Pattern Recognition, Vol. 22, No. 1, pp. 49–64, 1989.
Printed in Great Britain.

0031-3203/89 $3.00 + .00
Pergamon Press plc
Pattern Recognition Society

RECOGNITION OF 3-D OBJECTS IN RANGE IMAGES USING A BUTTERFLY MULTIPROCESSOR*

BIR BHANU† and LAWRENCE A. NUTTALL‡

† Honeywell Systems and Research Center, 3660 Technology Drive, Minneapolis, MN 55418, U.S.A.

and

‡ Department of Computer Science, University of Utah, Salt Lake City, UT 84112, U.S.A.

(*Received* 29 *September* 1987; *in revised form* 9 *May* 1988; *received for publication* 23 *May* 1988)

Abstract—The recent advent of Multiple Instruction Multiple Data (MIMD) architectures together with the potentially attractive application of range images for object recognition, motivated the development of a successful goal-directed 3-D object recognition system on a 18 node Butterfly multiprocessor. This system, which combines the use of range images, multiprocessing, and rule-based control in a unique manner, provides several new insights and data points into these research areas.

Several topics pertinent to current research were explored. First, a new method of surface characterization using a curvature graph was proposed and tested. It was determined that by jointly using information provided by the principal curvatures, the potential exists for uniquely identifying a larger variety of surfaces than has heretofore been accomplished. Second, a 3-D surface-type data representation, coupled with the depth information available in range images, was used to correctly recognize and interpret occluded scenes. Finally, it was determined that both multiprocessing and a rule-guided/goal-directed search can be successfully combined in an object recognition system. Multiprocessing was employed both at the object level and within objects. This enabled the achievement of near linear speedups for scenes containing fewer objects than the number of available processors.

3-D object recognition Butterfly multiprocessor Curvature graphs
Surface characterization Range images

1. INTRODUCTION

Object recognition approaches in computer vision can conceptually be classified into two categories. The first, or traditional approach, involves the use of statistical and structural techniques. Over the years this approach, by itself, has proven to be inadequate in handling some of the more difficult real world problems where noise and improper illumination exist, and the problem domain has not been constrained to well-defined geometric objects. The second approach attempts to overcome these problems in much the same way a human does, through the use of contextual information, experience, or expert knowledge. The advantages derived from the second approach, however, typically come at the expense of speed.

For many computer vision applications, real time processing are mandatory. For nearly a decade pipelined or Single Instruction Multiple Data (SIMD) image processors have been successfully used to overcome many of the speed constraints associated with the large volumes of data found in image processing. They have been successful in image processing because many low-level algorithms and enhancement techniques can be applied uniformly across an entire image. On the other hand, many of the higher level computer vision tasks such as image understanding or object recognition, depend on algorithms which are local in nature and contain logic, pixel addressing, and control sequencing which are not easily performed on typical image processors. Such tasks are more naturally suited to the use of a Multiple Instruction Multiple Data (MIMD) machine. With such architectures now becoming a reality, the next step seems obvious; to test whether the advantages of more flexible control sequencing, and the contextual and expert knowledge utilized by high-level vision algorithms, can in fact be gained without sacrificing speed.

The problem domain chosen for this research (i.e. object recognition in range images) is one that stands to gain much from such an approach. Depth information provided by 3-D range images, and the utilization of contextual knowledge and rule-based control are particularly useful in resolving some of the problems associated with 3-D object recognition tasks. This project implemented a goal-directed object recognition system on a Butterfly multiprocessor in order to investigate some of the issues mentioned above. Section 2 provides a brief overview of this system. As

* This work was supported in part by NSF grants DCR-8506393, CCR-8704778 and DMC-8502115 at the University of Utah. The support rendered by Perceptics Corporation is sincerely appreciated.

part of this project, two primary areas of research were studied. The first relates to surface characterization via curvature, and is discussed in Section 3. The results of object recognition and occlusion are also discussed. The second is presented in Section 4 and reviews the methodology and results of the multiprocessor implementation of the object recognition system.

2. SYSTEM DEFINITION

The purpose of the object recognition system in this research was to conduct a goal directed search in order to identify all objects in a range image matching a specified goal.

Fig. 1. Image scene_3: White Scanner data.

2.1. *System hardware*

The image processing system developed for this research was implemented on an 18 node BBN Butterfly multiprocessor. Each node consists of an MC68020 processor, an MC68881 floating-point co-processor, memory, and an interface to the Butterfly switch. Sixteen of these nodes have 1 Mbyte of on-board memory, while the other two have 4 Mbytes. The Butterfly is connected to a VAX 11/785 via two serial lines and an ethernet interface. The serial lines are used primarily for booting the Butterfly, while normal access occurs over the ethernet. Because the Butterfly does not have a file server, executable code as well as images were downloaded over the ethernet.

2.2. *System input, output, and data structures*

There were two inputs to the system. (1) A three-dimensional range image containing one or more objects. Each object was composed of surfaces from the following types: cones, interior cones, cylinders, troughs, spheres, dishes and planes. There were no restrictions as to object size and orientation, in fact the system took advantage of scale information to assist in the resolution of occluded objects. This system was designed to accept objects which did not have concave boundaries. (2) The second system input was a high-level description of the goal object to be located in the input image. As output the system returned the description and location of all objects in the input image, as well as identifying those objects which matched the goal.

2.3. *Image data*

The broad scope of this study necessitated several different types of images. The first image, scene_3 (Fig. 1), is an actual range image containing 3 objects, a sphere, a cylinder, and a cube. It was taken from the Utah Range Database,[4] a collection of range images created by a Technical Arts 3-D White Scanner Model 100-A. Range data returned by the White Scanner is in the form of x, y and z coordinate values relative to a world coordinate system. Because the White Scanner derives range information via triangulation, the z or range data is not orthogonal to the scanning plane. In order to facilitate processing, a raster-formatted image was created from this data. Essentially, the x and y information was discarded and the image was displayed according to scan lines, using the nonorthogonal z values. The effect of this nonorthogonality will be discussed in Section 3.

In order to test the object recognition algorithms on a larger variety of composite objects than those available from the Utah Database, five 16 bit, 512 × 512, synthetic range images were also created. These images represent range data sampled at regular x and y intervals relative to a fixed x, y, and z coordinate system. The pixel values are range values or distances from the x-y plane. These images differ from those generated by the White Scanner in two ways: first, they are regularly sampled in x and y, and second, the plane of this raster scan is perpendicular to the z axis. As required by this project, each image contained multiple objects at arbitrary orientation. Each object was composed of various combinations of the basic curved surface shapes: cones, spheres, cylinders, and planes. Objects containing different sizes of all of these surface types, as well as similar surface types at different radii or curvature were included.

Four of these five images contain 1, 3, 5 and 7 composite objects respectively, and are referred to throughout this paper as images R1–R7. Images R5, and R7 are shown in Figs 2 and 3. These images contain different numbers of the same objects placed at different locations in the image. The same objects were used primarily to provide a stable level of processing complexity when analyzing the multi-processing performance of the system.

Proceeding from the upper left corner to the lower right corner of image R7 (Fig. 3), the seven objects in images R1–R7 can be described as follows. (1) A closed can consisting of a cylindrical surface of radius 30, and a planar bottom. (2) An open can consisting of a cylindrical surface of radius 40, rotated so that the trough-like interior of the can can be seen. (3) A cone of radius 40 also rotated so that the interior of the cone is visible. (4) A sphere of radius 40. (5) A

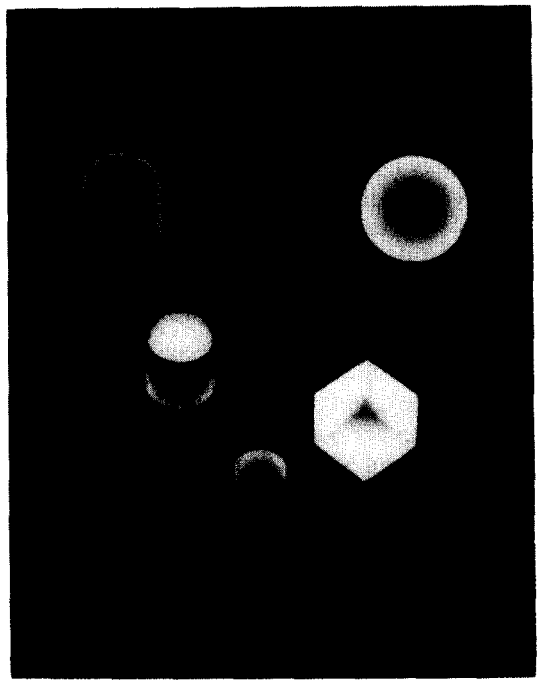

Fig. 2. Image R5: synthetic data.

