Recognition of 2-D Occluded Objects and their Manipulation by PUMA 560 Robot[1]

Bir Bhanu and John C. Ming
UUCS 85-111
August 31, 1985

Department of Computer Science
The University of Utah
Salt Lake City, Utah 84112

Abstract

A new method based on a cluster-structure paradigm is presented for the recognition of 2-D partially occluded objects. This method uses the line segments which comprise the boundary of an object in the recognition process. The length of each of these segments as well as the angle between successive segments comprise the only information needed by the program to find an object's position. The technique is applied in several steps which include segment clustering, finding all sequences in one pass over the data, and final clustering of sequences so as to obtain the desired rotational and translational information. The amount of computational effort decreases as the recognition algorithm progresses. As compared to earlier methods, which identify an object based on only one sequence of matched segments, the new technique allows the identification of all parts of the model which match with the apparent image. These parts need not be adjacent to each other. Also the method is able to tolerate a moderate change in scale and a significant amount of shape distortion arising as a result of segmentation or the polygonal approximation of the boundary of the object. The method has been evaluated with respect to a large number of examples where several objects partially occlude one another. A summary of the results is presented.

Index Terms - Clustering, Occlusion, Recognition, Segment Matching, Sequencing, Shape Matching

Name and Present Address of the Corresponding author:

Bir Bhanu, Department of Computer Science, 3160 Merrill Engg. Building, University of Utah, Salt Lake City, Utah 84112

[1]This work was supported in part by NSF Grants DCR-8506393 and DMC-8502115

Table of Contents

1. Introduction

Recognition of partially occluded objects is of prime importance for industrial automation applications. The problem of occlusion in a two-dimensional scene introduces errors into many currently used vision algorithms which cannot be resolved. Occlusion occurs when two or more objects in a given image touch or overlap one another. In such situations vision techniques using global features to identify and locate an object fail because descriptors of part of a shape may not have any resemblance with the descriptors of the entire shape.

Because we know that occlusion will be present in all but the most constrained environments, other methods have been developed [1, 2, 4, 11, 13]. These techniques can be classified either as boundary based [2, 5, 13, 11] or using the local features (such as holes) if they are available [4]. Many of these techniques are computationally intensive. They can not handle significant distortion in the shape, change in scale, and do not give good matching results over a wide range of industrial objects. Some of these factors led Price [12] to use a conceptually simple technique to solve the occlusion problem by following the *order* of matched segments. He uses a device called a disparity matrix to identify and follow the order of the boundary segments. Unfortunately, this method allows an object to be recognized by only one sequence of boundary segments that are matching in the image. No attempt is made to correlate several sequences of matched segments in order to determine a better matching of the model.

In order to overcome the factors which caused the Price method to fail, we have used a cluster-structure paradigm which provides information about the orientation and position of the objects in the image. Some techniques and structures used by Price have been retained, but the method of matching is entirely new.

The experimental setup for recognizing occluded objects and their manipulation is shown in Figure 1. Before describing how the method works, we give a simple desription of the Price method and detail where the program fails. Following this section, a brief description of the general clustering concept is given, together with the details about how the results can be measured and controlled.

2. Earlier Methods

The Price technique proceeds in the following manner. Initially, the program assumes that we have a linear border approximation of a given model and an image. Price's method then compares every segment in the model with every segment in the image. If the segment pairs are compatible in terms of length and angle between successive segments, the rotational offset between the two segments is entered into the disparity matrix. The entry is indexed by the segment number in the model and the segment number in the image. If the segments are incompatible, the program places an error code at the appropriate location in the matrix. After all line segments have been compared, the matrix contains the offsets, or disparities, for all pairs of line segments. By traversing this newly formed matrix diagonally, the program finds the longest sequence in the matrix that contains compatible entries. From this longest sequence, the Price method then computes the transform dictated by the segment pairs in the sequence. This value is the final result of the procedure.

Unfortunately, the Price technique is very expensive computationally. Since the program must treat every entry in the disparity matrix as a possible starting location, the program traverses the matrix once for every entry that exists. While this fact does not pose a very large problem in the simplest cases, matching takes a long time for models and images with more that about 20 or 30 segments each. Another major problem encountered by the Price procedure concerns the ability to use more than one sequence in the overall matching of the model to the image. The program only uses the longest sequence found in the traversal because it cannot determine the compatibility between more than one sequence. In the cases where a large amount of occlusion is present, Price's technique will not be very successful. Thus, while Price's early attempt to solve the occlusion problem met with limited success, it did not fully deal with the problem.

3. Clustering Techniques

Clustering, in its most general form, groups a set of objects into subsets where objects in a subset are more similar than the objects in other subsets [8, 9]. After every sample has its feature vector computed, the clustering technique (K-Means Algorithm) creates an arbitrary number of K cluster centers, into which all samples will be placed. In order to determine which center to place a given sample in, the program computes the distance from the sample to each of the cluster centers. This distance is merely the Euclidean distance from the sample to the center, using the values in the feature vector. The sample belongs to the cluster center which is closest to it. When every sample has been assigned to a unique cluster center, the program recomputes the value of each of the cluster centers. The new cluster center is the average of all samples which are currently in that cluster. After the new cluster centers have been determined, the program then redistributes all of the samples again, using the new centers this time. The process continues until no further changes take place in the location of the cluster centers. At that point, the samples in each of the clusters are then said to be compatible with each other. Thresholds help to determine when a cluster center becomes stationary, since most features are real numbers and will always change by a minute amount. Also, because the Euclidian distance may be affected by the choice of the features present in the feature vector, the feature values should always be normalized so that each feature contributes equivalently to the overall distance.

The only problem inherent in the clustering technique involves the choice of the number of cluster centers to be used at any given time. Since the choice depends on the structure of the data that is being clustered, the number of centers cannot usually be a constant value for every instance. This fact is probably more true in the computer vision environment than in in any other application. Depending on the lighting conditions, the segmentation techniques used, the amount of occlusion present, and many other factors, the number of cluster centers must be altered. Fortunately, there are measures which can be used to evaluate the success of the clustering with any given number of centers used [3, 6, 7, 10]. These performance measures determine the scattering of the samples within each individual cluster as well as the distance between each of the cluster centers themselves. This information is held in a matrix form known as a scatter matrix [9]. The scattering of the samples in a particular cluster is defined as within-cluster scatter matrix, S_w. The overall position of all clusters in relation to each other becomes the between-cluster scatter matrix, S_b. By definition, the beta value for a certain clustering equals the trace of the within-cluster scatter matrix multiplied by the trace of the between-cluster scatter matrix, i.e., $\beta=Tr(S_w)Tr(S_b)$. As the number of clusters increases, the value of beta will reach a maximum and then slope towards 0. The number of clusters at which the value of beta is a maximum is the desired value and gives the best results. Thus, by setting the number of clusters to be one, clustering the samples, computing the beta value, comparing the beta value with its last value, and continuing until the maximal beta value is reached, the program can find the best number of clusters for any given problem.

The use of clustering and the performance measures in the body of the OCCLUSION program will be discussed in the next section. Because theses techniques can be quickly computed, even when the number of samples is very high, they allow the vision process to.be fast as well as accurate. The clustering method groups all sets of compatible matches into a single cluster, regardless of their position in the image. Thus, this process can find multiple sequences in a model that may be matching in the given image. While the Price method was only able to find a single matching sequence, the new procedure will find as many sequences in the image as possible. Also, since the traversal of the matrices in Price's method took so much time, the new technique improves the speed of the vision process as well.

4. Algorithm Description

The OCCLUSION program uses some of the principles and techniques that were implemented by Price [12]. The disparity matrix structure has been kept as well as the method of finding sequences from a given set of points. These methods were used only because they are efficient and require little time. The program can be divided into the following main steps:

1) Determine Disparity Matrix
2) Initial Clustering
3) Sequencing
4) Final Clustering
5) Transform Computation

Figure 2 indicates the block diagram of the entire clustering algorithm while Figure 3 shows the manner in which the number of cluster centers is selected. Each of these steps will be described individually in this section. The input to the program consists of two sets of data. The first set contains the object model data, which is a set of vertices that define the boundary of the model. This model is the object that we are searching for in the image. The second data set contains the description of the image that has been acquired. This data is also a set of vertices on the boundary that describe the scene that was taken by a video camera.

4.1 Determine Disparity Matrix

The first step of the program consists of the formation of the disparity matrix. This matrix is identical to the matrix used in the Price algorithm [12]. From the set of vertices for the object and the image received by the program, the algorithm determines the segment length of each line and the angles between successive lines. At this point, every segment in the object model will be compared with every segment in the image. If the segment lengths and successor angles are compatible, the program computes the rotational and translational disparity between the pair of segments. The algorithm stores these values in the disparity matrix, indexing the values according to the segment number in the model and the image. This process proceeds until all segments have been compared. At this point, the disparity matrix appears exactly as it would in the Price method. However, since the values in this matrix will be used in the next step, they must be normalized first. The program determines the range of rotational and translation values present in the matrix, and then normalizes every value over their appropriate range. The normalized values are kept in a normalized disparity matrix, since the initial disparity matrix needs to be retained for later use.

The computation time required to complete this step comprises about 10 to 20 percent of the total execution time of the program. Since all of the values must be compared with each other, the exact percentage depends on the total number of segments present.

4.2 Initial Clustering

After all of the normalized values have been placed into the disparity matrix, the algorithm clusters these values. The initial number of cluster centers becomes one. The clustering proceeds as described in the previous section. At each step, the program clusters all of the samples, recomputes the value of the new cluster centers, and continues until none of the cluster centers change their positions. After computing the scatter matrix values for the current cluster results, the algorithm compares the current beta value with the last beta value. If the value has decreased, then the previous beta value and the number of clusters become the final result of this processing step.

This step of the program takes the most computational time of all of the steps due to the large amount of samples that are clustered. For example, if the model contains 25 segments and the image contains 100 segments, the disparity matrix will contain 2500 entries. Out of this number, 1000 samples may be present. If the program has to cluster these samples 3 times until beta is maximized, 3000 distances must be computed. However, this amount of computation is far less than the comparable computation that would need to be done by the Price method.

After the number of clusters have been determined and the results are known for that particular value, the program selects the cluster with the largest number of samples. The data in this cluster will be used by the rest of steps in the algorithm to determine the location of the object. However, since some of the other clusters may contain approximately the same number of samples as the largest one, the program also uses any cluster which is within 20% of the largest cluster. Each cluster is considered separately and the final transform comes from the cluster which yields the highest confidence level. Thus, the program now passes each cluster that has been selected to the following algorithm steps, one at a time.

4.3 Sequencing

Since the clustering results provide no information concerning the physical structure of the model, this information must be provided at this time. Using the samples in the current cluster, the program finds all sequences in these samples. For instance, if the first sample indicates that segment 1 in the model matches segment 27 in the image (represented by the notation [1,27]), the program then searches for the pair [2,28], since this pair should logically follow the first pair on the borders of the model and the image, respectively. Since there may be some missing and extra segments in the polygonal approximation of the model and the image as a result of segmentation and various other reasons, we allow up to 2 extra or 2 missing segments when locating the sequences. This procedure continues until all possible sequences have been located in the data of the current cluster. This step provides the only structural information within the algorithm and cannot be omitted.

