ANALYSIS OF SCENES CONTAINING SEVERAL OCCLUDING CURVILINEAR OBJECTS[1]

K. R. Yam[2],
W. N. Martin[2], and
J. K. Aggarwal[2,3]

TR-135 February, 1980

[1]This research was supported in part by the Air Force Office of Scientific Research under grant AFOSR 77-3190.

[2]Computer Sciences Department
The University of Texas at Austin
Austin, Texas 78712

[3]Electrical Engineering Department
The University of Texas at Austin
Austin, Texas 78712

Abstract

This report describes the techniques used and the results obtained by a system developed to analyze edge pictures derived from scenes containing several occluding curvilinear objects. A library of allowable object types is maintained to aid the recognition processes of the system. The images of the scenes are analyzed to yield edge pictures from which surface boundaries are extracted. These boundaries are then transformed into a description which effectively and efficiently represents the shape of the boundaries and facilitates the matching of those shapes to the stored library models.

ANALYSIS OF SCENES CONTAINING SEVERAL OCCLUDING CURVILINEAR OBJECTS

Section I

This report describes the techniques used and the results obtained by a system developed to analyze edge pictures derived from scenes containing several occluding curvilinear objects, Yam [1]. This broad problem is made manageable by a few simplifying assumptions and restrictions. It is assumed that the objects in any one scene are all instances of a single object type. For a given scene the object type is not known, however, it is restricted to be one of a set of object types defined by the system. Each object type is represented by the simple closed boundary derived from the edge picture which is the library view of the standard object for that type.

In each scene it is assumed that at least one of the object instances is completely visible and that all of the instances are at similar depths from the camera. In this way the scenes are much like overhead views of parts bins. Each bin might contain one type of part with the parts stacked in a pile. This pile might be fairly randomly arranged except that there would be at least one topmost part that is not occluded.

In this system the scene is viewed by a TV camera yielding a 128×128 pixel image with eight bits per pixel representing the intensity information. This image is transformed into an edge picture by the algorithm described in McKee and Aggarwal [2]. Each pixel of an edge picture is marked by a 1, i.e., as an edge point, if the corresponding area in the image contains a significant discontinuity in intensity value; otherwise the pixel value is set to 0. The edge finding algorithm also thins the connected components of 1's in the edge picture so that the intensity discontinuity boundaries are only one pixel thick. That is, every edge point has at most two 8-connected neighbors which are also edge points, except if the original edge point occurs at the intersection of several boundaries. These intersection points are called vertices, and the connected set of edge points joining two vertices is an edge. Figure 1 shows an edge picture with annotations indicating the vertices, eg. V1, and edges, eg. E6, derived from an example scene containing two objects.

One can note that these sorts of vertices and edges partition the edge picture into a set of areas, which we call surfaces. Every edge separates exactly two surfaces and each surface is bounded by a simple closed curve comprising one or more edges connected at their common vertices. For the background surface, "bounded" is not quite the appropriate word, however, there is a set of vertices and edges which separates the background from the surfaces of the objects in any connected "pile". This concept of surface is fundamental to the approach of this system. As stated earlier, each allowable object type is defined in terms of the boundary of the surface derived from a library view of a standard object. This boundary is represented by a spatial coordinate list of its edge points. The order of the list is such that a sequential scan through the list corresponds to a clockwise traversal of the boundary with consecutive list elements being neighbors in the image. By "clockwise" we mean that for each traversal from the pixel of a list element to the pixel of the immediately succeeding list element the surface of interest is to the right of the direction defined by that