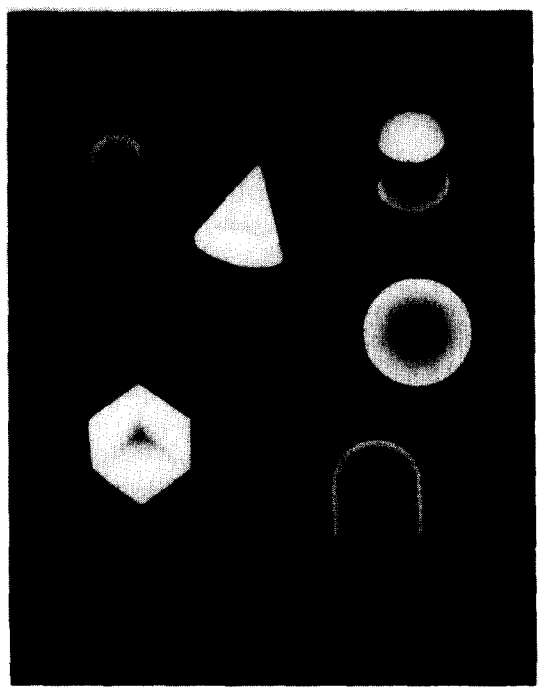

Fig. 3. Image R7: synthetic data.

sphere of radius 60. (6) A cube rotated so that 3 planar faces are visible. (7) A "dome can" consisting of a spherical surface resting on a cylindrical surface, both of radius 50. Image R1 consists of (6), image R3 consists of (1), (5) and (6), and image R5 consists of (1), (2), (5), (6) and (7).

The last of the synethetic images, "occlude", shown in Fig. 4, contains two cylinders. The larger cylinder of radius 30 was rotated so that the interior of the cylinder was visible. This object was then occluded by a smaller cylinder of radius 15.

2.4. *The object data structure*

One of the main goals of computer vision is to take an input image and transform it in some way so that we can understand the real world it portrays. This portrayal of the real world exists in the data representation we choose. Hence, it is one of the first issues which must be resolved when designing a vision system such as the object recognition system developed for this research.

In general, a 3-D data representation should satisfy the following considerations. (1) It should be invariant to translation, rotation, and scale. In other words, it should be view independent. Although high level object descriptions should not depend on scale, it may still be possible to capitalize on size or scale differences as a further discriminator between objects. (2) The model should be as memory conservative as possible. (3) The data representation should facilitate the matching of randomly oriented objects. (4) It should maximize the advantages and information derived from range images. In particular, range data provides depth information invaluable in interpreting occluded objects. Data representations must include this adjacency and edge-type information.

Three-dimensional object representations are typically classified into three categories: surface or boundary descriptions; sweep; and volumetric representations. Besl and Jain[2] give an excellent critique

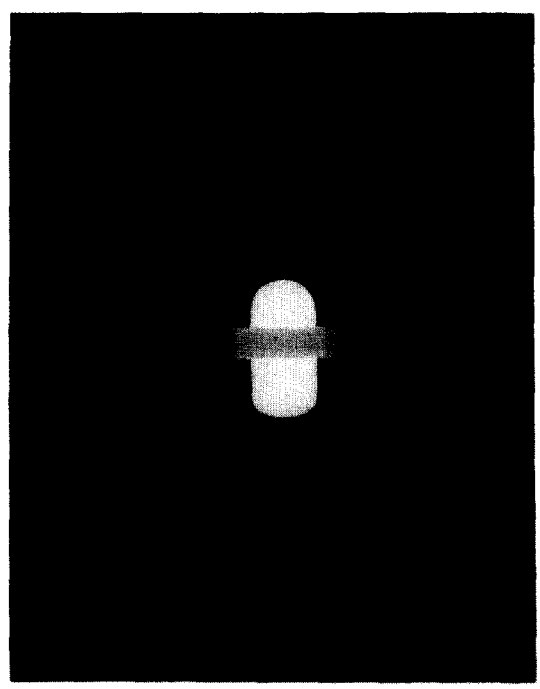

Fig. 4. Image occlude: synthetic data.

of 3-D object representations. They point out that many of the volumetric representations (e.g. CSG) and the algorithms required to compute them, are often memory and compute intensive. In addition, there are many surfaces which are not easily defined in terms of a closed form formulae, resulting in nontrivial descriptions for complex objects. Generally speaking, these same objections apply to the "sweep" representations (e.g. generalized cylinders). Surface representations are often given at a higher level of abstraction[7] and were therefore considered the best choice for the application pursued in this research.

The primary data structure used by this system, therefore, was the "object", similar in concept to the "winged-edge" data structure.[1] In keeping with the concept of a surface representation, objects were merely lists of surfaces. Each surface had associated with it a surface type, a list of adjacent surfaces, and a description of the connecting edges. In addition to this surface list, each object contained several Boolean types indicating the processing state of the object.

2.5. *Multiprocessor control*

Similar to several of the rule-based vision systems developed by other researchers,[5,6] this system incorporates several sets of control and high-level processing rules to direct multiple processors in an optimized search for a goal object. Figure 5 displays the general processing steps performed on each object in order to accurately identify it. As indicated in the figure, matching was performed after various stages of processing. Depending on its state of completion, an object's surface types, number of surfaces, surface adjacencies, and connecting edge types were compared with the goal object during the matching process. While it is evident that these processing steps were performed sequentially, parallelism was achieved by performing them simultaneously on each object, as well as by subdividing some of these tasks among the multiple processors. Results from the multiprocessing portion of this research are discussed in Section 4.

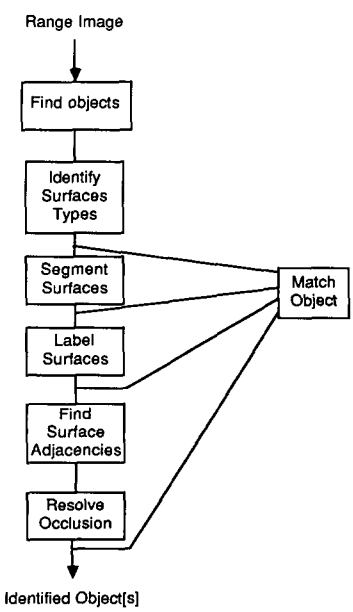

Fig. 5. Basic processing flow. In this figure lines drawn to the "match object" box merely indicate that matching is performed after each of the four processing steps indicated. Matching does not alter the sequential flow of processing, but merely allows prioritization of objects, so that these processing steps are performed on the most promising objects first.

As depicted in Fig. 6, the system can conceptually be viewed as data on which to operate, specific work to be done, and multiple processors to perform the work. All processors are equally capable of performing all processing, and may either remove or place data or work on the data and job queues. The word "queue" is used loosely here. Work or data may be placed at either the front or back of these control structures allowing them to be used as either a queue or a stack.

The system control strategy is quite simple. Basically, each processor possesses an identical set of control rules which directs it in a search for work to do. Work can be found in one of two data queues or one of two job queues. The job queues contain specific work to be done, (e.g. calculate curvature measures for row 10 of object 2), while the data queues contain image objects which are awaiting the next stage of processing.

The fact that this study implements a goal-directed recognition system, strongly suggests the use of a top down approach using backward chaining rules. Backward chaining requires a reasonably complex interpretation and control scheme typically implemented within special purpose "expert system" languages. Unfortunately, none of these languages, including Lisp (the language upon which most of these higher level languages are based), were available on the Butterfly. Consequently the goal-directed, or backward chaining, nature of this system was accomplished via forward chaining "if then else" rules which prioritized the data and work to be done. Because the "Butterfly" allows the insertion of data only at the beginning and end of queues, four queues were used in this scheme to provide essentially eight priority levels. After each stage of processing, objects were placed at either the front or the back of either the low or high priority data queues. When all processing had been completed on an object, it was either announced as a match or discarded.

