), Toy (3 regions) and Glass (5 regions).

Once we know the values of each coefficient, we get the surface types. Levin[6] uses the following classification approach:

Let

- $T_1 = A + B + C$,
- $T_2 = AB + BC + AC - D^2 - E^2 - F^2$,
- $T_3 = ABC + 2DEF - AE^2 - BF^2 - CD^2$,
- $D_2 = AB + BC + C + A + AC + B - D^2 - E^2 - F^2 - G^2 - H^2 - J^2$,
- $D_3 = ABC + AB + AC + BC + 2 * (DEF + FGJ + DGH + EHG) - (C+1)D^2 - (A+1)E^2 - (B+1)F^2 - (B+C)G^2 - (A+C)H^2 - (A+B)J^2$,

- $D_4 = \begin{vmatrix} A & D & E \\ D & B & F \\ E & F & C \end{vmatrix}$.

The classification is done as follows[6]:

- *ellipsoid*: $T_3 \neq 0$ *and* $(T_2 > 0$ *and* $(T_1 * T_3) > 0$ *and* $D_4 < 0)$
- *invalid*: $T_3 \neq 0$ *and* $(T_2 > 0$ *and* $(T_1 * T_3) > 0$ *and* $D_4 > 0)$
- *point*: $T_3 \neq 0$ *and* $(T_2 > 0$ *and* $(T_1 * T_3) > 0$ *and* $D_4 = 0)$
- *hyperboloid 2 sheets* (*hyperboloid-2*) : $T_3 \neq 0$ *and* $(T_2 \Leftarrow 0$ *or* $(T_1 * T_3) \Leftarrow 0$ *and* $D_4 < 0)$
- *hyperboloid 1 sheet* (*hyperboloid-1*) : $T_3 \neq 0$ *and* $(T_2 \Leftarrow 0$ *or* $(T_1 * T_3) \Leftarrow 0$ *and* $D_4 > 0)$
- *cone*: $T_3 \neq 0$ *and* $(T_2 \Leftarrow 0$ *or* $(T_1 * T_3) \Leftarrow 0$ *and* $D_4 = 0)$
- *elliptic paraboloid*: $T_3 = 0$ *and* $D_4 < 0$
- *hyperboloid paraboloid*: $T_3 = 0$ *and* $D_4 > 0$
- *conic section*: $T_3 = 0$ *and* $D_4 = 0$.

6. THE MATCHING

Our recognition system has a database with the models of the objects we want to recognize. The properties of the *models* are obtained in the same way that the properties of the *data* objects are.

Matching is performed in two stages. In the first stage, we use an interpretation tree for each model object to select a list of possible matches.[4]

In the interpretation tree, there are n levels, where n is the number of regions in the *data* object. Each node has $m + 1$ siblings, where m is the number of regions in the *model* object. Each node in the tree represents a match of a region of the object *data* (the node level) to a region of the *model* object (the position of the node with respect to its siblings). A path from the root to a node in the bottom level of the tree matches each region of the data to some region in the model or a *null* region. The null region allows for the case that the data object has more regions than the model and occlusion. The tree is pruned in a depth-first manner. We apply the unary constraint, region compactness, between single regions and the binary constraints between the current data region and model region and each of the regions matched in its path. If all the constraints are satisfied, continue going down the tree; otherwise, back up and continue onto the next branch.

We prune each tree using unary and binary constraints. In a unary constraint the properties between *one* region in the model (m) and *one* region in the data (d) are compared.

We use one unary constraint, the region compactness constraint.

- *Compactness* (φ): Compactness is defined as the ratio of area over perimeter squared. $\varphi = \frac{A}{P^2}$.

If $|\frac{\varphi^d}{\varphi^m}| < (1+thr_\varphi)$ and $|\frac{\varphi^d}{\varphi^m}| > (1-thr_\varphi)$ then the regions are consistent, where d and m superscripts denote respectively data and model. The threshold is thr_φ.