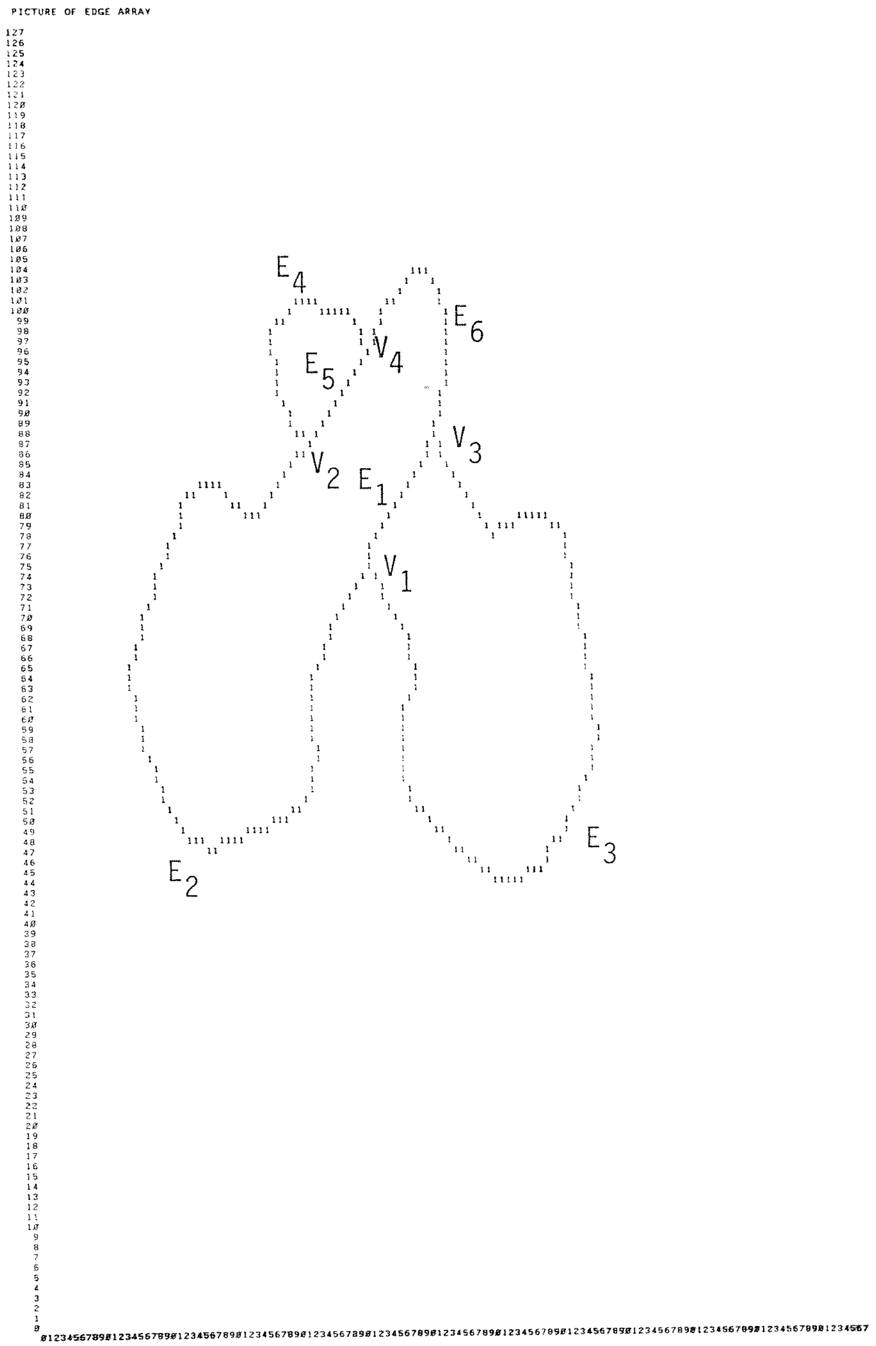

Figure 1. An edge picture containing two objects, four vertices, six edges, and four surface boundaries.

traversal.

This spatial coordinate list is then transformed into a representation (to be discussed later) which allows the system to match boundary shapes regardless of their orientation and scale. The shapes matched by the system are the various object models against the surfaces in the given image. These two types of shapes are conformable because both the object models and the image surfaces are simple closed boundaries. In fact, the surface corresponding to the unoccluded object on the "pile" should be a perturbed instance of one of the object models. In this manner the main task of the system can be stated in the following steps: form the edge picture; extract the edge points of the surface boundaries; form the shape representation of each boundary; and determine which of the boundaries match one of the object models. This latter step will, at the same time, determine which surfaces are unoccluded and "recognize" which type of objects the scene comprises. In addition, the shape matching process will calculate the scale of the scene objects relative to the model object. This scale factor along with the knowledge of the type of scene object will allow the system to analyze the remaining, occluded portions of the image.

The extraction of the surface boundaries is discussed in Section II. The shape representation and matching is described in Section III. Section IV presents the "recognition" process. While an example is discussed and some concluding remarks are made in Section V.

Section II

This section will present the algorithm used by the system to extract the surface boundaries. Now to be conformable to the object models the system needs to form the surface boundaries so that they are closed curves with the coordinate points ordered in a clockwise traversal. This could be done by sequentially scanning the edge picture until an edge point was found. This scan would necessarily start in the background surface and the edge point found by the scan would be on the boundary of that surface. One could then determine the clockwise edge point neighbor of the given edge point, store the neighbor's coordinate values, and continue tracing the boundary in a clockwise direction until the original point was encountered again. Since every non-vertex edge point bounds two surfaces this boundary tracing algorithm could then be applied to the once traversed edge points by determining the clockwise neighbor relative to the remaining surface. Continuing this process until every non-vertex edge point has been traversed exactly twice, the system could form the coordinate list of the boundary of every surface in the edge picture.

In the system discussed here, however, the edge picture is first preprocessed to yield a table of edges. For every edge, this table contains an edge name, the two associated vertices, and a pointer to the coordinate list for the non-vertex edge points of the edge. From this table another table, called the Edge Traversal Order table, is formed with each element containing an edge name, a "rear" vertex, a "front" vertex, and a boundary name. Every element of the edge table generates two ETO's, elements of the Edge Traversal Order table. Each of the two ETO's are given the edge table element's edge name, then one of the ETO's has the first vertex associated with the edge table element as its "rear" vertex and the other associated vertex as its "front" vertex, while the other ETO has the associated vertices in the opposite entries. These two ETO's correspond to the two traversals of the edge in opposite directions, where the clockwise traversal is from the rear vertex to the front vertex. The boundary name entry is initialized to a null value and will be filled in by the algorithm given in Figure 2.

The process begins with this initialized ETO table and terminates with a set of lists representing the set of surface boundaries. Each list corresponds to one boundary and is a circularly connected list of ETO's ordered to yield a clockwise traversal of the boundary when the edges of successive list elements are connected at their common vertices, i.e., front to rear or rear to front. Figure 3 shows the edge table, ETO table, and resulting boundary lists derived from the edge picture of Figure 1. These boundary descriptions are now transformed into a shape representation as discussed in the next section.

Algorithm:

1. Initialize the ETO table from the edge table of the input scene.
2. Select an unmarked ETO from the table and call it CURRENT; call the front vertex of CURRENT V1; call the edge of CURRENT E1; and call the rear vertex of CURRENT VS.
3. Initialize the boundary list with CURRENT as the first and only element.
4. Find the set of ETO's with V1 as a rear vertex and call the set S.
5. From S determine the ETO for which the edge is rightmost with reference to E1 at V1, and then call it CURRENT.
6. Add CURRENT to the boundary list.
7. Mark CURRENT as visited in the ETO table.
8. Set E1 to be the edge of CURRENT.
9. Set V1 to be the front vertex of CURRENT.
10. If V1 is not the same as VS then return to step 4, otherwise continue.
11. Store the completed boundary list.
12. If any ETO's remain unmarked, i.e., not visited yet, then return to step 2, otherwise terminate: all surface boundaries have been found.