When seeking work, processors were directed by the control rules to first check the high priority job queue. The idea was that the most important work to be done was that which had already been identified. If no specific high priority jobs were available, a check for data (i.e. image objects) on the high priority data queue was made. Both work and data were always extracted from the front of the queues. The processing rules were then applied to the object in order to further identify work to be done. On completion of some of the processing tasks, the object was marked,

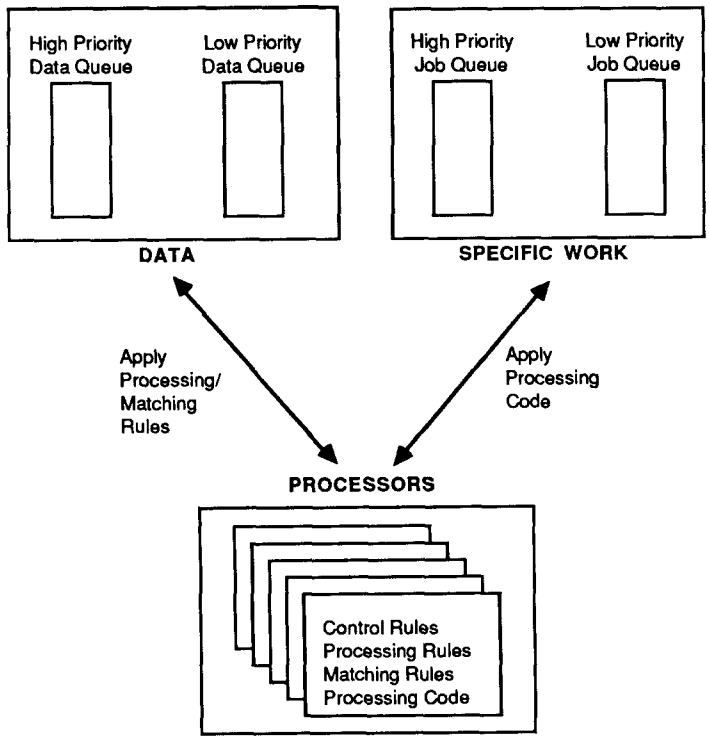

Fig. 6. System components.

indicating that an evaluation of its "match likelihood" was needed. This flag caused one of the processing rules to fire, which in turn caused the matching rules to be applied to the object.

In contrast to the control rules, the processing and matching rules were applied directly to an object, and assessed that object's state of processing or match likelihood. Many of the jobs to be performed on objects were easily performed in parallel, such as calculating curvature values. Assistance for completing such jobs was requested from other processors by placing a specific task on a job queue. So as to avoid deadlock, results from these tasks were collected by a separate task also placed on the job queue. The task responsible for collecting results also placed the completed data object back on the appropriate queue, depending on its match likelihood. System processing was terminated on the occurrence of one of two events: (1) when the first match had been found, or (2) after all objects had been processed and identified. The second event was recognized when there were no further jobs to be done, and no more data on the data queues. In both events, objects matching the specified goal were identified.

3. SURFACE CHARACTERIZATION VIA CURVATURE GRAPH

The success of the object data structure described in Section 2 depends almost entirely on the method selected for characterizing surfaces. Two of the criteria for the selection of a data representation enumerated in that section suggest that the chosen representation should maximize information derived from range images, and also be invariant to translation and rotation. Surface curvature can readily be obtained from range data, and satisfies both of these criteria. Because it is an intrinsic property of objects, curvature is a natural choice to be used for maching and object recognition, and is becoming an increasingly popular method of surface classification.[2,3] For these reasons a surface data representation with curvature as the primary descriptor was used for this project. As indicated, curvature has been used by many researchers for the purpose of surface classification. Some of these approaches used single curvature measures, others used combinations of curvature measures, and still others used combinations of the signs of curvatures. At best, 8 surface types[2] were identifiable: pit surfaces, minimal surfaces, ridge surfaces, saddle ridges, peak surfaces, flat surfaces, valley surfaces, and saddle valleys.

In this research, a significant variation from these approaches was explored. The motivation for doing so came when realizing that conical surfaces cannot be satisfactorily recognized using only the signs of the Gaussian and mean curvatures. While analyzing alternate uses of curvature for this purpose, it was realized that (at least theoretically), a complete mapping of all surface types should be achievable, not just the limited classifications accomplished to date. In this paper, the term surface type generally refers to

the nature of the surface at any point on that surface. In discrete range images, points are pixels, and the description of the surface at that pixel is limited by the spatial resolution of the image. Large surface patches whose pixels have surface types with similar characteristics are generally recognized at a high level as one surface. Discovering how to identify those characteristics common to such surfaces is an active field of research.

3.1. *Curvature graphs*

In this study, surface types were determined according to an estimation of the principal curvatures for each pixel on the surface of interest. It was recognized by Fan *et al.*[3] that the magnitude and orientation of the principal curvatures completely and uniquely define a surface. Although their work concentrated on the identification of jump boundaries, folds, and ridge lines, this concept can be extended to enable the identification of a continuum of surface types. This is possible by preserving the information inherent in the principal curvatures rather than arithmetically combining them as is done to obtain the Gaussian and mean curvatures. Specifically, each pixel's surface type is determined according to the location of its principal curvature values on a graph, where the principal curvatures (i.e. the minimum and maximum curvatures) are the coordinate axis of that graph (see Fig. 7). Throughout this paper this graph will be referred to as the curvature graph. By definition, the maximum curvature cannot be less than the minimum curvature. Surface types will, therefore, not be found in the shaded area to the right of and below the 45° diagonal line as shown in Fig. 7. Every position above this 45° diagonal, however, represents a unique set of principal curvatures and hence can be considered a different surface type. Pixels with the same principal curvatures have identical surface types. It becomes clear then, that there is in reality a potential for representing a continuum of surface types. In fact it can be seen from Fig. 7, that the 8 types identifiable using the combinations of the Gaussian and mean curvatures are subsets of the curvature space represented on the curvature graph. These 8 surface types define large areas and are indicated in Fig. 7, as are also the surface types which were of particular interest to this research. Surface patches consisting of many pixels whose surface types have some common characteristic, are often perceived as a single "surface". Some familiar surfaces, such as spheres and cylinders, consist of pixels having identical principal curvatures or surface types. Principal curvatures from such surfaces map to a single location on the curvature graph and are therefore easily identified. Other surfaces, such as cones, have pixels whose surface types are not exactly the same, but which still form an identifiable pattern on the curvature graph.

It is conceivable that surface types from an entire object may create a pattern on the curvature graph which is unique for a particular application. To be specific, spherical surfaces ideally have minimum and

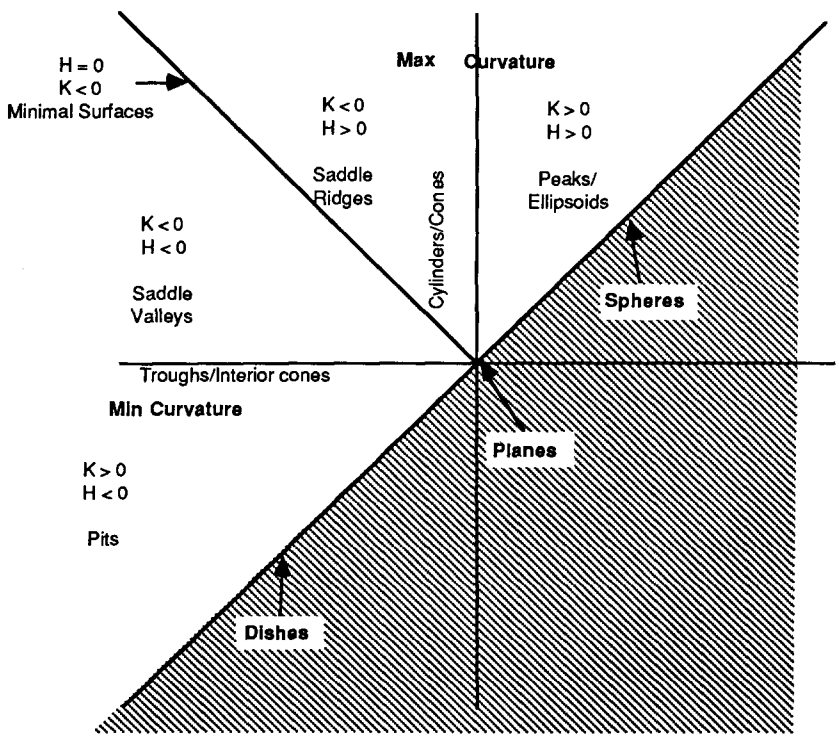

Fig. 7. Curvature graph: regions identified by signs of the Gaussian and mean curvatures (K—Gaussian Curvature, H—Mean Curvature), as well as basic surface types are labeled.

maximum curvatures which are equal. Points from such surfaces should plot directly on the 45° diagonal. Cylinders have minimum curvatures of 0 and a single maximum curvature. Cylindrical surfaces will therefore be found at a single point on the y axis corresponding to the value of its maximum curvature. Cones will also be located on the y axis, but because each cone's maximum curvature ranges from infinity to the curvature at its base, points plotted from this surface will form a pattern of points spread along the y axis, rather than being located at a single point. Reflecting plots of these surfaces around the 135° diagonal yields the expected plots for the inverses of these surface types (e.g. dishes, troughs, and interior cones). Planes of course ideally have minimum and maximum curvatures equal to zero and will therefore be plotted at the origin. Because curvature is inversely proportional to the radius of curvature, even surfaces of similar types yet different radii will occupy different locations on the curvature graph and should therefore be distinguishable. The ability to resolve surfaces on the graph depends, of course, on the spatial resolution of the image and on the effects of noise and the quantization of the range data.