Any samples in the current cluster which were not placed in any sequence should be discarded. Since these points are not members of any sequence, they usually represent the extraneous data present in many of the clusters. The program should also remove any sequences which have a segment count of less than three (Three segments comprise the basic local shape structure). This removal insures that arbitrary data included in the initial clustering and sequenced by the current step is not included in the final processing steps. Because of their small length, these sequences are assumed to be invalid. Even if the sequences did indicate valid matches, their removal from the set of sequences does not introduce any error into the final matching that will be computed.

The final task to be accomplished at this step in the algorithm is the computation of the rotational and translation averages of each sequence that has been located. These averages are merely the averages of all of the samples that are present in each sequence. These sequences and their averages will be used in the final clustering step of the program.

The sequencing step requires the second largest amount of execution time within the entire program. Since it is still very costly to check the possibility of a sequence occurring at any given point, the program must check every point in order to locate the best choices. However, because the clustering results have greatly reduced that amount of choices that need to be checked, this step takes far less time than does the Price method. It is a one pass algorithm over the data.

4.4 Final Clustering

Using the sequences and the sequence averages obtained from the previous step, the algorithm clusters theses values to find those sequences which lead to the same rotational and translational results. As with the initial clustering, the program uses the iterative technique of clustering, evaluating, clustering, etc. After the value of beta has reached its maximum, the program again selects the cluster which contains the largest number of sequences and passes this cluster to the final program step.

While the initial clustering step had to deal with a large number of samples, this step of the program uses a trivial portion of the total program time. In almost all cases, the number of sequences will be less than 100, with an average somewhere near 10 to 20. Also, since the sequencing step has eliminated a good deal of the erroneous data, the beta value quickly reaches it maximum and this steps ends.

4.5 Transformation Computation

After all clusters which were selected have been sequenced and clustered a second time, the program determines the confidence level of the transformation determined by each cluster. The cluster with the highest confidence level is selected as the final transformation cluster. The program assembles the set of matched segments included in the sequences in this cluster. These segments should be sorted into increasing model segment number so that the sequences will indicate successive segments around the object boundary. The final output of the program is the rotation and the vertical and horizontal translation necessary to locate the object model within the image. The program also produces a confidence level which indicates the likelihood that the final matching is correct. The confidence level is found by dividing the cumulative length of all segments in the final matching by the total length of all segments in the object model. So, if the confidence factor is 80/200, we are 40% sure of the program results. This factor will be used by later versions of the program to decide if further processing should be done in order to insure the proper results. Confidence levels of 10 percent or more usually lead to the correct transformation.

5. Experimental Results

For the purposes of this experiment, a group of tools shall be placed in a pile on the surface of a white table. A camera will take a picture of this scene, the image will be digitized, the objects will be located, and a robot will remove the tools from the pile and place them in a box at a predetermined location. See Figure 1.

The process of recognizing and manipulating the tools should proceed in the following manner:

1) Model Specification
2) Image Acquisition
3) Polygonal Approximation
4) Object Recognition
5) Object Manipulation

Step number one should only need to be performed once for each tool which could be present in any of the images, while steps two through five will be performed once for each scene that is acquired. Each of these steps will be described separately in the following sections. However, the hardware needed for this system will be described first.

5.1 Hardware Requirements

The hardware requirements for the recognition system can be very simple or very complex, depending on the budget available to the user. The system described here assumes that the user has a limited budget and must work within these budget constraints. The hardware required may be divided into the following groups:

1) Camera and digitization equipment
2) Microcomputer for image processing
3) Robotic system for object manipulation

Each of these groups will be described individually in the paragraphs which follow. Figure 1 shows the overall layout of each of the components described below.

Camera and Digitization Equipment: For the purpose of image acquisition and digitization, we shall use an Apple Macintosh computer equipped with a MacVision digitizer and a Panasonic black and white camera. The Macintosh computer has been chosen for the task of image acquisition due to the ease of use of the Macintosh operating system as well as the portability of the unit. The MacVision digitizer and associated software is quite inexpensive and yields very good results in the form of a binary image. Gray level images are not available with this particular digitizer, although future versions should be equipped to produce them.

The camera will be mounted above the table on which the tools are placed and will remain stationary. Since the surface of the table is white, the necessary contrast between the object and the background should be achieved without much difficulty. If necessary, the lighting conditions of the room will be modified in order to correct any problems. The MacVision digitizer produces a binary image which can be saved onto floppy disk and then transferred to the microcomputer when necessary.

In order to provide a standard file format for the digitized picture, the images collected by the MacVision digitizer must first be converted to a MacPaint format. This file can then be easily transferred to the microcomputer in a very compact form. The file transformation can simply be accomplished by loading the MacVision file into the MacPaint program directly and then saved back out to disk in the standard MacPaint format. The MacPaint format is fully described in Appendix II. This file will then be sent to the image processing computer with the use of a modem or with a dedicated line.

Microcomputer for Image Processing: Due to the facilities which are available to this experiment at the present time, the computer used for image processing will be a Vax 11/750. The MacPaint files will be transferred to this computer via modem or dedicated line. After the files have been received, the algorithms used to recognize and locate the objects in the image will be used to find the location of each of the tools. The output of the programs will be the location of each of the tools in the pile, assuming all of them have been recognized. This information will then be passed to a robotic system for the manipulation of the tools.

Robotic System for Object Manipulation: After the location of each of the objects has been determined by the image processing step, the robotic system will take this information and retrieve each of the tools. The tools will be placed in a box which will lie next to the table on which they are placed. The robotic system which is available for use in this experiment is a Puma 560 robot with the necessary software (VAL II) used to control the movements of the robot. This robot is interfaced with the Vax 11/750 computer, so the transfer of object location information should be trivial.

5.2 Model Specification

In order to determine the ability of the program to find objects in an occluded scene, a set of 14 models was obtained and used in the matching algorithm. The models consist of a set of tools such as a hammer, screwdriver, pliers, wrench, and so on. The specification of the models of the tools which will be recognized is the most critical portion of the experiment. Each tool should be placed on the table individually and the image of this scene should then be processed. The user may have interactive control over this process if necessary to achieve good results at this stage. The representation of the tool should be very clear and well defined in terms of the segments which represent the object.

After the model image has been received, and before the tool has been removed from its position on the table, the user must also teach the Puma robot the initial position of this tool on the table. This will be done by using the teach pendant of the Puma to specify an approach point to the object as well as the actual grasping point of the tool. Because occlusion may cause any given grasping point on the tool to be missing, the user should specify three or four different locations which could be used to pick up the tool.

The final model description of each tool will consist of the boundary representation of the tool as well as the position of each of the grasping points on the tool which could be used by the robot. The segments which determine the border of each of the tools are obtained using the same border approximation algorithms that are described in the image acquisition section below.

5.3 Image Acquisition and Transferal

With the use of a Macintosh computer and MacVision, a commercially available digitizer, the process of image acquisition and digitization has been greatly simplified. The MacVision digitizer uses a standard video camera as the video input to the system. The digitizer can adjust the size of the image which is seen, depending on the choice of the user. The default size of the image is 200 by 200 pixels. This picture is displayed in a window on the Macintosh screen. The second option is a full Mac screen display of the video input. This image is 576 by 480 pixels in size. The time required to digitize the image depends on the size of the image chosen. For the smaller, window-sized image, the digitization time is approximately 5 seconds, while the full screen image takes about 22 seconds to complete. The current implementation collects images using the full screen format. Obviously, this system is only useful for images that remain stationary for the duration of the image acquisition period. One drawback to using the full screen option is that MacVisiion distorts the image received from the camera due to the fact that it places a 512 by 512 image onto the 576 by 480 screen display. Thus, the image is stretched horizontally to some extent.

The image that is digitized by MacVision is saved onto the floppy disk of the Macintosh. To provide a standard format that can be decoded on the VAX computer, the MacVision file must first be converted to a MacPaint file. This is accomplished by loading the file into the MacPaint program using the standard open file command. Once the file has been loaded into MacPaint, the user must simply save the file out to disk again. However, this time the file is saved as a MacPaint file and can be decoded properly.

The MacPaint files that have been saved on floppy disk must be transferred to the Vax computer to finish the rest of the low level processing. This process is normally done using a modem connecting the Macintosh to the Vax. After all of the files have been loaded into the Vax, the remainder of the processing takes place on the mainframe. Thus, the Macintosh merely serves as a convenient image acquisition tool, suitable for a wide range of applications. Since the Macintosh is portable, it is also very useful in obtaining images that cannot be collected directly in a laboratory environment.

Comments: The procedure described above provides the user with a very simple, inexpensive way of obtaining images which can be used by the algorithms to accomplish image analysis. The ease of use of the Macintosh user interface allows the user to collect a set of images in a very short time. The only drawback to image collection is the problem of lighting conditions needed by the MacVision system in order to provide a high contrast image. The images collected for this set of examples were obtained by using backlighting, which provides a very sharp contrast between the objects in the image and the background. Since the polygonal approximation step processes the MacPaint image directly, the quality of the image must be very good. Otherwise, the user may need to perform some pre-processsing of the image in order to obtain the necessary image quality or to use more elaborate segmentation techniques.

5.4 Polygonal Approximation

Once the image file exists in the Vax computer, the remaining processes are all completed using the available facilities. In order to create the border segments which comprise the outline of the objects in the image, the images undergo the following steps:

1) Data Decompression
2) Object Border Location
3) Initial Border Approximation
4) Final Border Approximation

Each of these steps is now described.

Data Decompression: The first processing step that must take place is to decode the information in the MacPaint file which has been uploaded from the Macintosh. The data in the file is compressed and must be expanded in order to obtain the pixel values needed by the remainder of the program. The decompressed size of the MacPaint image is 720 rows by 576 columns. See Appendix II for a complete description of the MacPaint file format. After the file has been decoded, the pixel values of 0 or 1 are stored in a character array of 720 by 576.

Object Border Location: In order to locate and represent the tools that are in the image, the program must first locate the border of the objects in the image. This is accomplished using a simple border following algorithm that follows and marks the border (8-connected) of the object in a clockwise direction within the image. Once the border following algorithm locates the borders of objects in the image, the remainder of the program operates only on the pixels that lie on this border.

The number of boundary points for the models range from 567 to 1425 pixels, while the number of boundary points in the occluded images vary from 1123 to 4025.

Initial Border Approximation: After the borders of the objects have been located, the program determines a rough polygonal approximation of the object by finding the points along the boundary that have the largest local curvature maximas. This is done by first computing the curvature for every point along the boundary of the object. Once these values have been determined, the program uses a smoothing factor to select the points on the boundary that have the maximum local curvature. The smoothing factor is used to control the minimum distance that these curvature maxima points must be from each other. Experience has shown that the smoothing factor must be a value between 6 and 26. Smoothing factors above or below this range give results which are either too coarse or too fine, respectively. Another important point to note is that smaller smoothing factors, which give better approximations at this stage, lead to a rougher approximation after the final stage of low level processing. For instance, in one case a model was run with a smoothing factor of 8 and the final border approximation contained 24 points. When the same model was run with a smoothing factor of 24, the final approximation had 32 segments. This apparent contradiction has to do with the split-merge algorithm's ability to reduce the total error of the approximation, based on the initial approximation of the object which is received. This factor will be discussed further in the next section.