Figure 2. Surface boundary extraction algorithm.

Edge Table

edge name	two associated vertices	
E1	1	3
E2	1	2
E3	3	1
E4	2	4
E5	4	2
E6	4	3

Edge Traversal Order Table

edge name	rear vertex	front vertex	boundary name
E1	1	3	B1
E2	1	2	B3
E3	3	1	B3
E4	2	4	B3
E5	4	2	B1
E6	4	3	B3
E1	3	1	B2
E2	2	1	B1
E3	1	3	B2
E4	4	2	B4
E5	2	4	B4
E6	3	4	B1

Boundary Lists

boundary name	circular list of ETO's yielding a clockwise traversal (name, rear, front)
B1	((E1,1,3),(E6,3,4),(E5,4,2),(E2,2,1))
B2	((E1,3,1),(E3,1,3))
B3	((E2,1,2),(E4,2,4),(E6,4,3),(E3,3,1))
B4	((E4,4,2),(E5,2,4))

Figure 3. Results of applying the surface boundary extraction algorithm to the input scene of Figure 1.

Section III

Shape representation of digital curves by chain codes was introduced by Freeman [3]. McKee and Aggarwal [4] developed a convenient and effective algorithm for representing the shape of a curve and determining the similarity in shape of two curves based on a modification of the chain code. The modifications to the simple chain code include the following: making the code "continuous", that is, adding a multiple of 8 to code values when necessary to make consecutive values differ by no more than 4. Compensating for the difference in horizontal and diagonal distances on a rectangular grid by repeating the code values for diagonal neighbors three times and repeating all other code values twice. Smoothing to limit the effects of noise by replacing each code value with the average of that value, the four code values before it, and the four code values after it.

These modified chain code values can now be thought of as a function of the arc length along the curve measured from the starting edge point of the chain code (see Martin and Aggarwal [5] for a different derivation of an equivalent representation). Now to make the representation manipulatable the pictorial graph of the function is considered and a piecewise straight line approximation of the graph is formed. In this way the graph can be retained through a small ordered list of lines with each line represented by its slope, length, and starting code value.

Two curves, represented in this way, can be compared for similarity in shape by calculating the area between their graphs. Of course, any given graph is sensitive to the orientation and scale of the generating curve. However, the straight line approximation of a graph can be manipulated to make comparisons which are not sensitive to these variations.

Scale changes to a curve are directly transformable to scale changes in the associated graph, while a shifting and normalizing procedure can be applied to the graph in order to represent the curve in a full range of orientations. For details of these manipulations see [1] and [4]. In this report we have included Figure 4, the library view for an object type, and Figure 5, the code value versus arc length graph with the straight line approximation superimposed, as an example of how the shape of a curve is represented.