Figures 8 and 9 show plots of principal curvatures taken from all of the surfaces represented in the synthetic image R7. For the most part surface types plotted exactly where expected and were easily distinguishable. In particular, notice the clear distinction between the cylindrical and conical surfaces. The only surface which was not easily identified was the interior surface of the cone. Because only a small interior portion near the base of the cone was visible, there were not enough pixels available to distinguish it from a trough like surface. Without the contextual information provided by neighboring surfaces, it is likely that the human visual system would also be unable to clearly identify this surface. Although the planar points from image R7 mapped to an identifiable region on the curvature graph, they were slightly displaced from the origin (Fig. 8). Figure 10, however, shows that planar points from the real data in scene_3, plotted directly at the origin as expected and the spherical and planar surfaces clearly segmented on the curvature graph. An initial investigation into this discrepancy indicated that the planar points in the synthetic data were abnormally effected by quantization during the synthetic data generation process. Figure 11 shows the curvature graph of the soda pop can. It was sitting on its top so that three different surface types were visible. A cylindrical surface of course makes up the majority of the can. The bottom of the can is an inverted sphere or dish, and there exists a narrow spherical-type surface between the

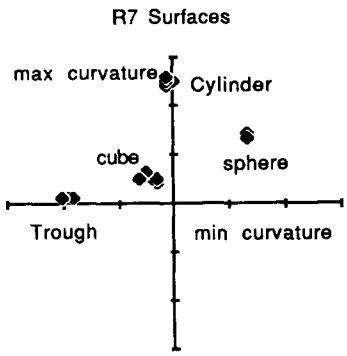

Fig. 8. Curvature clusters from image R7. Cylinder, sphere, cube, and trough.

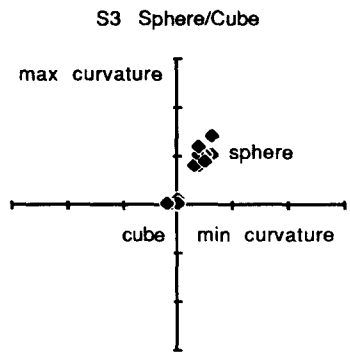

Fig. 10. Curvature graph results for sphere and cube in scene_3.

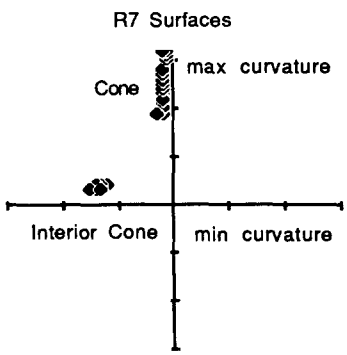

Fig. 9. Curvature clusters from image R7. Cone and interior cone.

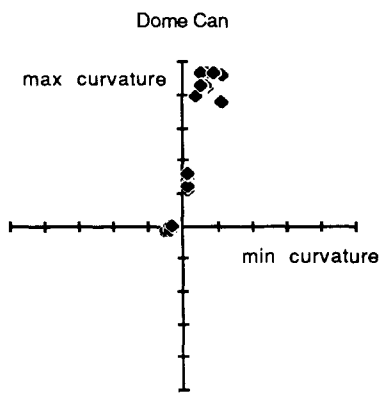

Fig. 11. Curvature graph results for pop can in scene_3. Shows three surfaces: sphere, cylinder, and trough.

cylindrical side and the bottom dish of the can. All three surface types on the can are clearly segmented on the curvature graphs.

It should be mentioned that curvature graphs do not give information as to how many pixels were plotted at each location on the graph. This is evident from the spherical surface plotted for the can, which at first glance seems to suggest that the spherical surface had the most points. This surface in fact was the smallest, and its proportionately large number of near edge pixels account for the less compact cluster. The cylindrical surface, on the other hand, actually contains the greatest number of surface points. Because these all had very nearly the same principal curvatures, the cluster on the graph is much more compact. A better indication as to representative surface sizes is obtained from the histogram analysis discussed in Section 3.2.

It is also evident that while the clusters from scene_3 were easily distinguishable and generally where they belonged, the nonorthogonality of this range data, as expected, did cause some distortions. In particular spheres did not lie on the 45° diagonal line as one would hope, but instead plotted somewhere between the diagonal and the min = 0 or y axis.

3.2. *Use of curvature graph for edge detection*

In addition to determining the usefulness of the curvature graph for surface characterization, this research also investigated the utility of the graph for edge detection and identification, indicating whether edges are jump or interior, concave or convex. Essentially jump edges are indicated by large discontinuities in the range data. These are typically found between objects and the background, or between occluding surfaces. One would expect, therefore, that relatively large curvature values should also be the characteristic of jump edges. In this paper interior edges refer to edges between two continuously connected surfaces on the same object, and are typically indicated by local extrema of curvature. The terms convex and concave define the direction of the surface change as one traverses the edge from one surface to the next. Convex edges are typically distinguished by large positive curvatures, while concave edges are identified by large negative curvatures. Because they give directional information, it is convenient to use them when describing interior edges. In the vicinity of jump edges, curvature values yield information about the nature of the discontinuity at the edge rather than the surface itself. Zero-crossings occur in the curvature at jump boundaries. The sign of the curvature on either side of these zero-crossings is useful in resolving occlusion problems.

One can actually think of a straight convex edge as a continuous extension of a cylinder, and a straight concave edge as the continuous extension of a trough or interior cylinder. The range of cylinders of course have radii of curvature varying from near zero to infinity. The smaller the radius, the more the cylinder looks like an edge. Straight interior and jump edges can be defined to exist somewhere along that continuum. Such edges therefore lie along the x and y axis of the curvature graphs. One expects jump edges to have curvature values well in excess of those found on normal surfaces. Exactly where the distinction between a surface and an edge lies is a matter of definition and may depend on the context. Typically nonstraight edges should also have relatively high curvature values but will not lie on the coordinate axis.

With one exception, all edges associated with the objects used in this study were clearly distinguishable from normal surface pixels by their extreme principal curvature values. The "dome can" in image R7, however, demonstrated the fact that some edges or boundaries lie on smooth transitions between surfaces and are not accompanied by large curvature discontinuities. As with all surface types, however, such boundaries occupy a distinct region on the curvature graph and should therefore be identifiable. This is shown in Fig. 12 which displays the plot of curvatures as one traverses from the spherical to the cylindrical surface on the dome can. For this application it was a simple matter to define pixels falling within the portion of graph between these two surfaces as edge or boundary points. This fact enabled the segmentation of the "dome can" for which no discontinuities in range, derivatives, or curvature values existed on the boundary between the spherical and cylindrical surfaces.

3.3. *Curvature histogramming*

As mentioned previously, the difficulties anticipated in applying the curvature graph concept are associated with digitization, quantization, and noise. To some degree, smoothing curvature values helped to minimize these problems, but at the same time somewhat altered the pattern for the conical surface on the curvature graph. The fact that all surface types can visually be distinguished when viewing the curvature graph, suggested that a statistical approach using well

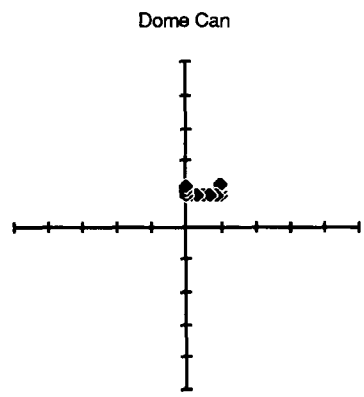

Fig. 12. This figure shows the smooth transition of curvature values from a spherical to a cylindrical surface on the dome can.

established pattern recognition techniques may reduce the adverse effects of noise, etc., as well as make the recognition of many complex surfaces possible. Ideally such statistical approaches should include the quantity information, that is, how many pixels on a surface are represented at each location on the curvature graph. A three dimensional curvature graph or space could be used where the three variables would be: the maximum curvature, minimum curvature, and the number of pixels possessing these curvatures. Clustering and statistical techniques could then be used to identify surfaces, (or possibly entire objects), in this 3-D space. Because the surfaces used in this study were easily identifiable without such an involved analysis, a simple histogram approach was used.