A smoothing factor of 8 was selected for the models in these examples, while the images were processed with a smoothing factor of 24 to insure a good polygonal approximation of the occluded scenes. The initial border approximations for the models range from 18 to 52 segments in length. The images contain from 21 to 71 initial segments.

Final Border Approximation: This stage is the last step of the low level processing. The input consists of the border points found in the border following step as well as the initial border approximation that was found using the curvature maxima algorithm. The program uses the split-merge algorithm to refine the curvature maxima approximation and also makes any end point adjustments that lead to a smaller total approximation error for the object.

The split-merge algorithm first takes all of the curvature maxima points and places these values into a linked list so that changes to the border can be quickly updated. The next step is to compute the error associated with each line segment in the initial approximation. The sum of the individual segment errors yields the total approximation error of the curvature maxima results. Once these values are known, the split-merge algorithm proceeds as follows:

Step 1 - After computing the pointwise error for every segment in the initial approximation, split the segment with the largest error into two segments and compute the new error for each of these segments. Continue to split the segments with largest error until the total error falls below some specified error value. This value is referred to as the *maximum error* value.

Step 2 - Repeat steps 2a and 2b until no further merges or adjusts take place:

Step 2a: Compute the error that would result if adjacent segments on the object boundary were combined together. Select the segment combination which causes the total error to remain below the *maximum error* value and also causes the smallest increase in total error and merge these segments together. Compute the resulting total error.

Step 2b: Compute the error that would result if each border approximation point were shifted one border location to the left or to the right. If any of these shifts lead to a decrease in the total error, shift the point accordingly. Compute the resulting total error.

Unfortunately, it is very difficult to determine the appropriate value for the *maximum error* for a wide range of differing examples. Attempts were made to relate the *maximum error* to the number of boundary points, the number of initial approximation points, or the initial value of the total approximation error. While these attempts proved successful in a small portion of the examples, none appeared to work well for every example. This fact is due primarily to the wide range of initial approximations produced by the curvature maxima step. For a given low value of the maximum error if the approximation is very good to begin with, the splitting step from above will have to split a *large number* of segments in order to substantially reduce the level of the approximation error. Conversely, if the approximation is very bad, splitting may not lead to a polygonal approximation with the necessary resolution for matching.

In order to overcome this problem, one has to closely examine the reduction in total error as the split procedure splits a given approximation of an object. Assume that the approximation is initially too coarse, which is achieved with a larger smoothing factor. As the split procedure splits the segments with the largest error, the total error value is quickly reduced. However, the splitting procedure eventually reaches a point at which any further splits cause only a very slight decrease in the total error, or which in some cases may lead to a very small increase in the total error, due to the structure of the data being approximated. Once this point has been reached, the use of any more splitting to achieve a better approximation is somewhat futile. Therefore, the program decides to stop splitting when it has counted 20 of these very slight increases in the total error value. At this point, the value of the *maximum error* is assigned to be the total approximation error of the current representation. The rest of the split-merge algorithm proceeds as described in step 2 in the above description.

Since the amount of splitting depends on the initial approximation of the object given by the curvature maxima algorithm, the selection of the appropriate smoothing factor is crucial in obtaining the best results from the split-merge algorithm. For example, if the smoothing factor is very low, the amount of splitting to be done will be quite small. However, since the curvature maxima approximation may not always be a good representation of the object, split-merge may not produce the best results. This is why the smoothing factor is set at a relatively high value so that the split-merge procedure can refine the representation in the appropriate manner.

The final output of the polygonal approximation step is the list of boundary approximation points (vertices) which split-merge produces. This list of points is written to a data file, which is used by the clustering algorithm in the matching process.

The final number of border segments received from the split-merge procedure range from 5 to 33 segments for the models and 26 to 71 segments for the occluded images.

Figure 4 shows the images collected for the 14 models used in this experiment. Figure 5 depicts the final polygonal approximation for each of these models produced by the split-merge algorithm. Figure 6 indicates, in chart format, some of the important values and data created in the polygonal approximation step for the models collected. Similarly, Figures 7, 8, and 9 show the polygonal approximation results for the 20 images that were used for the experiment. Tables 1 and 2 summarize the numerical data from the models and the images, respectively.

Comments: The image processing algorithms described above provide a very good approximation of the images that we have collected. As noted earlier, the smoothing factor is the only critical parameter that must be adjusted in order to obtain the proper resolution in the object representation. Experimental use of this parameter is the best means of determining the proper value for any given application.

Overall, the program has given very good results for the examples that we have used. Although the execution speed is not exceptionally good, the algorithms could very possibly be implemented in VLSI, which would allow the programs to work in a much shorter time.

5.5 Model Based Recognition

Once all of the models and images have been collected, the clustering algorithm can then be used to locate the models in the images. When the clustering program was run on the 20 images that were collected, the results were very good. Of the 56 models present in these images, 40 (71%) models were correctly matched. 4 of the 56 models were mistakenly matched to a different model. The remaining 12 model instances could not be matched.

Figure 10 shows the matching results for the twenty images that were shown in Figure 7. Solid red lines show the polygonal approximation of the images using the split-merge algorithm. The blue and green dotted lines show the polygonal approximation of the model at its matched location in the image. The green lines indicated the segments which were matched while the blue lines show the segments which did not contribute to the matching. Figure 11 indicates the results of the matching in graph format. Tables 3 and 4 summarize the actual verses experimental transformation values and the overall matching results, respectively.

Execution times for the clustering method range from .5 to 3.2 seconds. The confidence levels for the matched objects vary from 0 to 98%. The error analysis of the matchings which were correct yield a mean rotational error of -0.14 degrees and a standard deviation of 8.68 degrees. The mean translational errors are -11.83 and -7.65 pixels for x and y, respectively. The standard deviations of these values are 57.41 and 44.79.

5.6 Improved Model Performance

Of the 12 models that were not located in the images, the failure to find these objects is due to the substantial difference in the polygonal representations of the particular tool in the model and in the image. When the polygonal approximations of the object become too diverse, clustering is not able to overcome this problem. However, if the representation of the model is improved within the image, matching will occur and the transformation determined by the program will be better. This fact is readily seen in Figure 12. The model for the wrench (Fig. 4(g)) has been modified to correspond more closely to its representations in the images shown. Notice that the wrench matches in every image now and that the transformations that have been computed are better than the transforms obtained with the initial representation of the model.

In order to determine the ability of the clustering algorithm to perform on models and images with a large number of boundary segments, the program was run using some hand generated examples of some tools. The polygonal approximation of a screwdriver, a pair of pliers, and an occluded image containing the two tools was manually created. These examples are shown in Figure 13. The results of the clustering algorithm on these examples is presented in Figure 14. The screwdriver and the first representation of the pliers have been correctly matched. However, the second representation for the pliers has been incorrectly matched, as shown in Figure 14(c).

5.7 Comparison with Price Method

To contrast this new method with Keith Price's earlier work, we have run Price's algorithm on several of the examples used above. Figure 15 shows the results of this matching, which can be compared with the clustering results in Figure 10. The clustering method, which obviously yields far better results, also finds the matches in much shorter time. The model transformations also tend to be better because clustering can find several sequences along a boundary which may contribute to the final transformation. Price's method, on the other hand, may only select the longest sequence. Figure 16 indicates the results of the Price algorithm on the hand generated examples in Figure 13. Note that the algorithm was not able to match the second representation of the pliers at all.

5.8 Object Manipulation

After the transformation of each of the tools have been determined, this information will be sent to the Puma robot. Since the robot already knows the initial location of each tool from the model specification stage, the rotation and translation values can be used directly in a simple coordinate transformation to find the actual location of the tools on the table. Using a list of the segments which were matched in the image processing stage, the Puma can decide which of the grasping points can be used to grab the tool. Some simple tactile sensing can also be employed in order to insure that the tool has indeed been grasped. After each of the tools has been removed, they will be placed in a box at another predetermined location near the table. This final step of the experiment is currently underway. We are awaiting some final pieces of hardware in order to fully interface the Puma robot with the Vax computer.

6. Conclusions and Future Modifications

The cluster-structure paradigm allows the problem of occlusion to be easily compensated for and overcome. Since the clustering technique does not limit itself to a single sequence of line segments on the border of an object, the program can locate all of the matched segments of the model, which accounts for the high success rate. Although the program was not highly successful in the instances of severe occlusion, even the human interpreter would have problems locating some objects within these images.

Applications of this method will be limited to areas in which the environment can be partially constrained, since lighting conditions tend to be the most critical. However, most industrial applications in which vision is presently used already have this constraint applied. No further adaptations within the working environment should be necessary to allow this new technique to be used.

Future modifications to this algorithm will include the ability of the program to handle the situations when the model has a large amount of symmetry, so that the possible transform could be two completely equal values, both of which are correct. These problems are currently being investigated.

I. Appendix - Detailed Price Algorithm

This appendix provides a detailed description of the Price technique for occluded image vision processing. The algorithm is divided into 5 major divisions, which are indicated below. The English-style pseudo-code which follows describes the flow of the Price algorithm as we have implemented it.

```
#1. Program Initialization.

    1a. Set all threshold values and graphics parameters to their
        default values.
    1b. Input the program mode from the user at the terminal.
    1c. Input the graphics capability from the user at the terminal.
    1d. Print the main menu for the user.  Retrieve menu choice.
    1e. If algorithm complexity option is chosen, process user input
        as to choice of complexity and set the global flag accordingly.
        Read in the value of the number of starting points for matrix
        traversal, if necessary for the given input choice.
    1f. If algorithm parameters option is chosen, process user input
        as to choice of parameter and assign parameter to the input value.
    1g. If graphical parameters option is chosen, process user input
        as to choice of parameter and assign parameter to the input value.
    1h. If exit option is chosen, issue UNIX system exit command and
        halt program execution.
    1i. If choose data files option is chosen, proceed to step #2.

#2. Select and Process Data Files.

    2a. Input the model data file name from the user.
    2b. Attempt to open "filename.seg" file. If no error, read file data.
    2c. If error in 2b, attempt to open "filename.pts" file.  If no
        error, read file data.  Else, ask user to input new file name
        or enter data from the terminal.
    2d. Repeat steps 2a-2c for the apparent image data file name.
    2e. If .pts file was read for the model image, determine the following:
             i) Segment length between consecutive points in the data.
            ii) Angle between current segment and next clockwise segment.
        Store this data and also save the data in the file "filename.seg".
    2f. Repeat step 2e for the apparent image, if necessary.

#3. Determine Initial Disparity Matrix and Initial Transforms.

    3a. Compare every segment in the model data with every segment in
        the apparent image data.  If the length and the apparent image
        thresholds are met, enter orientation difference in the matrix
```

indexed according to segment number. If thresholds are not met, enter error number (value < 0).