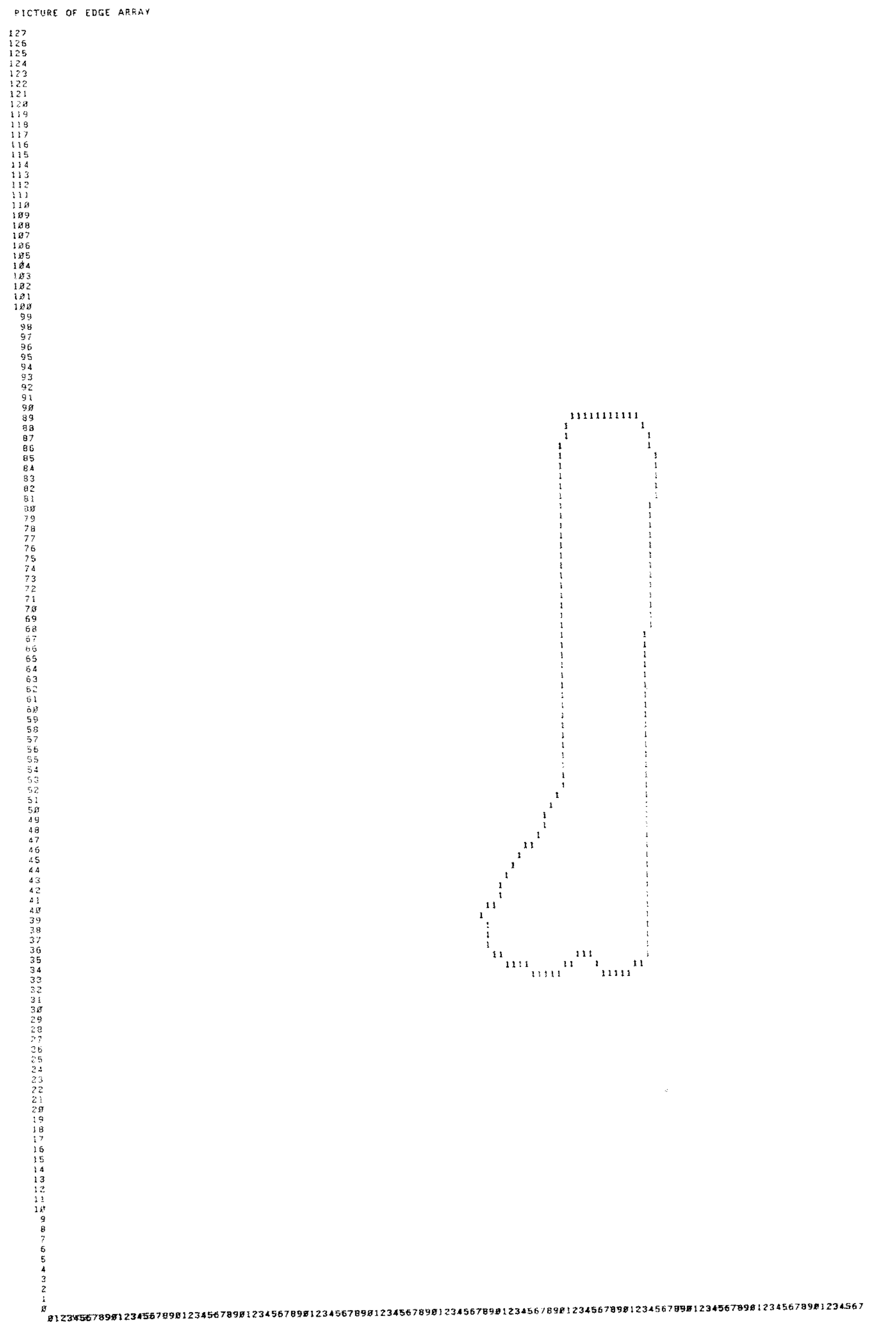

Figure 4. Library view of object type, OBJTYP1.

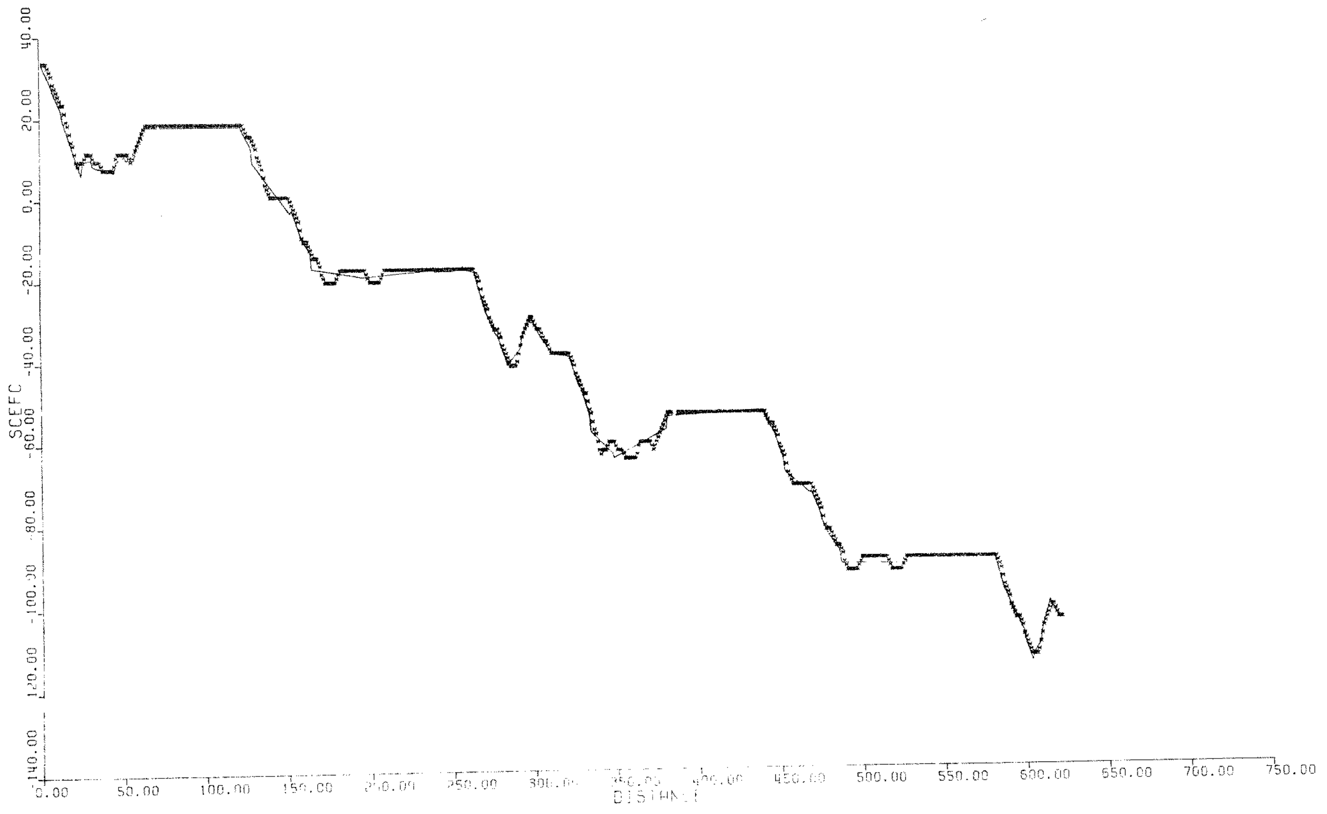

Figure 5. Modified chain code graph of object type shown in Figure 4.

Section IV

As stated in earlier sections this system is designed to analyze the edge pixtures of scenes containing several occluding curvilinear objects with the aid of a library of known object types. The shape information stored in the library is matched against the shape information derived from the surface boundaries of the input scene in order to both locate the unoccluded, i.e., foremost, object and recognize which type of object is in the scene. This function is performed by taking each surface boundary in turn, comparing its shape representation to every library model, and selecting the boundary-model pair which exhibits the most similarity in shape.

A given boundary is compared to a chosen model by first scaling the length of the code graph of the boundary to be the same as the length of the model code graph, and then repeatedly applying the shift and normalize procedure to produce a full range of orientations for the boundary code graph. For each orientation the scaled graphs are compared, i.e., the area between them is calculated, then the orientation resulting in the closest match is designated as indicating the actual similarity between that boundary-model pair. The similarity of all boundary-model pairs are then compared to find the best match. The selected boundary-model pair simultaneously determines which object type is in the scene and which of the surfaces corresponds to an unoccluded object instance.