The histogram approach involved mapping the 2D information in the curvature graph into a one dimensional histogram. Although this method loses spatial resolution on the graph, it does take advantage of the quantity information useful in identifying cones. This mapping takes place by assigning each pixel on an object's surface a single value according to its location on the curvature graph. Future use of the term mapped-curvature in this paper refers to this one dimensional value. Specifically, mapped-curvature = base_value × distance, where the base_value is determined by the region on the graph in which the principal curvatures lie. Distance is the distance from the plotted point on the graph to the origin of the graph. Because the object recognition portion of this research required only that a limited number of surface types be distinguished, defining broad regions corresponding to these required surface types proved to be adequate for this work. All range pixels whose curvature graphs were greater than a specified distance from the origin, were considered to be edge pixels and were not utilized for determining the classification of surfaces.

Once mapped-curvature values had been assigned over the entire object or region of interest, the surface types within that region were identified by examining the histogram of these values (see Fig. 13). The histogram was divided into regions corresponding to the different surface types. The broad categories required for this project were defined approximately over the following mapped-curvature values: spheres 0–50; cylinders 51–100; planes 101–150; troughs 151–200; and dishes 201–250. The visible area of each surface type represented in the region is derived by integrating the histogram over the correct value range.

Figure 14 shows the actual histogram of the mapped-curvature values derived from the pop can in scene_3 (Fig. 1). The three surface types, spherical, dish, and cylindrical are very distinct. In addition, relative surface areas can be derived from the cumulative distribution function. Even a visual inspection of the histogram indicates that the cylindrical surface of the can is two or three times as large as the dish portion, which is also two or three times as large as the spherical surface.

Table 1 summarizes the total number of pixels, average mapped-curvature values, and standard deviations found for each section of the histograms derived for each object in image R7. These sections are labeled according to the surface types to which they correspond. The *'s indicate for each object which of the surfaces are significant or valid. As evidenced in Table 1, all surfaces of all objects were correctly identified with the exception of the interior of the cone which was classified as a trough or the interior of a cylinder. This was due to the fact that not enough of the interior portion of the cone was visible.

This table also demonstrates that resolution between spheres, cones, and cylinders of different radii is possible. Because curvature is inversely proportional to the radius of curvature, however, the larger the radii, the less distinguishable were surfaces of different sizes. Specifically we note that the average mapped-curvature values for the spheres of radii 40, 50, and 60 are 19, 14, and 12 respectively. Similarly, values for the cylinders of radii 30, 40, and 60 are 77, 82, and 90. As expected, we see that the difference between curvatures for surfaces with larger radii is smaller than the difference between surfaces of smaller radii. Without testing on a larger variety of images these results demonstrate that as a minimum, both cylindrical and spherical shaped surfaces are distinguishable into at least 3 different size groups using

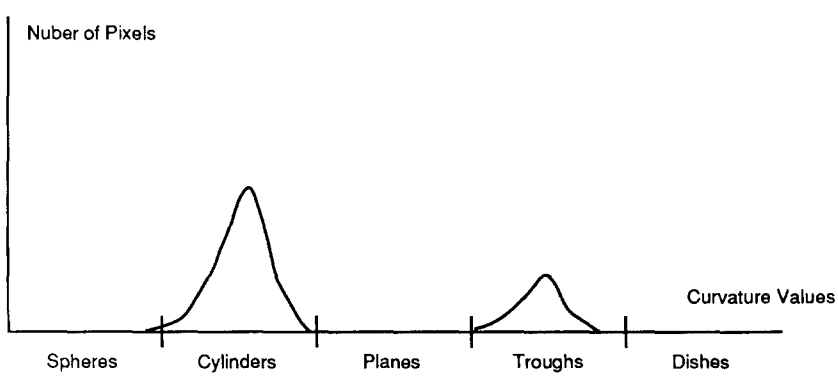

Fig. 13. Simulated mapped-curvature histogram for an open-ended can.

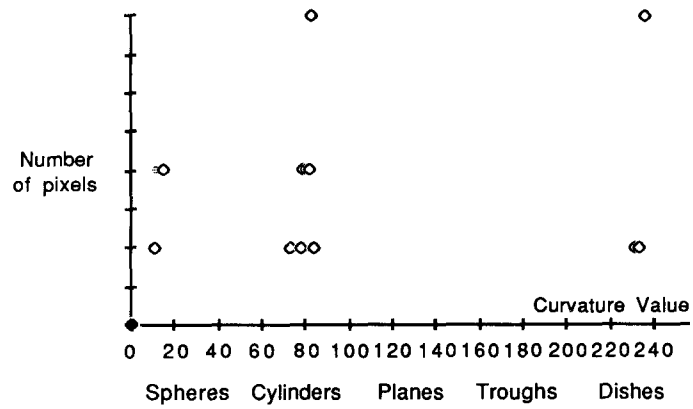

Fig. 14. Histogram of mapped-curvature values on the pop can in scene_3.

Table 1. Histogram results of mapped curvature values taken over objects in image R7

Can1				Cone			
Surface Type	Tot. Pixels	Ave. Val.	S. D.	Surface Type	Tot. Pixels	Ave. Val.	S. D.
Sphere	46	33	4	Sphere	0	0	0
Cylinder	*2124	77	5	Cylinder	*2040	82	8
Plane	*1159	119	23	Plane	254	120	16
Trough	26	155	2	Trough	*906	153	5
Dish	0	0	0	Dish	56	223	4
Can2				Small sphere			
Surface Type	Tot. Pixels	Ave. Val.	S. D.	Surface Type	Tot. Pixels	Ave. Val.	S. D.
Sphere	0	0	0	Sphere	*3269	19	2
Cylinder	*2759	80	4	Cylinder	0	0	0
Plane	174	135	12	Plane	80	139	1
Trough	*2304	157	5	Trough	88	139	1
Dish	0	0	0	Dish	0	0	0
Large sphere				Cube			
Surface Type	Tot. Pixels	Ave. Val.	S. D.	Surface Type	Tot. Pixels	Ave. Val.	S. D.
Sphere	*8573	12	2	Sphere	2	42	0
Cylinder	20	53	2	Cylinder	632	26	7
Plane	0	0	0	Plane	*5978	112	21
Dish	0	0	0	Dish	0	0	0
Dome can							
Surface Type	Tot. Pixels	Ave. Val.	S. D.	Surface Type	Tot. Pixels	Ave. Val.	S. D.
Sphere	*2787	14	2				
Cylinder	*4857	90	0				
Plane	158	144	0				
Trough	164	145	2				
Dish	0	0	0				

the histogramming approach. Because of the greater resolution at smaller radii, it appears the actual number of differentiable size groupings will in fact be much larger.

3.4. *Object recognition and occlusion*

The successful application of the curvature graph approach described in the previous section enabled the correct classification of surface types and edges in the object recognition system. As shown by the processing steps outlined in Fig. 5, once an object's surface types had been identified, segmentation and complete labeling was possible. The classifier output, showing the final results of classifying all objects in image R7 (Fig. 3), is shown in Fig. 15. The center of each object's bounding rectangle is shown as a coordinate pair, and listed with each surface are two numbers. The first is the identification of the surface type according to the following labels: 1—sphere; 2—dish; 3—cylinder; 4—trough; 5—cone; 6—interior cone; and 7—plane. The second number is the average mapped-curvature value for the surface. This number can be directly mapped to the radius of curvature.

The processing steps described thus far were sufficient to successfully identify all objects in each image with the exception of image "occlude" shown in Fig. 4. The last two processing steps shown in Fig. 5 are necessary to handle occlusion. In the following a simple example is given which illustrates the basic

```
There are 7 objects in the image:

object 1, at (56,56), has 2 visible surfaces:
        surf 1;   3      77
        surf 2;   7      120
object 2, at (64, 400), has 2 visible surfaces:
        surf 1;   4      158
        surf 2;   3      82
object 3, at (112, 200), has 2 visible surfaces:
        surf 1;   5      77
        surf 2;   4      155
object 4, at (248, 400), has 1 visible surfaces:
        surf 1;   1      12
object 5, at (256, 256), has 1 visible surfaces:
        surf 1;   1      19
object 6, at (368, 80), has 3 visible surfaces:
        surf 1;   7      122
        surf 2;   7      117
        surf 3;   7      103
object 7, at (424, 352), has 2 visible surfaces:
        surf 1;   1      14
        surf 2;   3      90
```

Fig. 15. Printed results after identifying all 7 objects in image R7. The two figures associated with each surface indicate their surface type and average curvature value.

concepts involved.