3b. If traversal model = exhaustive, then for every value in the matrix > 0, use that location as the starting point and do the following: (Loop exits when current row number becomes row number of the starting point)
   i) If the current location is > 0 and the threshold between the present value and the starting point value is met, enter the location of the point into the matched segments array. Step down and to the right in the matrix by one and goto step i.
   ii) Otherwise, step one position to the left, or one position up, or two positions to the left, or two positions up, until a value meeting the same criteria as above is found. When found, enter the location of the point into the matched segments array. Step down and to the right and goto step i.

3c. Otherwise, if length option was chosen, sort model segments according to length. Use the longest segment as the starting point for traversal and goto 3b. Repeat with next longest until the number of starting points defined by the user = 0.

3d. Otherwise, if predecessor angle option was chosen, sort the model-apparent image segment pairs according to the difference between predecessor angles. Use smallest difference segment location as the starting point for traversal and goto 3b. Repeat with the next smallest until number of starting points defined by the user = 0.

3e. Determine the longest matched segment in the matched segment array. Compute the initial transform from this segment by the following:
   i) The rotation equals the value entered in the matrix at the location of the matching pair.
   ii) The x and y translations equal the average difference between the endpoints of the current segment and its match after rotating the segment by the value from i.
   iii) Transform the entire model image according to the values from i and ii. Store the new image.

3f. Repeat 3e for the second longest matched segment.

3g. Segment the new image data as in 2e. Store this data also.

#4. Transformation Refinement.

4a. Compare every segment in the new model with every segment in the apparent image. If the position threshold is met, enter the ratio of positional difference over combined segment length into the new matrix indexed according to segment number. If not, enter the error code (value < 0).

4b. For every value in the new matrix > 0, use that location as the starting point and do the following: (Loop exits when current row number becomes row number of starting point)

   i) If the current location is > 0, enter the location of the point into the matched segments array. Step down and to the right in the matrix by one and repeat.

   ii) Otherwise, step one position to the left, or one position up, or two positions to the left, or two positions up, until a value meeting the same criteria as above is found. When found, enter the location of the point into the matched segments array. Step down and to the right and goto step i.

4c. Repeat 4a-4b for the second new model from step 3.

4d. The matched segments array contains the final set of matching segments computed by the algorithm.

#5. Evaluating the Results.

5a. If the ratio of the number of segments matched from #4 over the total number of segments in the model is greater than the confidence threshold, goto 5b. Else goto 5d.

5b. Determine the longest matched segment in the matched segment array. Compute the final transform from this segment by the following:

   i) The rotation equals the initial value of the rotation plus the orientation difference of the longest segment with respect to its matching segment in the apparent data.

   ii) The x and y translations equal the average difference between the endpoints of the current segment and its match after rotating the segment by the value from i.

   iii) Transform the entire model image according to the values from i and ii. Store the final image.

5c. Display the graphics or the data menus to allow the user to view the final data. When the user selects the quit option, goto 5d.

5d. Reinitialize all global variables. Goto #1.

II. Appendix - MacPaint Document Format

MacPaint documents use only the data fork of the file system; the resource fork is not used and may be ignored. The data fork contains a 512 byte header and then the compressed data representing a single bitmap of 576 pixels wide by 720 pixels high. At 72 pixels per inch, this bitmap occupies the full 8x10 inch printable area of the Imagewriter printer page.

Header: The first 512 bytes of the document form a header with a 4 byte version number (default=2), then 38*8=304 bytes of patterns, then 204 unused bytes reserved for future expansion. If the version number is zero, the rest of the header block is ignored and default patterns are used, so programs generating MacPaint documents can simply write out 512 bytes of zero as the document header. Most programs which read MacPaint documents can simply skip over the header when reading.

Bitmap: Following the header are 720 compressed scanlines of data which form the 576 wide by 720 high bitmap. Without compression, this bitmap would occupy 51840 bytes and chew up disk space pretty fast; typical MacPaint documents compress to about 10 Kbytes using the PackBits procedure in the Macintosh ROM to compress runs of equal bytes within each scanline. The bitmap part of a MacPaint document is simply 720 times the output of PackBits with 72 bytes input.

If the bitmap is to be decompressed without using the PackBits procedure in the Macintosh ROM, the following procedure may be used:

Any run of three or more equal bytes is compressed into a count byte and a single data byte. Runs of unequal bytes are passed on literally, preceded also by a count byte.

```
<count byte><data byte> ... count = -1...-127 -->

        replicate byte 2...128 times

<count byte><n data bytes> ... count = 0...127 -->

        copy 1...128 bytes uncompressed
```

References

[1] N. Ayache.
A Model-Based Vision System to Identify and Locate Partial Visible Industrial Parts.
In *IEEE Conf. on Computer Vision and Pattern Recognition*, pages 492-494. June, 1983.

[2] B. Bhanu and O.D. Faugeras.
Shape Matching of Two-Dimensional Objects.
IEEE Transactions on Pattern Analysis and Machine Intelligence PAMI-6(2):137-155, March, 1984.

[3] B. Bhanu, A.S. Politopoulos and B.A. Parvin.
Intelligent Autocueing of Tactical Targets.
In A. Oosterlinck and P.E. Danielsson (editor), *Architecture and Algorithms for Digital Image Processing*, pages 90-97. August, 1983.

[4] R.C. Bolles and R.A. Cain.
Recognizing and Locating Partially Visible Objects: The local-Feature-Focus Method.
The International Journal of Robotics Research 1(3):57-82, Fall 1982.

[5] W.K. Chow and J.K. Aggarwal.
Computer Analysis of Planar Curvilinear Moving Images.
IEEE Trans. Computers C-26:179-185, February, 1977.

[6] G.B. Coleman.
Image Segmentation by Clustering.
Technical Report USCIPI Report 750, Image Processing Institute, University of Southern California, Los Angeles, California, 1977.

[7] E. Diday.
Problems of Clustering and Recent Advances.
Technical Report 337, IRIA Laboria, Domaine de Voluceau, Rocquencourt, France, January, 1979.

[8] R. Dubes and A.K. Jain.
Clustering Techniques: The User's Dilemma.
Pattern Recognition 8:247-260, 1976.

[9] R.O Duda and P.E. Hart.
Pattern Classification and Scene Analysis.
John Wiley & Sons, 1973.

[10] K.S. Fu and T.Y. Young, Eds.
Handbook of Pattern Recognition & Image Processing.
Academic Press, 1985.
Cluster Analysis, Chapter by A.K. Jain.

[11] M.W. Koch and R.L. Kashyap.
A Vision System to Identify Occluded Industrial Parts.
In *IEEE Int. Conf. on Robotics and Automation*, pages 55-60. March, 1985.

[12] K.E. Price.
Matching Closed Contours.
In *Proc. 7th Int. Conf. on Pattern Recognition*, pages 990-991. July-August, 1984.
Also in Tech. Report 104, Intelligent Systems Group, University of Southern California, Los Angeles, October 19, 1983, pp. 29-37.

[13] J.L. Turney, T.N. Mudge and R.A. Volz.
Recognizing Partially Occluded Parts.
IEEE Trans. on Pattern Analysis and Machine Intelligence PAMI-7:410-421, July, 1985.

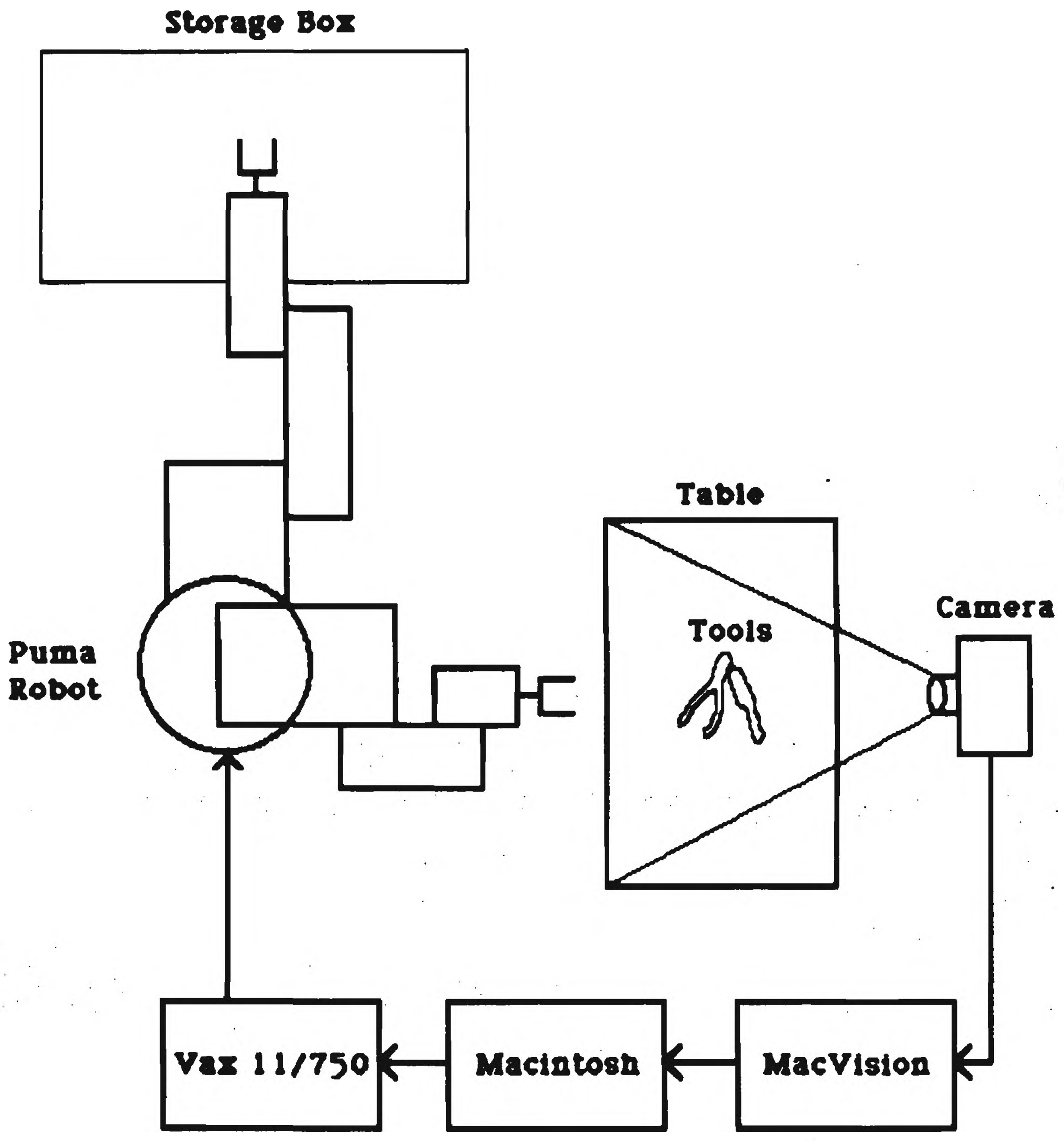


Fig. 1 System diagram for the recognition of 2-D occluded objects and their manipulation by PUMA 560 robot.