Knowing the object type present in the scene and the scale factor for normalizing the surface boundaries to the associated library model allows the system to process the remaining boundaries which, in part, correspond to occluded object instances. This processing can often determine which object parts are visible through these occluded views, but the success of this process depends heavily upon the severity of the occlusion.

Section V

The system described in the preceding sections was run on several example scenes with a library of six object types, one of which is shown in Figure 4. Figure 6 displays a scene composed of three instances of the object type of Figure 4. The extracted surface boundaries are shown in Figure 7. The boundary of Figure 7a is the background boundary and need not be considered. The results of comparing the other surface boundaries to the library object types are tabulated in Figure 8, with OBJTYP1 designating the object type of Figure 4, and BOUNDB, BOUNDC, and BOUNDD corresponding to the boundaries in Figures 7b, c, and d, respectively. Notice that BOUNDB and BOUNDD are most similar, i.e., have the least measured area, to OBJTYP5, while BOUNDC is most similar to OBJTYP1. In particular note that the BOUNDC-OBJTYP1 measure is the best of all the boundary-model pairs. Thus for the example scene of Figure 6, the surface boundary of Figure 7c is correctly identified as the unoccluded object instance and the object type of the scene is correctly recognized as being that of Figure 4. For further details about this example see [1].

In this report we have described an automated system for the analysis and recognition of scenes containing multiple occluding curvilinear objects. A few simplifying assumptions, such as the occurrence of at least one unoccluded object instance, and a library of object types are used in implementing a recognition scheme based on a shape similarity measure. For this sytem the shapes of interest are the surface boundaries extracted from an edge picture of the scene and are represented by a modified chain code. The success obtained in this constrained domain is encouraging and provides an impetus for further research into methods which can relax the major restrictions needed in the present system.

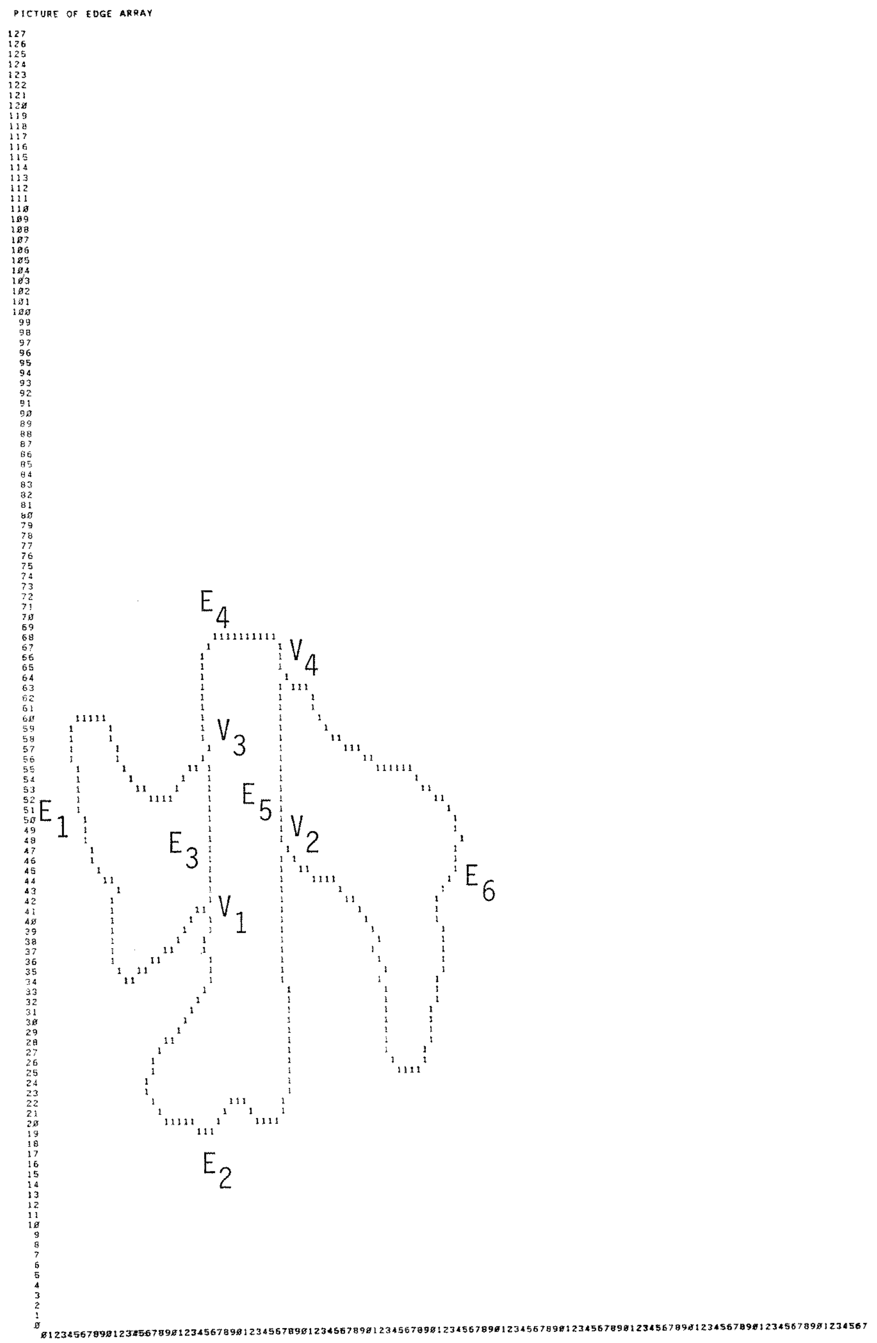

Figure 6. An example image of three instances of the object type shown in Figure 4.