Figure 16 shows a line drawing of image "occlude" (Fig. 4). Because of their experience, most observers would probably interpret the combinations of surfaces contained in the leftmost object to actually be two objects as shown on the right in Fig. 16. This interpretation combines surfaces 1 and 3 from the leftmost object into one surface occluded by surface 2. Although the interpretation shown on the right may be the most probable, we realize that the description of the leftmost object in fact has many possible interpretations. Without taking into account surface adjacencies and other factors, there are in fact 16 possible interpretations, one for each combination of the 4 surfaces involved. Some of these interpretations will of course not make any sense in light of other information.

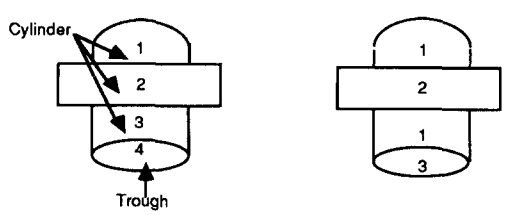

Fig. 16. Occluded objects: surfaces 1 and 3 in the left figure have been interpreted as being the same surface in the right figure.

One possible method for determining which are the meaningful interpretations, would merely involve evaluating each possibility separately, examining all of the adjacencies and situations where two surfaces should be interpreted as one. A different approach was utilized in this study. Simply stated, this method involved the recursive procedure of extracting the occluding surfaces, defining the resulting possible interpretations, and placing these new objects back on the data queues, themselves to be matched and evaluated for possible multiple interpretations due to occlusion. Extracting surfaces in this manner involved only the high-level data representations, and did not involve modifying image data in any way.

No matter which method is used to resolve occlusion, it will most likely depend on having surface adjacency information. The method applied to determine surface adjacencies was fairly straightforward. Briefly, this approach involved traversing the object from the center of one surface to the center of another and counting the number of edges between them. If there was only one edge, it was concluded that the two surfaces were adjacent. It is obvious that this approach depends on the fact that surface boundaries are not concave. In other words, they were not allowed to fold back on themselves. None of the objects used by this study had difficulties with this restriction.

Figure 17 shows the results of evaluating image "occlude", (Fig. 4), giving as the goal any object composed of two adjacent surfaces, one cylindrical, and the other a trough. The printout gives a description of all possible interpretations of the occluded objects. Objects 0–15 correspond to (binary) interpretations 0–15. Interpretations 12 and 13 were correctly identified as the only matching interpretations, while interpretations 0, 9, 10, and 11 were identified as being impossible. The printout lists object 13 as having only 2 visible surfaces. This is due to the fact that original surfaces 1 and 3 were successfully combined into a single cylindrical surface.

4. MULTIPROCESSING RESULTS

In order to properly evaluate the success of the multiprocessing aspects of this study, several basic measurements were used. They are total speedup, plots of speed vs the number of processors, processor utilization under varying circumstances and the progression of processing in time for each object to be recognized. These results are discussed in this section. For the sake of establishing the accuracy of timing measurements, duplicate measurements were occasionally performed which indicated that timing measurements may vary by approximately 50–100 ms from one measurement to the next, all other factors

```
MATCH:   Obj 13 matches goal; at 1392
MATCH:   Obj 12 matches goal; at 1409


object 0, has 0 visible surfaces:
object 1, has 1 visible surfaces:
      surf 1;   3      79
object 2, has 1 visible surfaces:
      surf 1;   3      66
object 3, has 2 visible surfaces:
      surf 1;   3      66
      surf 2;   3      79
object 4, has 1 visible surfaces:
      surf 1;   3      79
object 5, has 1 visible surfaces:
      surf 1;   3      79
object 6, has 2 visible surfaces:
      surf 1;   3      79
      surf 2;   3      66
object 7, has 3 visible surfaces:
      surf 1;   3      79
      surf 2;   3      66
      surf 3;   3      79
object 8, has 1 visible surfaces:
      surf 1;   4      158
object 9, has 0 visible surfaces:
object 10, has 0 visible surfaces:
object 11, has 0 visible surfaces:
object 12, has 2 visible surfaces:
      surf 1;   3      79
      surf 2;   4      158
object 13, has 2 visible surfaces:
      surf 1;   3      79
      surf 2;   4      158
object 14, has 3 visible surfaces:
      surf 1;   3      66
      surf 2;   3      79
      surf 3;   4      158
object 15, has 4 visible surfaces:
      surf 1;   3      79
      surf 2;   3      66
      surf 3;   3      79
      surf 4;   4      158
```

Fig. 17. Printed results after resolving interpretations of occluded objects in Fig. 16.

remaining constant. The initiation of timing should also be explained. Because the Butterfly does not have a file system, all code and image data was first down loaded to the Butterfly. Timing began once this had been accomplished and all processors had been started and initialized.

4.1. *Total speedup*

This is the simplest measurement of performance and is displayed in Table 2 for images R7, R5, R3, and R1. This table compares the time taken by a single Butterfly processor to process the applicable image, against the best or fastest processing time when using multiple processors.

4.2. *Speedup vs the number of processors*

This is one of the most common measures of performance of algorithms implemented on a multiprocessor. As shown in Figs 18 and 19, the results are displayed by plotting the reciprocal of the normalized processing time (speedup), against the number of processors used. Processing times were normalized by the processing time for a single processor. The ideal is to achieve a "linear speedup", where the plot is not only linear, but the slope of the plot is 1. To better understand this goal, it may help to realize that unless synergistic relationships are possible in multiprocessing, the best one could hope to achieve is that the processing time would be reduced by two when the number of processors have been doubled. This defines a function of the form, $f(x) = 1/x$, where $f(x)$ is time and x is the number of processors. As has been mentioned, however, the inverse function is normally plotted, $f(x) = x$, where $f(x)$ is 1/time and x is the number of processors. This of course defines a linear function of slope 1, and becomes the upper limit or goal. The overhead involved in multiprocessor communication, shared and remote data access, and work distribution usually make this a very difficult goal to achieve.

Table 2. Total speedup

Image	Single processing time	Best time
R7	21.07 s	2.64 s
R5	18.10 s	2.20 s
R3	8.09 s	1.48 s
R1	3.89 s	1.41 s

Figure 18 shows the speedup plot obtained when processing image R7 (Fig. 3), containing 7 objects. The plot is indeed linear with a slope of about 0.7 through 10 processors, at which time the plot becomes non-linear showing essentially no speedup beyond 12 processors. This is to be expected due to the fact that most of the parallelism is achieved by processing different objects simultaneously. If this were the only form of parallelism employed, however, the plot would in fact only be linear through 7 processors, corresponding to the 7 objects in the image. The fact that the linearity extends well beyond 7 indicates that efforts to utilize parallelism within objects were useful.

The plot for image R3 with three objects, (Fig. 19), also demonstrates that while multiprocessing benefits do extend beyond the one to one ratio of processor to objects in the image, the number of useful processors is still linked to the number of objects in the image. For three objects, linearity with a slope of about 0.6 is maintained through 6 processors.

It is of interest to see what effect multiprocessing had on speedup when processing was halted on

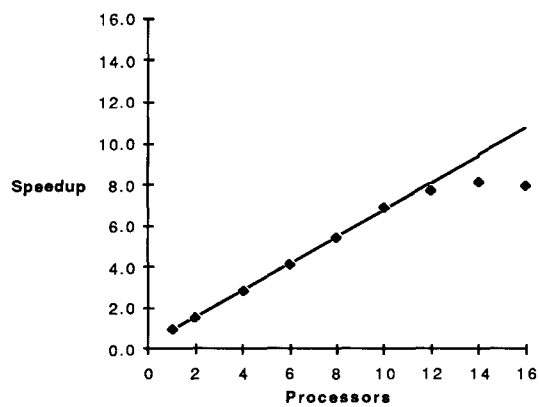

Fig. 18. Speedup plot for R7.