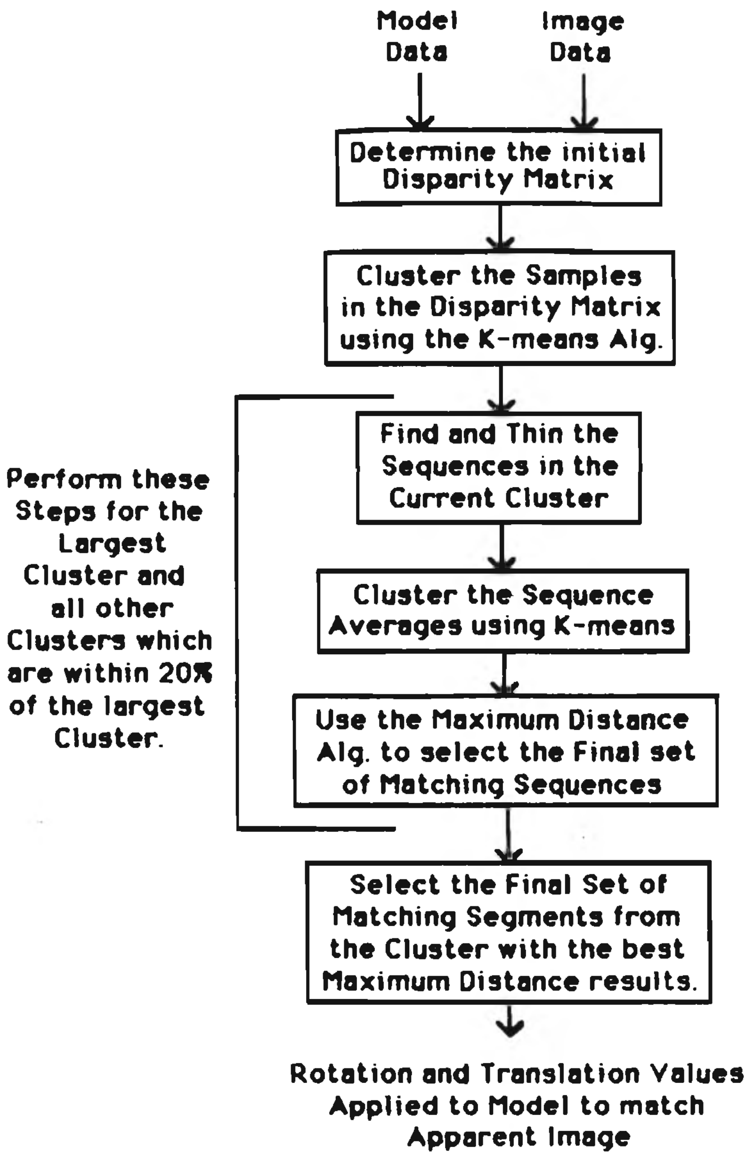

Model Data
Image Data
Determine the initial Disparity Matrix
Cluster the Samples in the Disparity Matrix using the K-means Alg.
Find and Thin the Sequences in the Current Cluster
Cluster the Sequence Averages using K-means
Use the Maximum Distance Alg. to select the Final set of Matching Sequences
Perform these Steps for the Largest Cluster and all other Clusters which are within 20% of the largest Cluster.
Select the Final Set of Matching Segments from the Cluster with the best Maximum Distance results.
Rotation and Translation Values Applied to Model to match Apparent Image

Fig. 2 Block diagram of the clustering based occlusion algorithm.

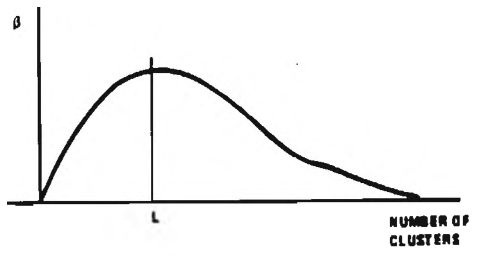


Fig. 3 A clustering quality measure.

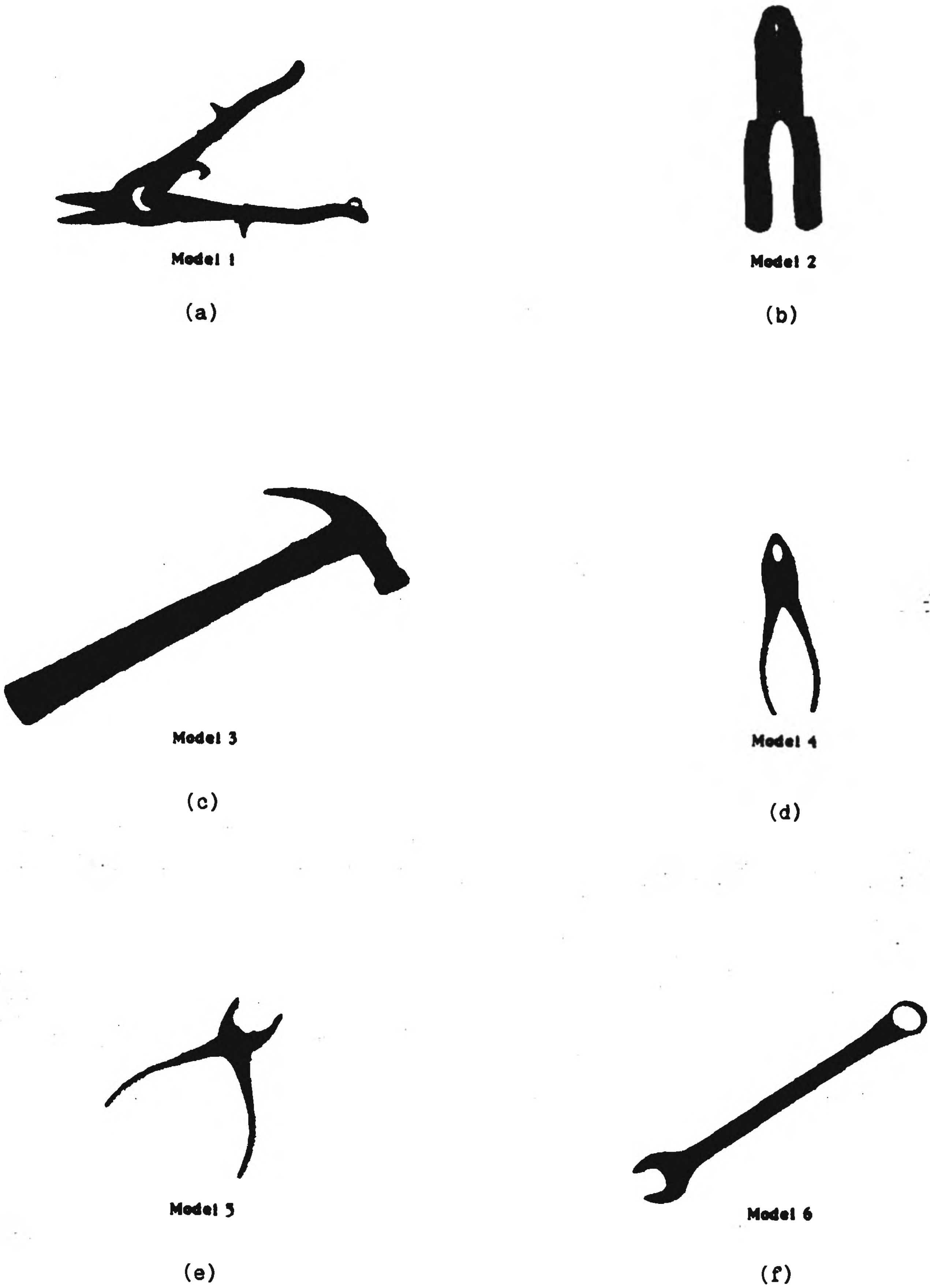
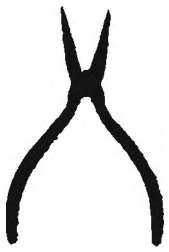


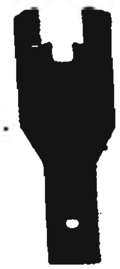
Fig. 4 Images of the object models (a to n).

(g)

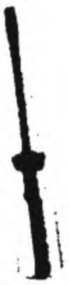


(h)

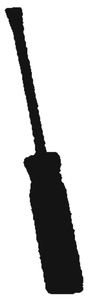
(i)



(j)



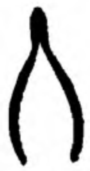
(k)



(l)

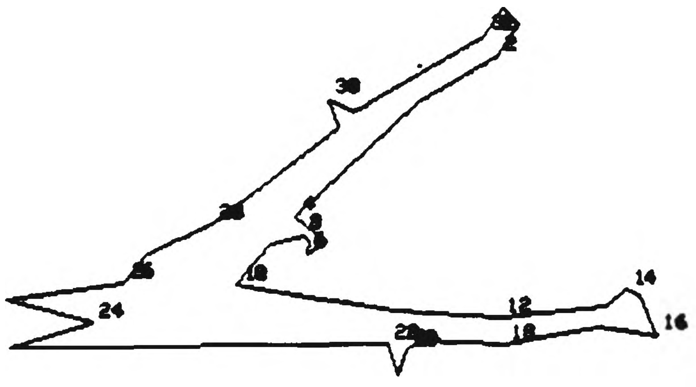
Fig. 4 (Continued)

(m)

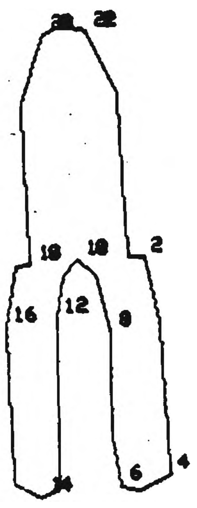


(n)

Fig. 4 (Continued)

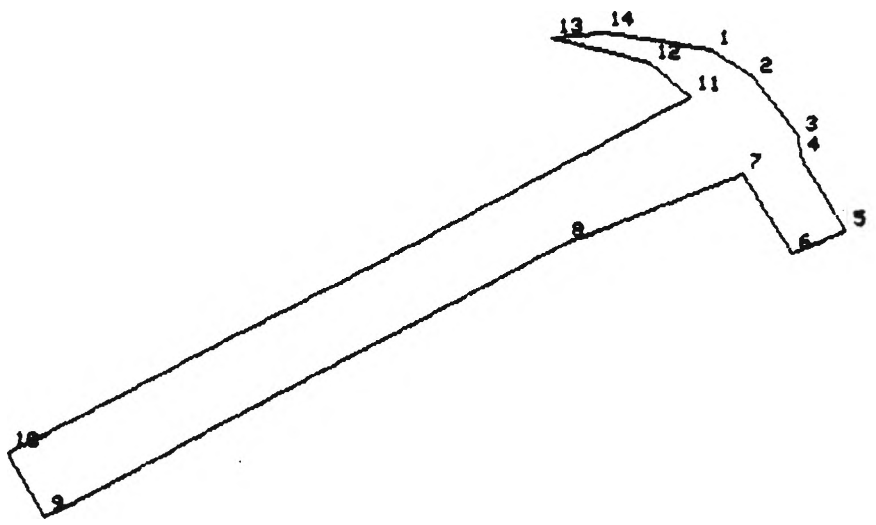


(a)



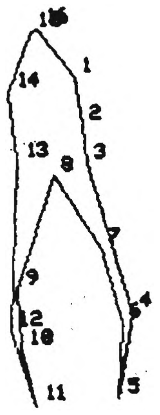
(b)

Fig. 5   Polygonal approximation of the object models (a to n).