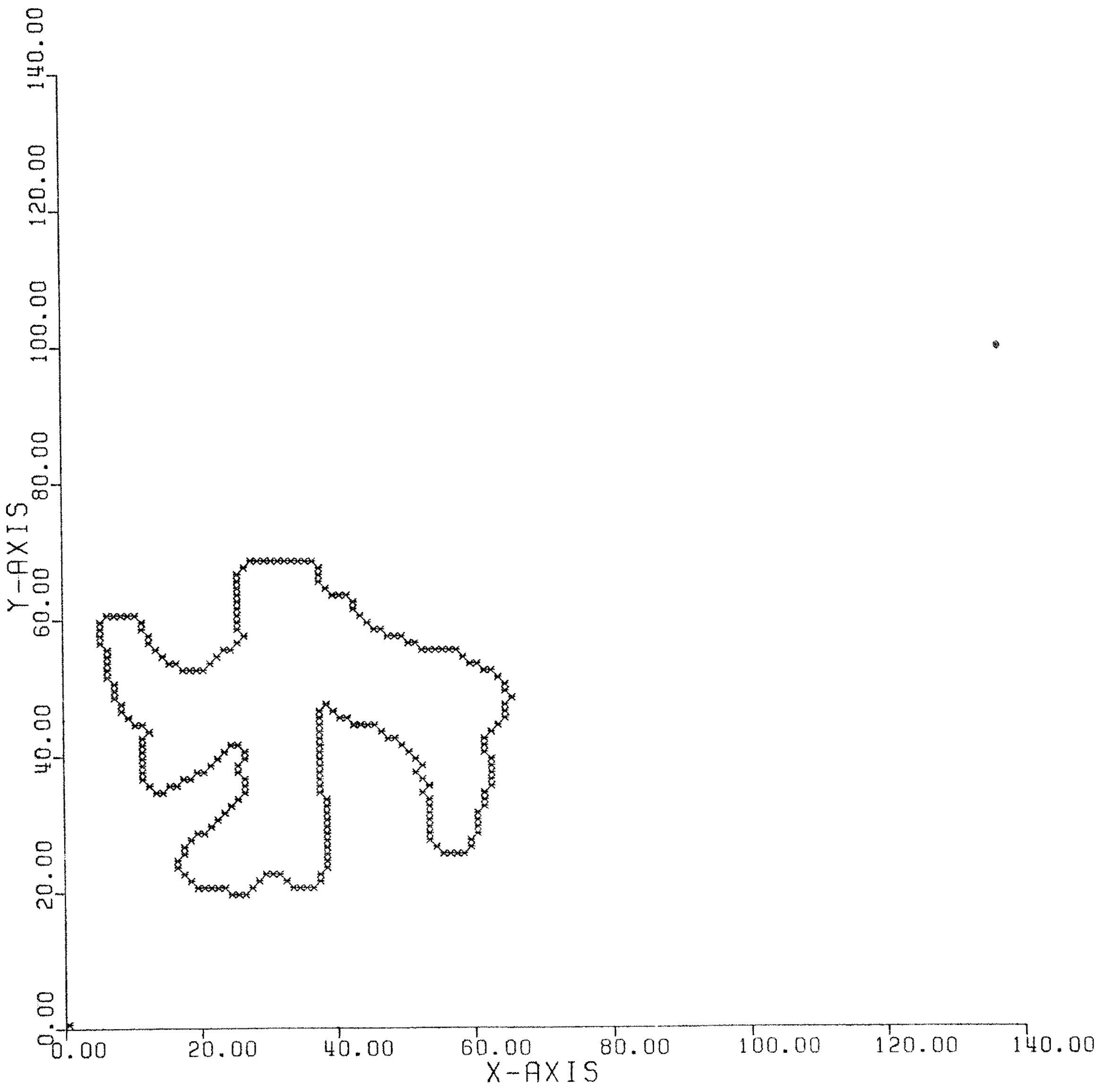

Figure 7a. Background surface boundary extracted from the input image shown in Figure 6.