In a binary constraint the properties between *two* regions in the model (m) and *two* regions in the data (d) are compared. We use two binary constraints, the relative distance between two centroids and the ratio of surface areas difference:

- *Relative distance between the centroids of two regions* (ϖ): Let δ be the distance between the centroids of two regions:
  $\delta = \sqrt{(\bar{x}_1 - \bar{x}_2)^2 + (\bar{y}_1 - \bar{y}_2)^2 + (\bar{z}_1 - \bar{z}_2)^2}$.
  The relative distance between the centroids is defined by $\varpi = \frac{\delta}{\max(P_1, P_2)}$.
  Then, the relative distance between the centroids constraint is given by:
  if $|\frac{\varpi^d}{\varpi^m}| < (1 + thr_\varpi)$ and $|\frac{\varpi^d}{\varpi^m}| > (1 - thr_\varpi)$ then the regions are consistent, where d and m superscripts denote respectively data and model. The threshold for this constraint is thr_ϖ.
- *Ratio of surface area difference* (φ) is defined as:
  $\alpha = \frac{|A_1 - A_2|}{\max(A_1, A_2)}$.
  The surface area constraint is the following:
  If $|\frac{\alpha^d}{\alpha^m}| < (1+thr_\alpha)$ and $|\frac{\alpha^d}{\alpha^m}| > (1-thr_\alpha)$ *then the regions are consistent*, where d and m superscripts denote respectively data and model and err_α is the area threshold.

The result will be a set of possible matches. In the second stage, the surface type properties are used to select the best match. The model that matches the greatest number of surface types would be chosen.

7. EXPERIMENTAL RESULTS

In our experiments, we obtained intensity images of five objects: Bottle, Cup, Toy, Glass and Mustard Container (shown in Fig. 1). We built a database containing these objects as follows. First, we used stages one, two and three of our method to find the object regions in the images and to calculate those region properties which would be stored in the database. The database holds five records, each holding the corresponding object's properties obtained from the 3-D information.

To test our method, we used several intensity images for each object with slightly different viewing angle as shown in Fig. 6. Given each input image, the system obtains 3-D information using shape from shading, segment the object, compute the region properties and match those against the models in the database.

The segmentation produces the following results. The intensity images for the Bottle object were always segmented into three main regions, which were classified as ellipsoids (except for two regions which were classified as invalid types). The images for the Cup were generally segmented into two or three regions, corresponding to the handle, the liquid container and a noisy region obtained from a shaded

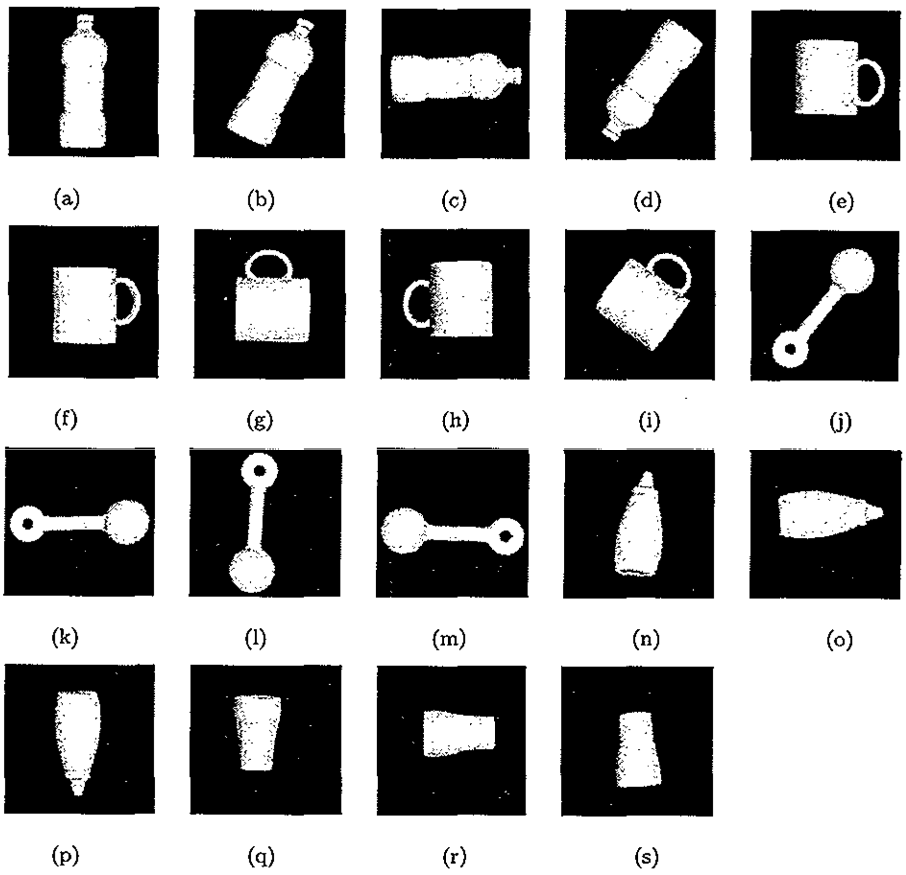

Fig. 6. Intensity images for (a) Bottle1, (b) Bottle2, (c) Bottle3, (d) Bottle4, (e) Cup1, (f) Cup2, (g) Cup3, (h) Cup4, (i) Cup5, (j) Toy1, (k) Toy2, (l) Toy3, (m) Toy4, (n) Must1, (o) Must2, (p) Must3, (q) Glass1, (r) Glass2, (s) Glass3.

surface of the cup. These regions were classified as ellipsoids, or as hyperboloids of one sheet. The Toy was always segmented into three major regions, which were consistently classified as ellipsoid, hyperboloid of one sheet and ellipsoid. The segmentation for the Mustard varied, producing two, three or four regions. The Glass produced only one region, an ellipsoid.