Model_3

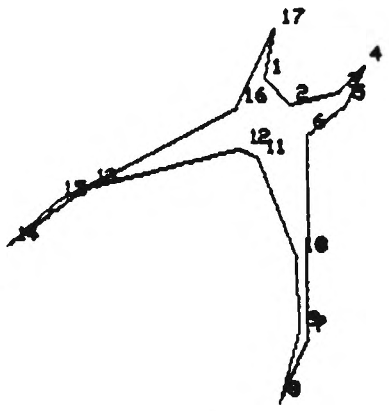
(c)



Model_4

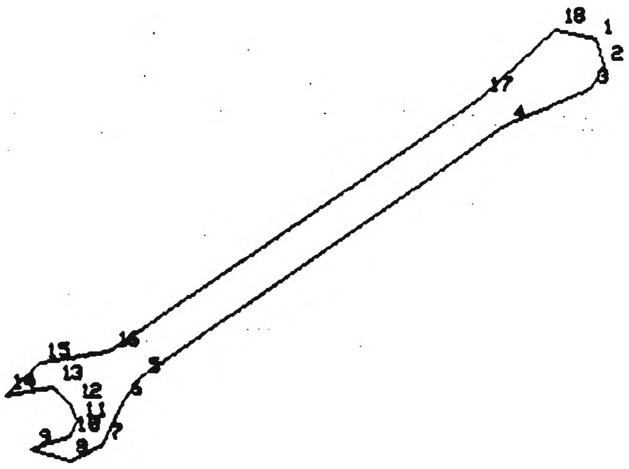
(d)

Fig. 5 (Continued)



Model_5

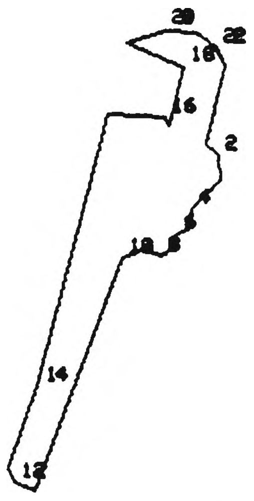
(e)



Model_6

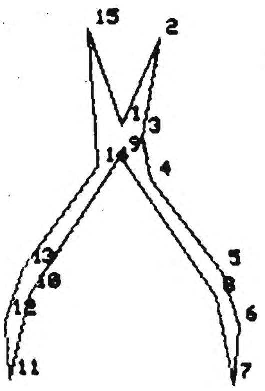
(f)

Fig. 5 (Continued)



Model_7

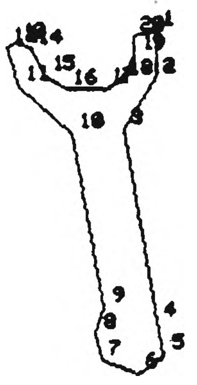
(g)



Model_8

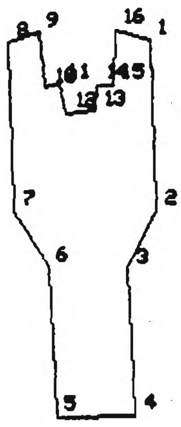
(h)

Fig. 5 (Continued)



Model_9

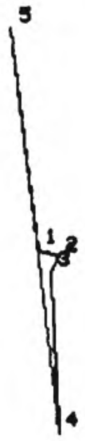
(i)



Model_10

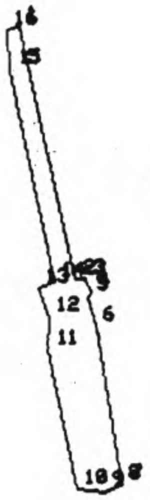
(j)

Fig. 5 (Continued)



Model_11

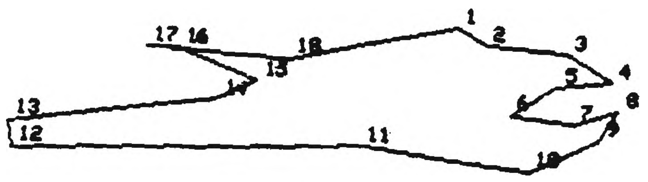
(k)



Model_12

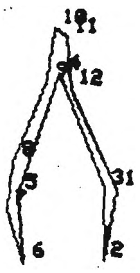
(l)

Fig. 5 (Continued)



Model_13

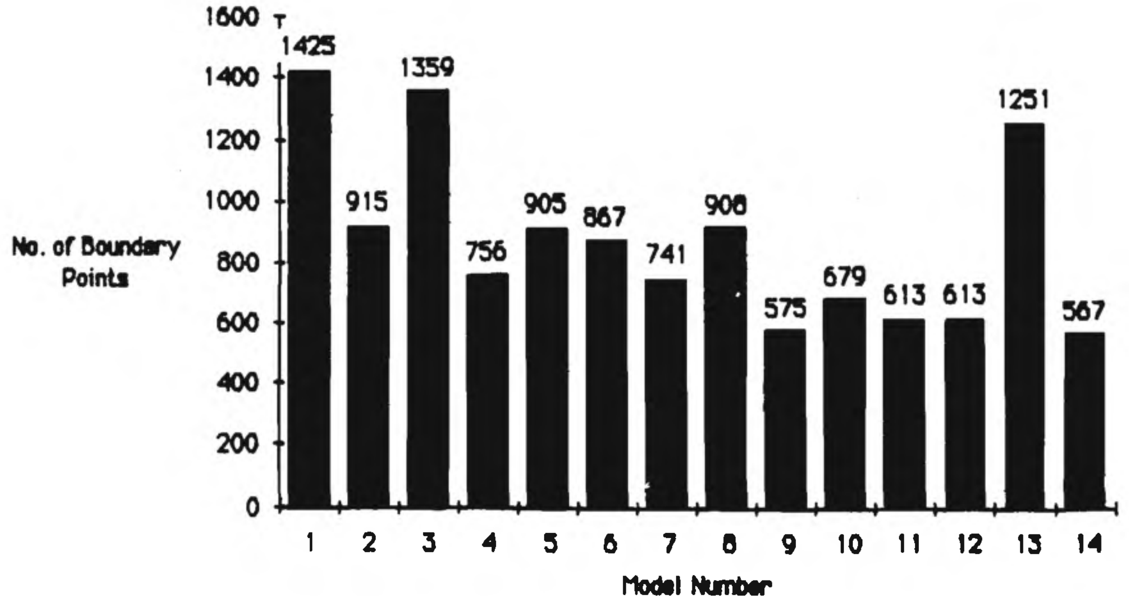
(m)



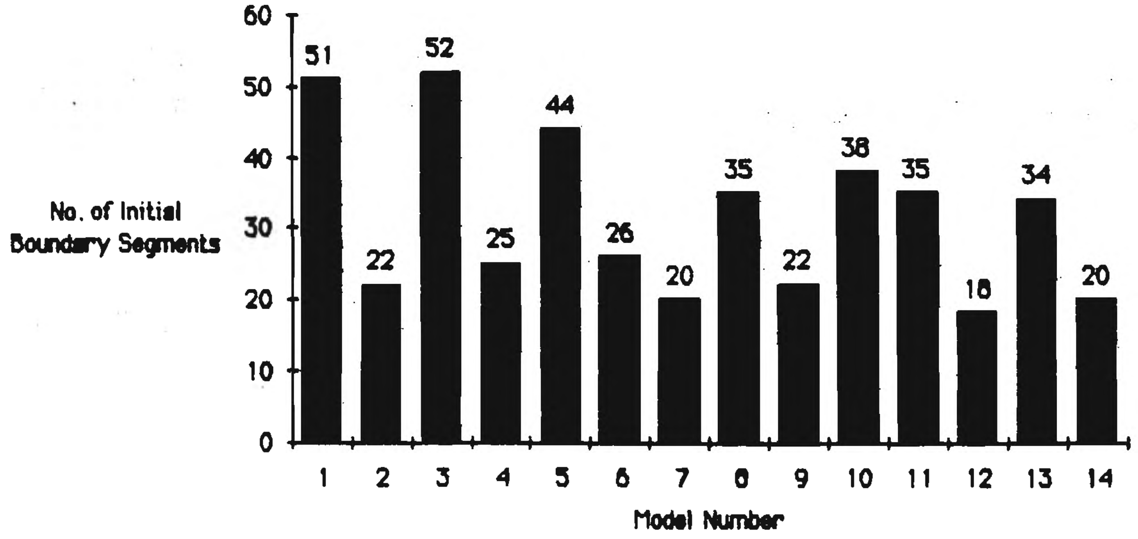
Model_14

(n)

Fig. 5 (Continued)