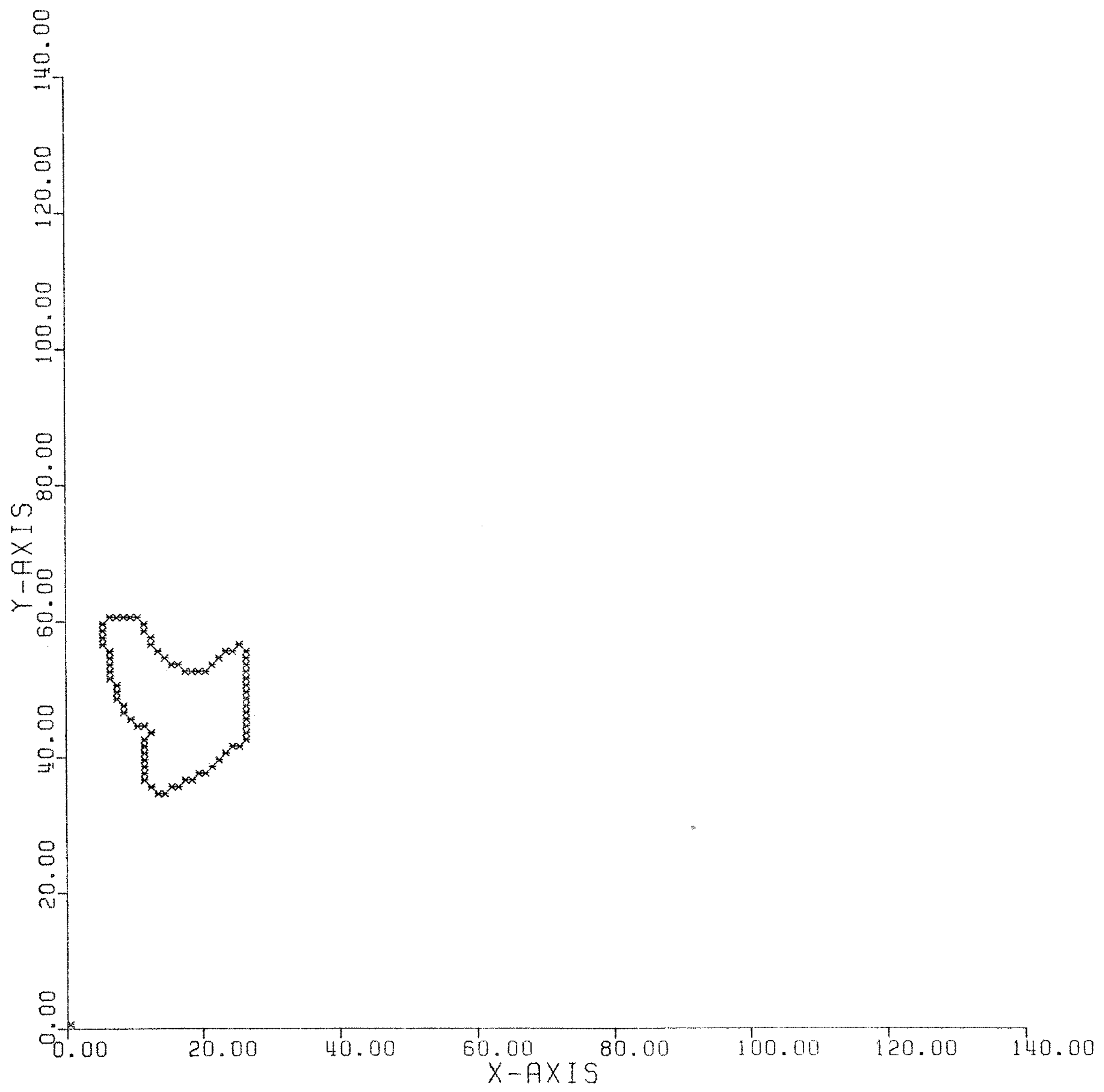

Figure 7b. BOUNDB, a surface boundary extracted from the input image shown in Figure 6.

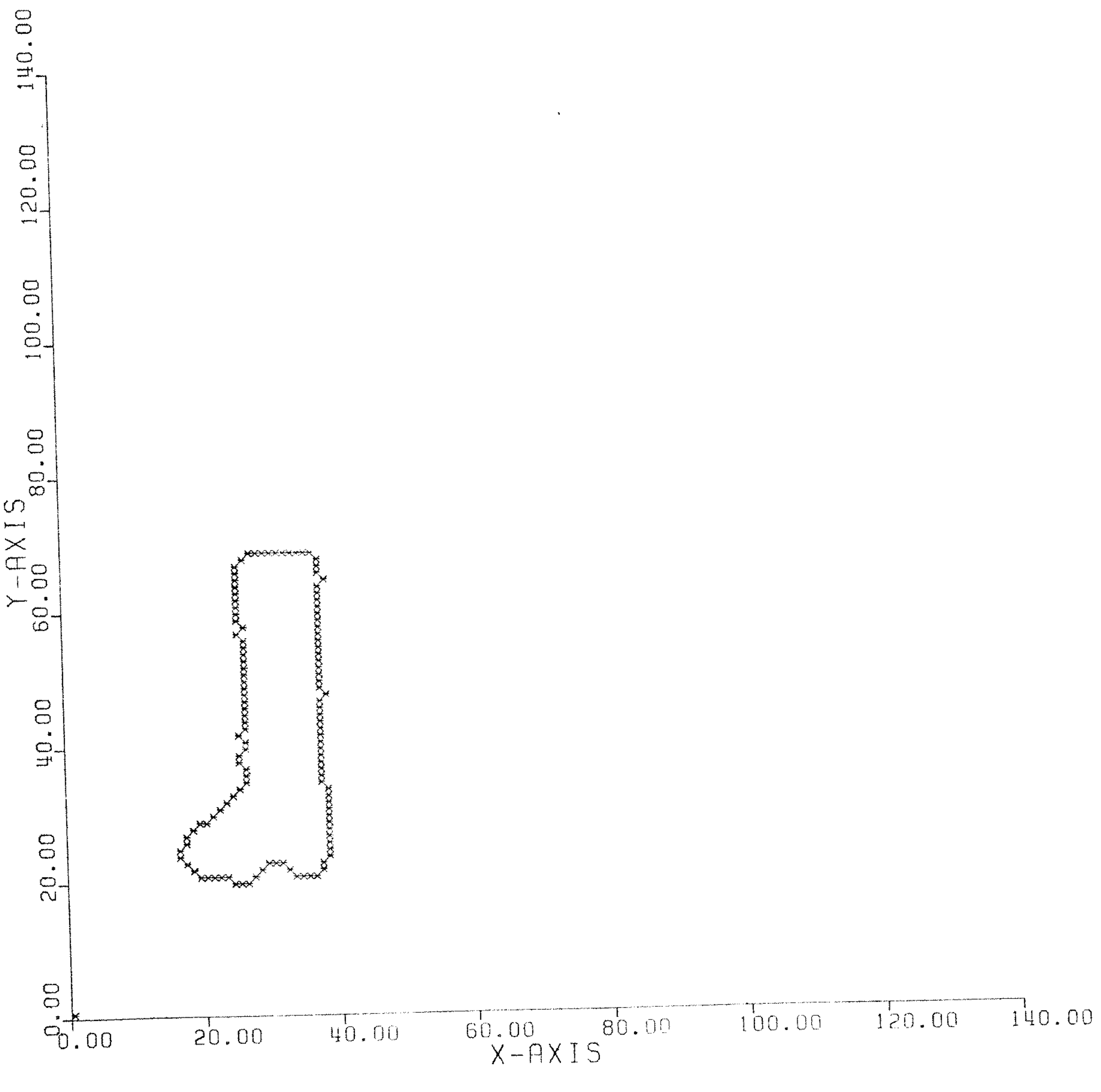

Figure 7c. BOUNDC, a surface boundary extracted from the input image shown in Figure 6.