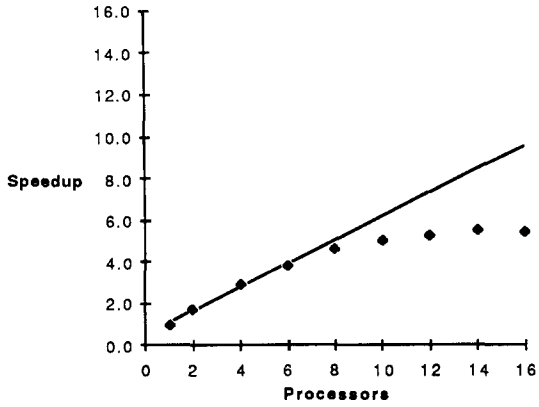

Fig. 19. Speedup plot for R3.

detection of the first matching object. Figures 20 and 21 show speedup plots on R7 when the goal object was the closed can (object 1) and the large sphere (object 4) respectively. Because the can was the first object found and was the goal object, it was processed immediately with no processing time diverted to the other objects. One would expect that adding additional processors would do little to improve speed in this scenario. In fact, the slope of the line in Fig. 20 is about 0.04. Changing the search goal to an object located in the middle of the image, (Fig. 21), once again demonstrates the benefits of multiple processors. The approximate slope of this line is 0.4

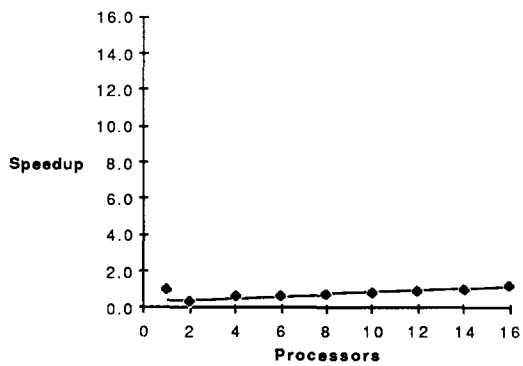

Fig. 20. Speedup plot for R7 goal: closed can (object 1).

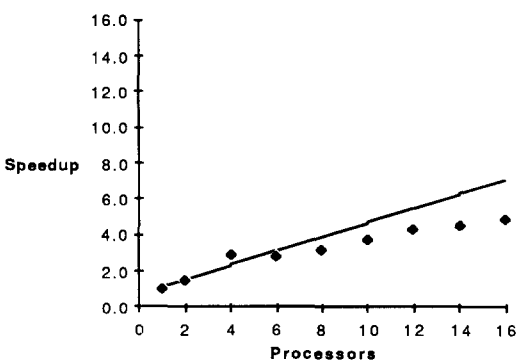

Fig. 21. Speedup plot for R7 goal: large sphere (object 4).

compared to 0.7 when processing all objects. Similar results were observed when performing goal-directed searches on images R5 and R3.

Table 3 shows a different viewpoint of essentially the same data. It compares times obtained when processing all objects to those obtained when halting processing on the identification of the first match. The number in the goal object column refers to the order in the image in which the goal object was initially found. That is, a goal object of 3 means that the goal object specified is the third object to be located by the find objects routine (see Fig. 5). Results, of course, depend greatly on the order in which the objects are initially located. These results show that performing a goal-directed search did in fact yield additional speedups from 12 to 33% beyond those times recorded when processing all objects.

The utility of combining multiprocessing with a goal-directed search has been and is still a debatable issue. It should be noted that at least for the complexity of the images and processing required in this research, it was found that although performing a goal-directed search did in fact diminish the usefulness of multiprocessing, overall speedups (with 16 processors) from 6.5 to 10.2 times were achieved using multiprocessing in combination with a goal-directed search for the images R3, R5 and R7. This was due primarily to the fact that most of the processing time was spent performing low-level image processing, which had to be accomplished before sufficient knowledge was available to prioritize the work. Images of higher complexity may very well require greater portions of processing to be accomplished at higher levels, enabling the goal-directed approach to have an even greater effect.

For the control strategy used in this system, the advantages to be expected from multiprocessing and heuristic search actually seem to be somewhat independent, and can be determined from the following two observations. (1) The more compute intensive the processing to be performed, the greater the benefit of multiprocessing. (2) The sooner object characteristics can be determined, the greater the advantage of heuristic search. In other words, if there remains a considerable portion of computer intensive work after work prioritization is possible, the greater is the benefit of performing a goal-directed search.

4.3. *Processor utilization*

Processor utilization in this study is defined as a processor's processing time divided by the total elapsed time. Processing time is the time spent by a processor performing real work. In other words, it does not include idle time checking data and job queues. Total elapsed time is the interval between the start and the time at which all the objects have been identified. Total processing time is the sum of the individual processing times (i.e. the total amount of work required to process the entire image). Average

Table 3. Speedup resulting from goal-directed search. Times are given in ms

Goal object	Image	Total time	First match time	% Speedup
1	R7	2601	2285	12%
4	R7	2601	2223	15%
6	R7	2601	2285	12%
2	R5	2270	1788	21%
4	R5	2270	1992	12%
5	R5	2270	2007	12%
1	R3	1479	984	33%
2	R3	1479	1245	16%
3	R3	1479	1144	23%

processing time is the total processing time divided by the number of processors. Finally, the average processor utilization is the average processing time divided by the total elapsed time.

Processor utilization was computed for 16 scenarios; one for each of four different processor configurations on each of the 4 images R1–R7. Figure 22 shows four plots portraying the utilization measurements made for each of the four configurations, (i.e. 16, 12, 8, and 4 processors), relative to processing done in image R7. Figure 23 shows the utilization for 16 processors on image R1. The other cases were not included because of their similarity to the results observed in these two figures. Table 4 displays the total elapsed times, total processing times, and the average processor utilization for each of the 16 scenarios measured.

The most important observation to be made from examining Figs 22 and 23, is that they show a fairly even distribution of work among the processors as long as there is sufficient work to be performed in the image. Beginning with image R3 it was observed that several processors were doing an uneven portion of the work. This became even more pronounced on image R1 (Fig. 23). This, of course, is to be expected. In fact it is somewhat surprising that all of the 16 processors were able to participate in the processing of even the single object in image R1.

Table 4. Processor utilization. Times are given in ms

Image	Processors	Elapsed time	Total process time	Average utilization
R7	16	2,528	25,083	62.0%
R7	12	2,951	24,939	70.5%
R7	8	3,844	24,241	78.8%
R7	4	7,327	25,183	85.9%
R5	16	2,162	20,177	58.3%
R5	12	2,634	20,514	64.9%
R5	8	3,492	20,165	72.2%
R5	4	5,639	19,968	88.5%
R3	16	1,558	9,555	38.3%
R3	12	1,502	9,410	52.2%
R3	8	1,740	9,332	67.0%
R3	4	3,087	9,289	75.2%
R1	16	1,484	5,738	24.2%
R1	12	1,237	5,294	35.7%
R1	8	1,382	5,392	48.8%
R1	4	2,765	6,728	60.8%

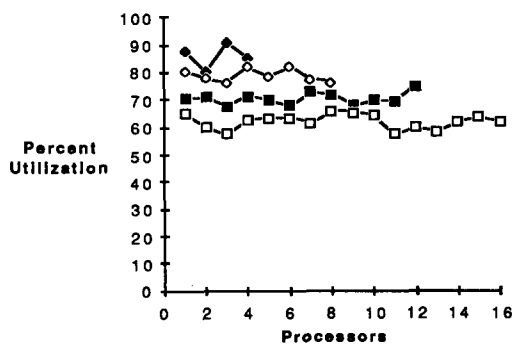

Fig. 22. Processor utilization on R7. Top to bottom, utilization is shown for 4, 8, 12, and 16 processors respectively.

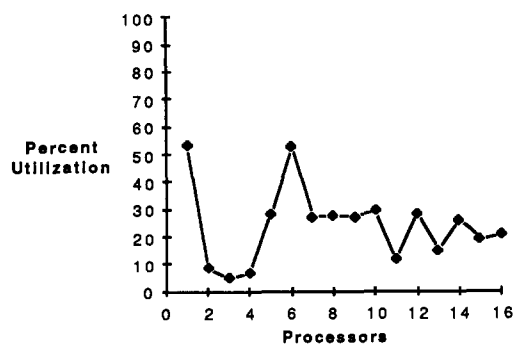

Fig. 23. Processor utilization on R1 with 16 processors.

A number of observations can be made by examining the data shown in both Table 4 and Fig. 24. The primary conclusion is that in general, as image complexity increases, processor utilization also increases. Second, as one would expect, the total processing time on each image was essentially independent of the number of processors being used. This adds credibility to the average processor utilization values which were based on this total processing time. Finally, as was discovered when examining speedup, both images R1 and R3 show an increase in total elapsed time when increasing the number of processors from 12 to 16. This confirms the earlier conclusion that there is probably an overhead and memory contention penalty to be paid when increasing the number of processors beyond that warranted by the amount of work to be accomplished in the image.