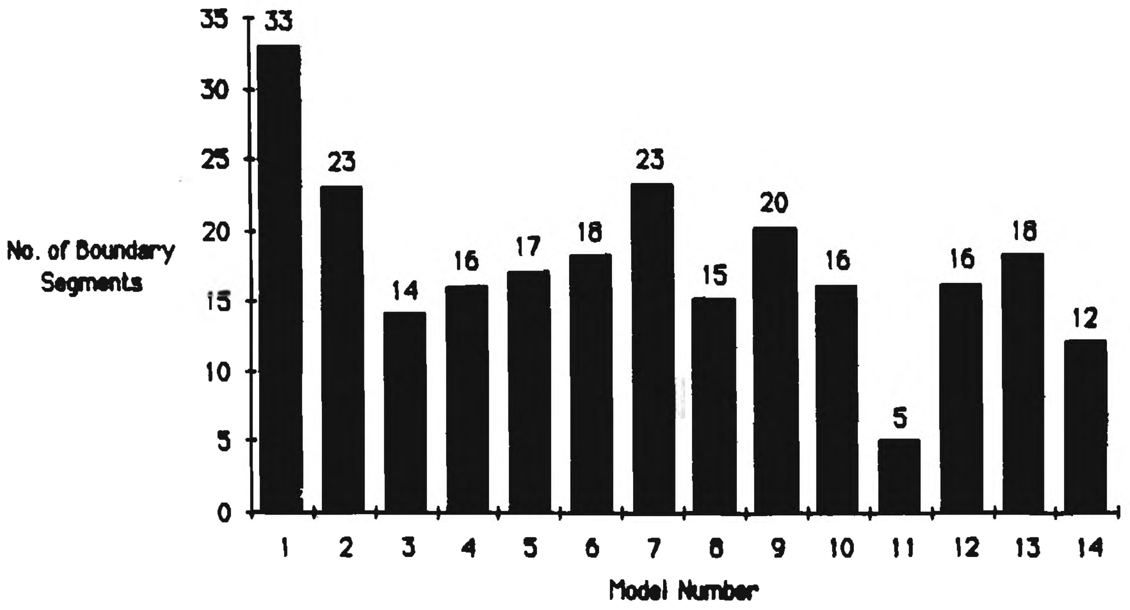


(a) Model boundaries

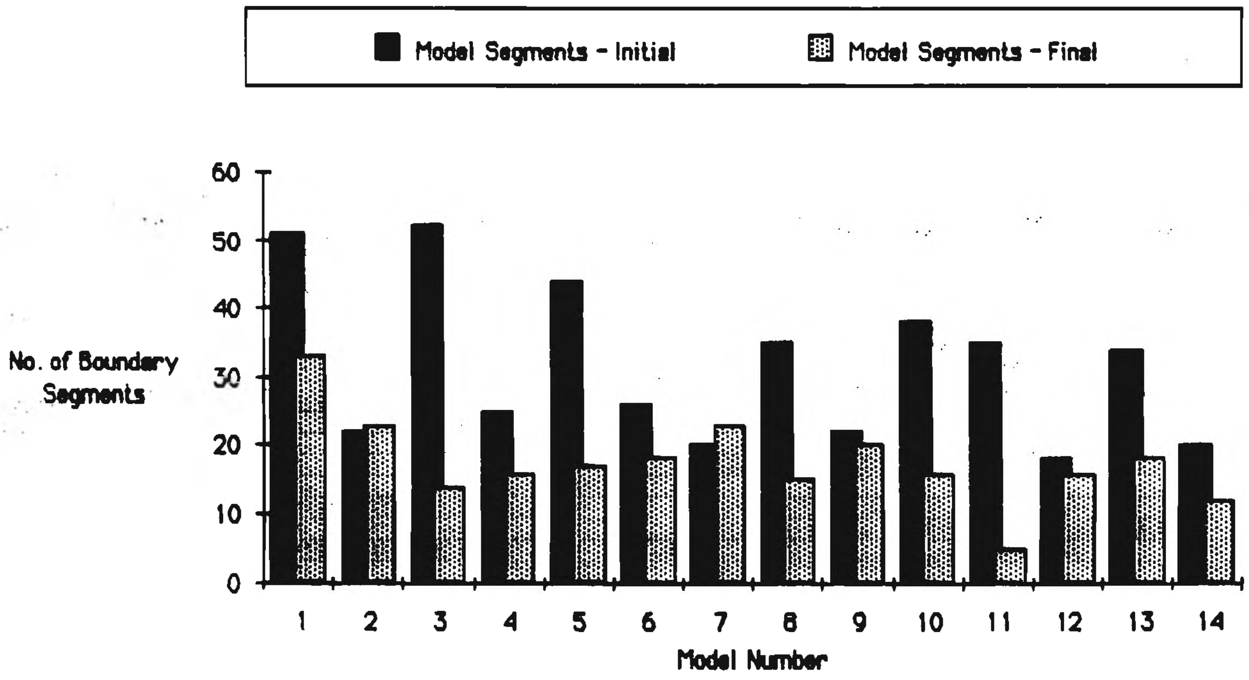


(b) Model segments - initial

Fig. 6 Polygonal approximation performance charts for the models (a to j).

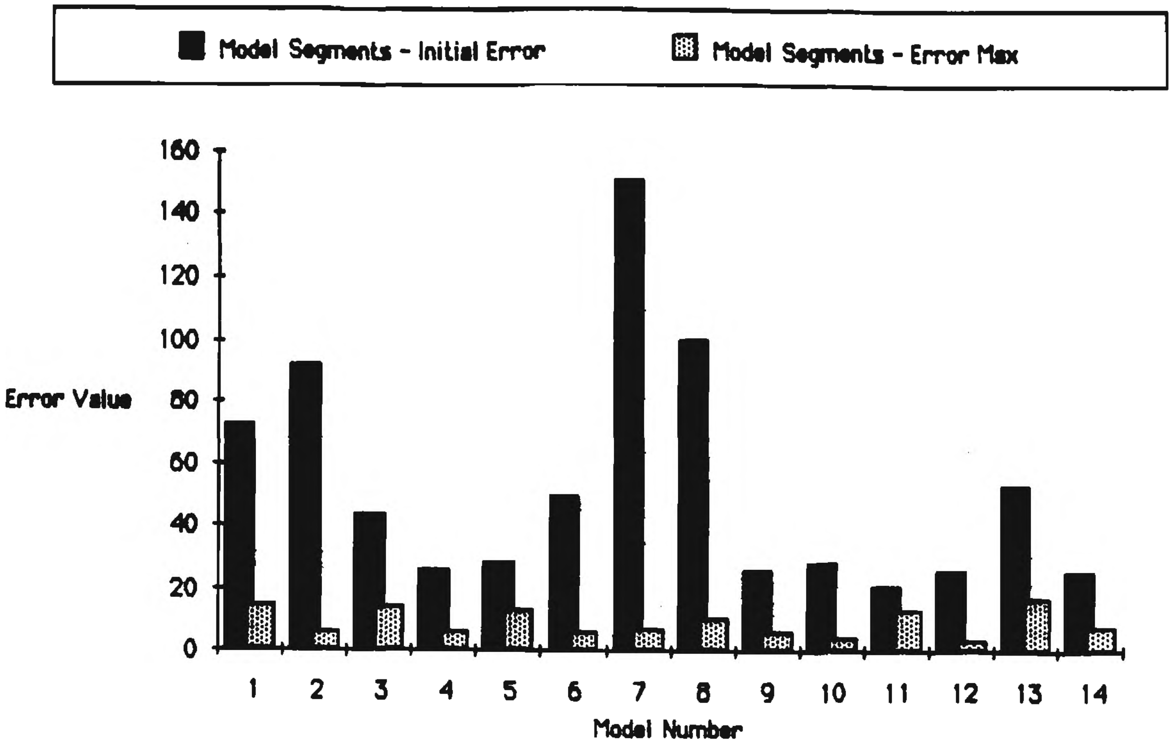


(c) Model segments - final

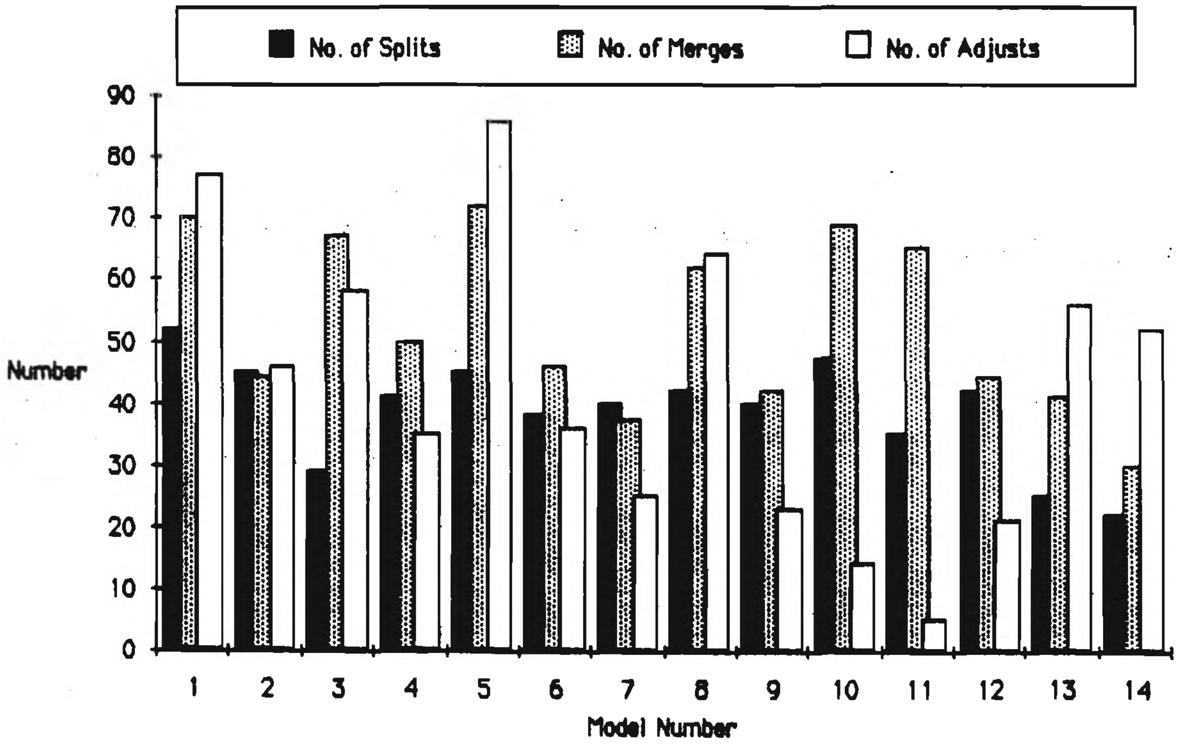


(d) Model segments - initial vs final

Fig. 6 (Continued)

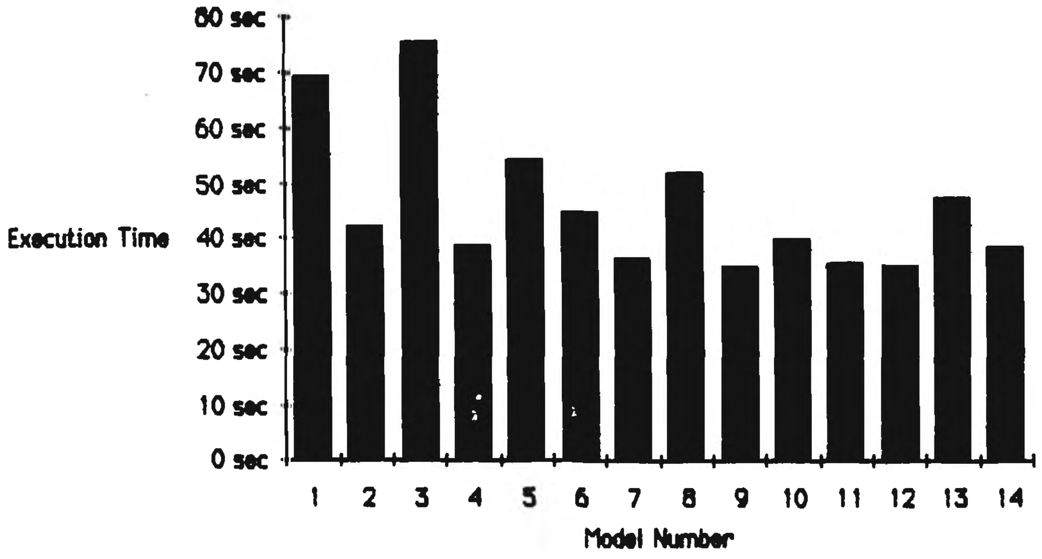


(e) Model segments - initial vs error maximum

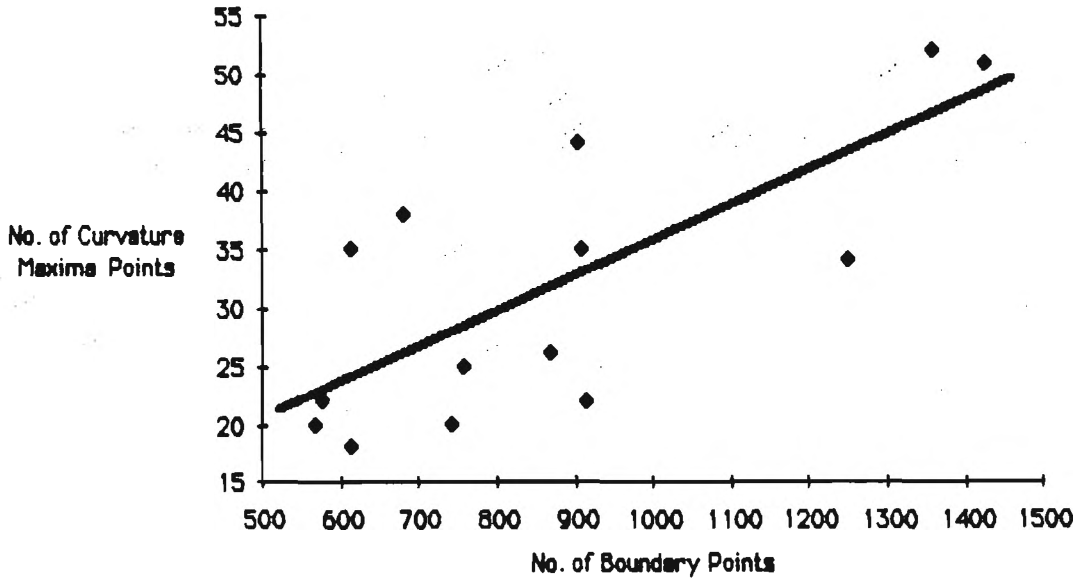


(f) Models - No. of splits, merges, and adjusts

Fig. 6 (Continued)