In Figs. 7 through 11, we summarize the properties obtained for each region in the object, namely 3-D surface area, perimeter, centroid (x, y, z) and the surface type. In each figure we give the object properties in the model (*), then the computed properties for the test data.

For the Bottle, Cup, and Toy, all the intensity images for different views were matched correctly. The Mustard was correctly identified in two of three trials. In the third case, the input (with four regions) failed to match the Mustard model (which had only two regions). Consequently, the input allowed two null matches, which were also satisfied by the Cup model. In addition, the Cup model matched more surface types, so the Mustard was classified as a Cup.

*Bottle1**	*Region 1*	*Region 2*	*Region 3*
area	4402.7	5491.2	5174.6
perimeter	270.0	287.0	270.0
centroid	(129.6,75.6,-0.6)	(128.7,140.8,-0.4)	(127.5,210.3,-0.6)
type	ellipsoid	ellipsoid	ellipsoid
Bottle2	*Region 1*	*Region 2*	*Region 3*
area	4657.5	5473.9	5290.3
perimeter	262.0	275.0	262.0
centroid	(162.9,73.6,-0.9)	(133.8,132.6,-0.6)	(102.2,195.4,-0.8)
type	ellipsoid	invalid	ellipsoid
Bottle3	*Region 1*	*Region 2*	*Region 3*
area	4347.4	5356.8	5332.3
perimeter	248.0	267.0	282.0
centroid	(180.1,114.5,-0.6)	(48.1,118.7,-0.5)	(117.0,116.8,-0.4)
type	ellipsoid	ellipsoid	ellipsoid
Bottle4	*Region 1*	*Region 2*	*Region 3*
area	5146.4	5422.0	4286.9
perimeter	240.0	248.0	234.0
centroid	(187.0,59.7,-0.4)	(147.4,116.2,-0.3)	(110.0,168.1,-0.4)
type	ellipsoid	invalid	ellipsoid

Fig. 7. Computed properties of Bottle.

*Cup1**	*Region 1*	*Region 2*
area	14838.4	1973.6
perimeter	462.0	260.0
centroid	(130.3,123.7,-1.2)	(208.7,134.1,-2.0)
type	ellipsoid	hyperbolid-1

Cup2	*Region 1*	*Region 2*
area	14581.7	1915.4
perimeter	462.0	227.0
centroid	(127.2,134.2,-1.0)	(203.7,126.6,-1.7)
type	ellipsoid	hyperbolid-1

Cup3	*Region 1*	*Region 2*	*Region 3*
area	2406.1	844.9	13834.8
perimeter	318.0	221.0	470.0
centroid	(126.3,58.8,-1.7)	(70.3,131.1,-1.7)	(135.4,138.4,-1.1)
type	ellipsoid	hyperbolid-1	ellipsoid

Cup4	*Region 1*	*Region 2*	*Region 3*
area	13476.8	763.9	2451.3
perimeter	475.0	212.0	294.0
centroid	(132.4,117.6,-1.3)	(76.4,125.6,-3.2)	(206.4,125.7,-2.3)
type	ellipsoid	hyperbolid-1	hyperbolid-1

Cup5	*Region 1*	*Region 2*
area	1865.9	14579.0
perimeter	277.0	398.0
centroid	(183.3,67.6,-1.1)	(129.2,127.4,-0.6)
type	ellipsoid	ellipsoid

Fig. 8. Computed properties of Cup.