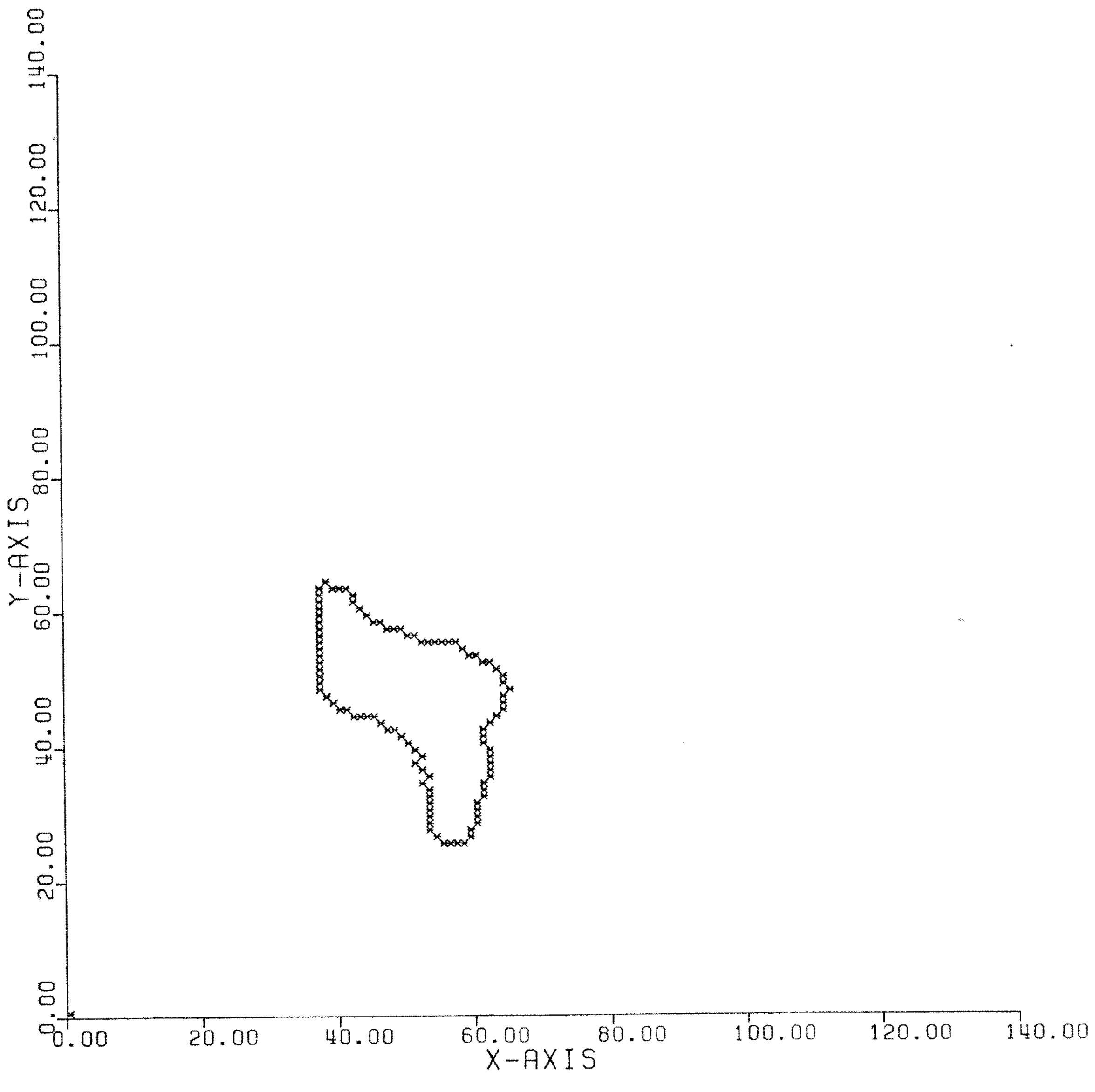

Figure 7d. BOUNDD, a surface boundary extracted from the input image shown in Figure 6.

	INPUT IMAGE SURFACE BOUNDARY		
measured area between code graphs	BOUNDB	BOUNDC	BOUNDD
OBJTYP1	6.18	2.59	6.69
OBJTYP2	8.12	8.71	7.74
OBJTYP3	9.58	6.06	7.54
OBJTYP4	9.06	8.57	7.21
OBJTYP5	5.42	5.41	6.02
OBJTYP6	6.32	5.09	7.21

Figure 8. Results of shape comparison between the library models and the boundaries of the input image shown in Figure 6.

References

1. Yam, K.R., "Computer analysis and recognition of multiple partially occluded objects," Master of Arts Thesis, University of Texas at Austin, December 1979.
2. McKee, J.W., and Aggarwal, J.K., "Finding the edges of the surfaces of three dimensional curved objects by computer," *Pattern Recognition,* vol. 7, no. 1, pp. 25-52, 1975.
3. Freeman, H., "Computer processing of line-drawing images," *Computer Surveys,* vol. 6, no. 1, pp. 57-97, March 1974.
4. McKee, J.W., and Aggarwal, J.K., "Computer recognition of partial views of curved objects," *IEEE Transactions on Computers,* vol. C-22, pp. 780-800, September 1977.
5. Martin, W.N., and Aggarwal, J.K., "Computer analysis of dynamic scenes containing curvilinear figures," *Pattern Recognition,* vol. 11, pp. 169-178, 1979.