In general, Fig. 24 confirms the conclusion that the fewer the processors and/or the higher the image complexity, the better the processor utilization. The natural expectation is that increasingly complex images could be effectively and efficiently processed by increasing numbers of processors. Contrary to this expectation, however, all of these plots indicate that there is a limit to the processor utilization. In particular, the plot for the four processor configuration shows a slight decline in the processor utilization when increasing image complexity from 5 to 7 objects. This seems to suggest that if more complex images were available, the utilization of processors may not increase significantly. This apparent limitation is most likely the result of "hot spots" or memory contention between processors attempting to access the same global data structures at the same time. It is probable that there are refinements, possibly at a low level utilization of the Butterfly memory and processors, which would improve these results somewhat. In order to make more definite conclusions such refinements should be studied, and tests with more complex images should be made.

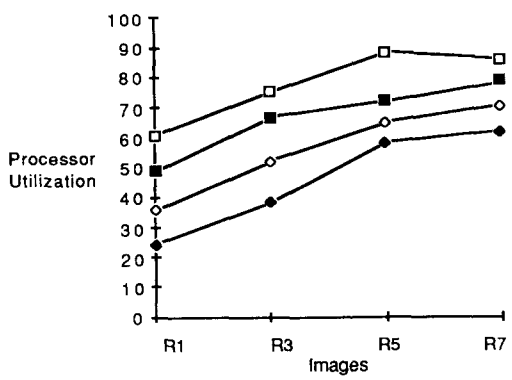

Fig. 24. Utilization summary. The plots from top to bottom show processor utilization for 4, 8, 12, and 16 processors respectively, as image complexity increases from one to seven objects in images R1–R7.

PR 22:1-E

5. CONCLUSIONS

A new method of surface characterization using curvatures is presented. It uniquely classifies each surface type according to a plot of its principal curvatures in conjunction with a histogram analysis. Goal-directed search is used for the recognition of objects. Range information is used to resolve ambiguities associated with occlusion. Combining rule-based control with distributed processing leads to several interesting control issues, which are discussed in detail. The performance of the system is evaluated with respect to the number and types of objects, size of the images, location of objects, occluded/non-occluded objects, speed vs number of processors and processor utilization. By applying the use of multiprocessing, not only to process image objects simultaneously, but also to accelerate processing within each object, near linear speedups were obtained. Results from this study seem to indicate that multiprocessing is warranted any time there is a great deal of computationally intensive processing, independent of whether a rule-guided approach is used. The requirements of this system were such that most of the processing time was spent performing low-level tasks, which had to be accomplished before sufficient knowledge was available to prioritize the work. Consequently, greater advantage was achieved through the use of multiprocessing than was realized using the rule-guided search.

By equally equipping all processors with the ability to locate and process work, this system was able to achieve a balanced work load between as many as 16 processors when processing images with as few as 5 objects. As expected, however, the average processor utilization depended directly upon the amount of work (i.e. the number of objects), in the image. It appeared that there may be a utilization limit for this system of about 88%, due most likely to the overhead involved with the control algorithms, and delays due to memory contention. It would be valuable to explore in more depth the exact nature and extent of this apparent limitation.

SUMMARY

The recent advent of Multiple Instruction Multiple Data (MIMD) architectures together with the potentially attractive application of range images for object recognition, motivated the development of a successful goal-directed 3-D object recognition system on a 18 node Butterfly multiprocessor.

A new method of surface characterization using curvatures is presented. It uniquely classifies each surface type according to a plot of its principal curvatures in conjunction with a histogram analysis. Goal-directed search is used for the recognition of objects. Range information is used to resolve ambiguities associated with occlusion. Combining rule-based control with distributed processing leads to several interesting control issues, which are discussed

in detail. The performance of the system is evaluated with respect to the number and types of objects, size of the images, location of objects, occluded/non-occluded objects, speed vs number of processors and processor utilization. By applying the use of multiprocessing, not only to process image objects simultaneously, but also to accelerate processing within each object, near linear speedups were obtained. Results from this study seem to indicate that multiprocessing is warranted any time there is a great deal of computationally intensive processing, independent of whether a rule-guided approach is used. The requirements of this system were such that most of the processing time was spent performing low-level tasks, which had to be accomplished before sufficient knowledge was available to prioritize the work. Consequently, greater advantage was achieved through the use of multiprocessing than was realized using the rule-guided search.

The system was able to achieve a balanced work load between as many as 16 processors when processing images with as few as 5 objects. As expected, however, the average processor utilization depended directly upon the amount of work (i.e. the number of objects), in the image. It appeared that there may be a utilization limit for this system of about 88%, due most likely to the overhead involved with the control algorithms, and delays due to memory contention.

REFERENCES

1. B. G. Baumgart, Geometric modeling for computer vision, Technical Report AIM-249, STAN-CS-74-463, Department of Computer Science, Stanford University (1974).
2. P. Besl and R. Jain, Invariant surface characteristics for 3D object recognition in range images, *Comput. Vision, Graphics Image Processing* **33**, 33–80 (1986).
3. T. J. Fan, G. Medioni and R. Nevatia, Description of surfaces from range data, *Proc. DARPA Image Understanding Workshop*, pp. 232–244 (1986).
4. C. Hansen and T. Henderson, Utah range data base, Technical Report UUCS-86-113, Department of Computer Science, University of Utah (1986).
5. D. M. McKeown, W. A. Harvey and J. McDermott, Rule-based interpretation of aerial imagery, *IEEE Trans. Pattern Analysis Mach. Intell.* **7**, 570–585 (1985).
6. A. M. Nazif and M. D. Levine, Low level image segmentation: an expert system, *IEEE Trans. Pattern Analysis Mach. Intell.* **6**, 555–577 (1984).
7. B. Bhanu, Representation and shape matching of 3-D objects, *IEEE Trans. Pattern Analysis Mach. Intell.* **6**, 340–351 (1984).
8. D. J. Ittner and A. K. Jain, 3-D surface discrimination from local curvature measures, *Proc. IEEE Conf. on Comput. Vision Pattern Recognition* (1985).

About the Author—BIR BHANU received the S.M. and E.E. degrees in electrical engineering and computer science from the Massachusetts Institute of Technology, the Ph.D. degree in electrical engineering from the University of Southern California and the M.B.A. degree from the University of California, Irvine.

Dr Bhanu is a Staff Scientist at Honeywell Systems and Research Center, where he serves as Principal Investigator of the Strategic Computing Computer Vision Program from DARPA, Obstacle Detection Program from NASA, and Machine Learning Program from a government agency. Additionally, he is conducting IR&D research efforts in contextual analysis, robotic combat vehicle navigation, multisensor integration, parallel algorithms, photointerpretation and surveillance. He has also worked with IBM on image processing, INRIA-France on 3-D object recognition, and Ford Aerospace and Communications Corporation on Automatic Target Recognition. While on the faculty of the University of Utah he was the Principal Investigator on several NSF and industry-funded research projects in machine intelligence. Presently he is also an Adjunct Associate Professor of Computer Science at the University of Utah.

His current interests include computer vision, robotics, target modeling, distributed sensing and control, parallel computer architectures, pattern recognition and artificial intelligence. He has more than 100 publications in these areas and is a reviewer for over a dozen technical publications and government agencies. He has given national short courses on intelligent automatic target recognition. He is the guest editor of a special issue of IEEE Computer on "CAD-Based Robot Vision". He is listed in the American Men and Women of Science, Who's Who in the West and Personalities of Americas.

He is a member of ACM, AAAI, Sigma Xi, Pattern Recognition Society, SPIE, IEEE (Senior member) and IEEE Computer Society.

About the Author—LARRY A. NUTTALL received a B.A. degree in Physics and a B.S. degree in Computer Science from Brigham Young University in 1974 and 1981 respectively. Mr Nuttall received the M.S. degree in Computer Science from the University of Utah in 1987.

Mr Nuttall was previously employed as a Staff Engineer with the Hercules Advanced Methods group, Hercules Aerospace Division, Magna, Utah. He is currently employed as a Senior Engineer with the Computer Vision and Imaging Technology Group of Perceptics Corporation, Knoxville TN. In both positions he has worked on various image processing design and implementation problems, including the detection and tracking of moving objects in digital images, and the design and development of NDE image inspection systems.