Each of the Glass images was matched to several different models in addition to the correct model. Since each of the Glass images had only one region (an ellipsoid), only the region compactness constraint and surface type constraint were used. All the other models had two or more regions, and at least one of these could be matched

*Toy1**	*Region 1*	*Region 2*	*Region 3*
area	4067.7	2228.2	2767.0
perimeter	224.0	199.0	238.0
centroid	(164.9,74.1,0.5)	(117.6,134.7,0.6)	(69.8,194.8,0.6)
type	ellipsoid	hyperbolid-1	ellipsoid

Toy2	*Region 1*	*Region 2*	*Region 3*
area	4117.3	2523.2	2475.8
perimeter	208.0	218.0	236.0
centroid	(200.1,106.3,0.5)	(46.4,110.1,0.6)	(119.2,108.0,0.5)
type	ellipsoid	ellipsoid	hyperbolid-1

Toy3	*Region 1*	*Region 2*	*Region 3*
area	2749.0	2256.9	4103.9
perimeter	249.0	225.0	212.0
centroid	(128.2,47.6,0.8)	(123.1,125.9,0.7)	(118.3,202.4,0.7)
type	ellipsoid	hyperbolid-1	ellipsoid

Toy4	*Region 1*	*Region 2*	*Region 3*
area	4075.7	2852.3	2228.8
perimeter	208.0	254.0	219.0
centroid	(55.7,124.0,0.7)	(206.4,132.5,0.7)	(132.1,128.0,0.7)
type	ellipsoid	hyperbolid-1	ellipsoid

Fig. 9. Computed properties of Toy.

Must1	*Region 1*	*Region 2*	*Region 3*
area	669.8	650.4	10248.6
perimeter	92.0	115.0	398.0
centroid	(138.8,53.0,0.1)	(136.5,72.4,0.3)	(126.8,151.7,0.4)
type	hyperbolid-1	ellipsoid	ellipsoid

*Must2**	*Region 1*	*Region 2*
area	10716.4	705.9
perimeter	415.0	95.0
centroid	(116.1,111.0,0.2)	(208.5,107.0,0.1)
type	ellipsoid	ellipsoid

Must3	*Region 1*	*Region 2*	*Region 3*	*Region 4*
area	686.0	668.0	9607.2	617.1
perimeter	89.0	100.0	333.0	144.0
centroid	(79.1,69.1,0.2)	(93.9,82.1,-0.1)	(150.3,131.6,0.0)	(202.5,168.9,-0.5)
type	invalid	invalid	ellipsoid	hyperbolid-1

Fig. 10. Computed properties of Mustard.

to the Glass image's region. A region number constraint could help to solve this problem. In Fig. 11, we summarize the results of the matching.

We also analyzed the efficiency of the constraints used. In Fig. 12, we show how often a given constraint helped to prune the tree. The constraints were applied in the order shown in the table: compactness, relative distance between the centroids and relative difference between the 3-D areas. The most helpful constraint was region compactness which is the ratio of the 3-D surface area over perimeter squared. By using the 3-D data, we are able to obtain more information from each region as compared to the 2-D anlaysis. In addition, using 3-D data allows us to compute properties, such as surface types, which in 2-D could not be computed.

*Glass1**	*Region 1*
area	8795.8
perimeter	415.0
centroid	(111.4,112.9,0.3)
type	ellipsoid

Glass2	*Region 1*
area	8800.8
perimeter	376.0
centroid	(137.3,116.8,0.54)
type	ellipsoid

Glass3	*Region 1*
area	8988.9
perimeter	401.0
centroid	(127.2,147.3,0.54)
type	ellipsoid

Fig. 11. Computed properties of Glass.

Input	*Cup*	*Bottle*	*Mustard*	*Glass*	*Toy*
Cup1	√				
Cup2	√				
Cup3	√				
Cup4	√				
Bot1		√			
Bot2		√			
Bot3		√			
Bot4		√			
Toy1					√
Toy2					√
Toy3					√
Toy4					√
Glass1	√	√	√	√	
Glass2	√	√	√	√	√
Glass3	√	√	√	√	
Must1			√		
Must2			√		
Must3	√				

Fig. 12. Summary of results.

Input	*compactness*	*dist. cent.*	*diff. area*
Cup1	(26) 90%	(02) 7%	(01) 3%
Cup2	(29) 88%	(04) 12%	(00) 0%
Cup3	(73) 100%	(00) 0%	(00) 0%
Cup4	(19) 100%	(00) 0%	(00) 0%
Cup5	(33) 97%	(00) 0%	(01) 3%
Bot1	(49) 40%	(33) 58%	(02) 2%
Bot2	(32) 36%	(28) 32%	(28) 32%
Bot3	(28) 32%	(24) 28%	(34) 40%
Bot4	(30) 42%	(36) 50%	(06) 8%
Toy1	(34) 44%	(35) 45%	(08) 11%
Toy2	(37) 53%	(17) 24%	(16) 23%
Toy3	(51) 80%	(11) 17%	(02) 3%
Toy4	(39) 76%	(10) 20%	(02) 4%
Must1	(24) 36%	(27) 41%	(15) 23%
Must2	(16) 49%	(15) 45%	(02) 6%
Must3	(94) 65%	(30) 21%	(20) 14%
Glass1	(03) 100%	N.A.	N.A.
Glass2	(02) 100%	N.A.	N.A.
Glass3	(02) 100%	N.A.	N.A.

Fig. 13. Evaluation of constraint effectiveness.

8. DISCUSSION

We have presented a method that uses a shape from shading algorithm to perform 3-D object recognition for simple objects. This method extracts the 3-D information from a single intensity image, then segments the object into regions. After computing the properties of the regions, it compares the input object with the model objects in the database.

The experimental results are quite encouraging. The models are easy to build; we just use intensity images. The system computes object properties and store the properties in the database. The compactness constraint has been found to be the most helpful constraint in pruning the interpretation tree. Finally, we have shown that the estimated depth map can be used in object recognition.

REFERENCES

1. P. J. Besl, *Surfaces in Range Image Understanding*, Springer-Verlag, New York, 1988.
2. P. J. Besl and R. C. Jain, "Three-dimensional object recognition", *ACM Comput. Surv.* **17** (1985) 75–145.
3. T. Fan, *Describing and Recognizing 3-D Objects Using Surface Properties*, Springer-Verlag, New York, 1990.

4. W. E. L. Grimson and T. Lozano-Perez, "Model-based recognition and localization from sparse range or tactile data", *Int. J. Robot. Res.* **3** (1984) 3–35.
5. C. H. Lee and A. Rosenfeld, "Improved methods of estimating shape from shading using the light source coordinate system", *Artif. Intell.* **26** (1985) 125–143.
6. J. Levin, *QUISP: A Computer Processor for the Design of Quadratic-Surface Bodies*, Ph.D. Dissertation, Computer Science and Engineering Department, Rensselaer Polytechnic Institute, Troy, New York, University Microfilms International, 1980.
7. P. S. Tsai and M. Shah, "A simple shape from shading algorithm", *IEEE CVPR* **54** (1992) 734–736.

Ana Ortega received B.S. in computer science in 1994 (Summa Cum Laude and Honor's in the Major) from the University of Central Florida (UCF), Orlando.

During her senior year she participated in a program called Research Experience for Undergraduates (REU) where she learned about computer vision and participated in research. Ana was a member of the UCF's computer programming team and Upsilon Pi Epsilon (computer science Honor Society).

Currently, she is enrolled in the Computer Science masters program at UCF. Her interests are computer vision and databases. She is currently working for a company called Visteon Corporation, developing client-server applications for the health care industry.

Mubarak Shah received his B.E. degree in 1979 in electronics from Dawood College of Engineering and Technology, Karachi, Pakistan, and was awarded a five year Quaid-e-Azam (Father of Nation) scholarship for his Ph.D. He spent 1980 at Philips International Institute of Technology, Eindhoven, The Netherlands, where he completed E.D.E. diploma. Dr. Shah received his M.S. and Ph.D. degrees both in computer engineering from Wayne State University, Detroit, Michigan, respectively in 1982 and 1986. Since 1986 he has been with the University of Central Florida, where he is currently Professor of Computer Science, and the Director of Computer Vision lab.

Dr. Shah has served as a project director for the national site for REU, Research Experience for Undergraduates in Computer Vision, funded by the National Science Foundation for the last ten years. This year he received another NSF REU grant for three more years. Approximately one hundred undergraduate students from different schools in Florida have participated in this program during the last ten years. The REU participants have co-authored 50 research papers.

Dr. Shah has received *Teaching Incentive Program* (TIP) Award, *IEEE Distinguished Visitors Program Speaker* Award, *IEEE Outstanding Engineering Educator Award,* and two *TOKTEN* (in 1995 and 1997) awards by UNDP.

Dr. Shah has published one book (*Motion-Based Recognition*, Kluwer Academic Publishers, with Ramesh Jain), and over 50 research papers in refereed journals and conferences on topics including optical flow, structure from motion, motion-based recognition, gesture recognition, activity recognition, lip-reading, snakes, object recognition, edge and contour detection, multisensor fusion, shape from shading and stereo, and hardware algorithms for computer vision.

Dr. Shah is an associate editor of *IEEE Transaction on PAMI*, and *Pattern Recognition*, and serves as a referee for several journals. He has served on the program committees, and chaired sessions of several conferences. He was an organizer of *Workshop on Recent Advances in Computer Vision*, held in Karachi, Pakistan, Jan 1–2, 1998, and *Workshop on Computer Vision* held in Islamabad, Pakistan, January 3–5, 1995. Both workshops were supported by grants from NSF.

Photograph of Dr. Shah is unavailable.