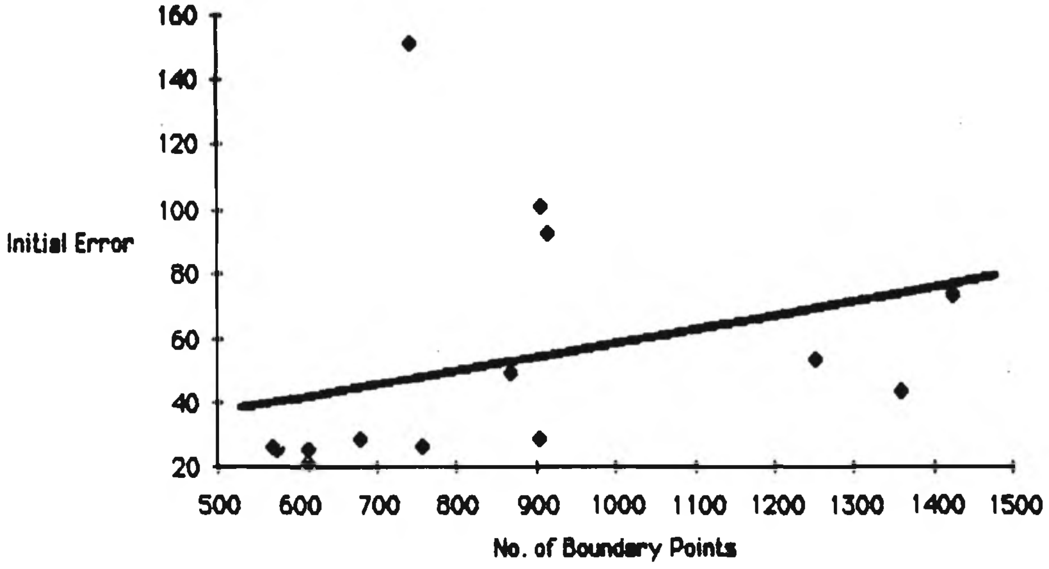


(g) Models - execution time

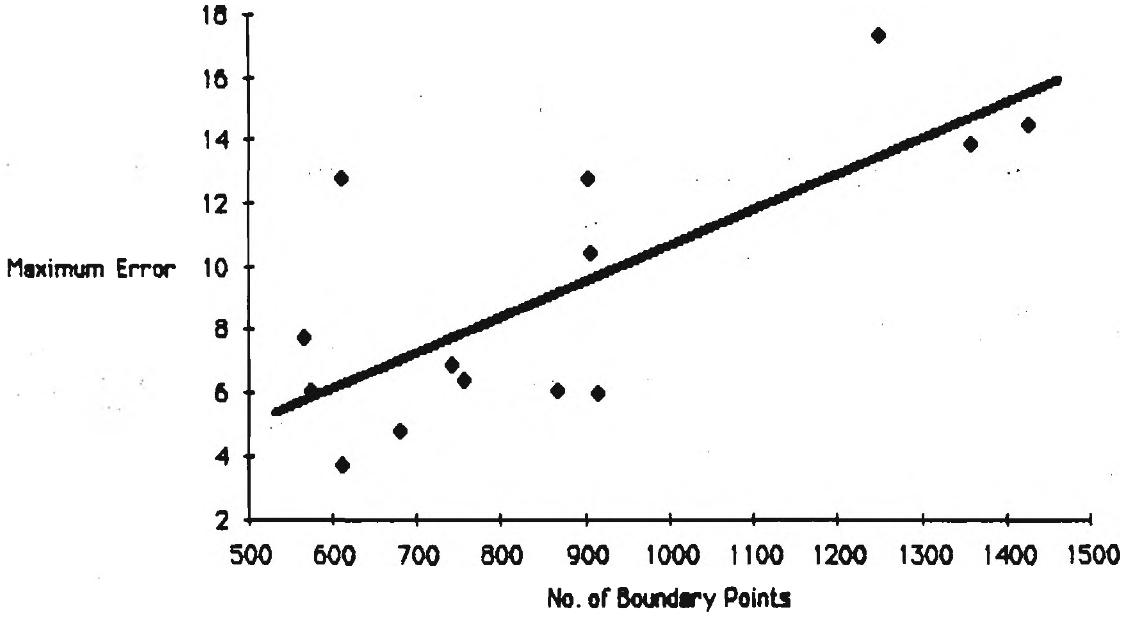


(h) Models - curvature data

Fig. 6 (Continued)

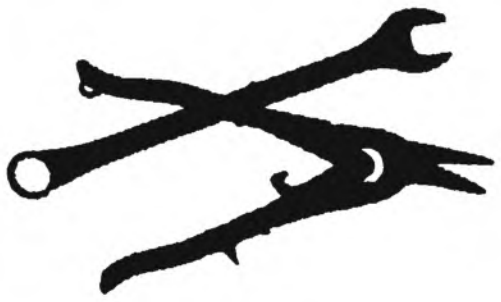


(i) Models - initial error

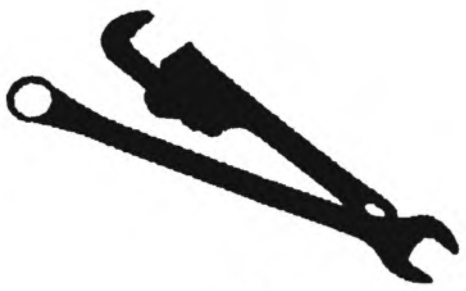


(j) Models - maximum error

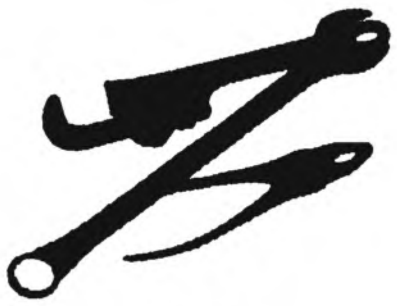
Fig. 6 (Continued)



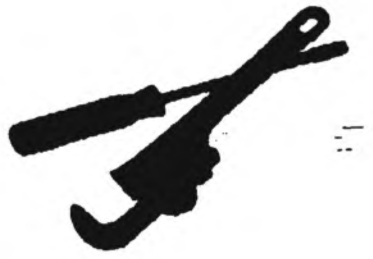
(a)



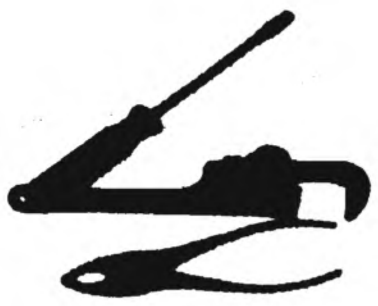
(b)



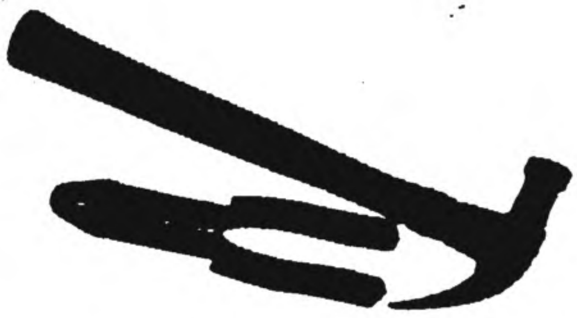
(c)



(d)

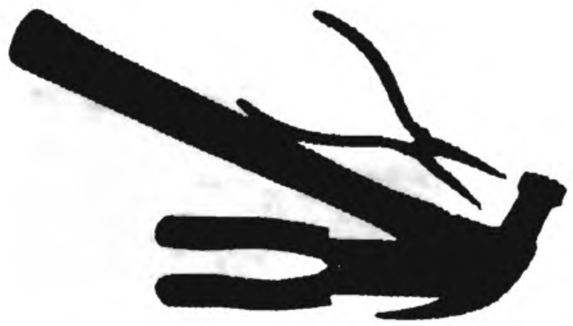


(e)

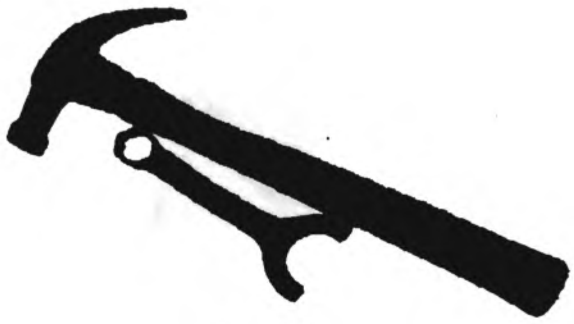


(f)

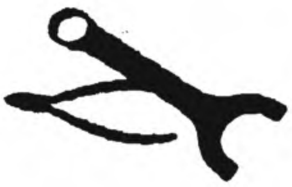
Fig. 7 Images of the occluded objects (a to t).



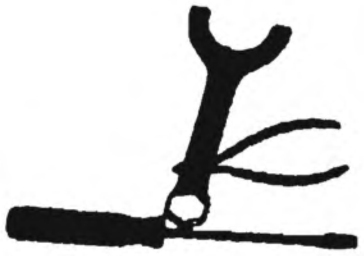
(g)



(h)



(i)



(j)

(k)

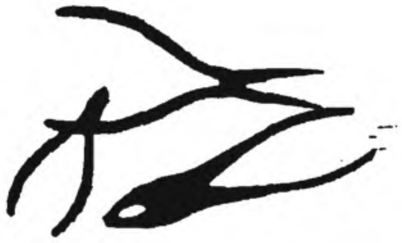
(l)

Fig. 7 (Continued)

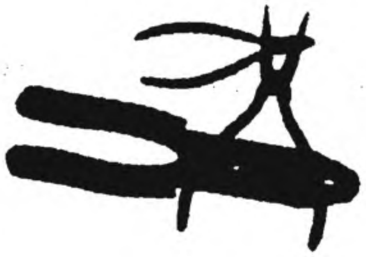
(m)

(n)

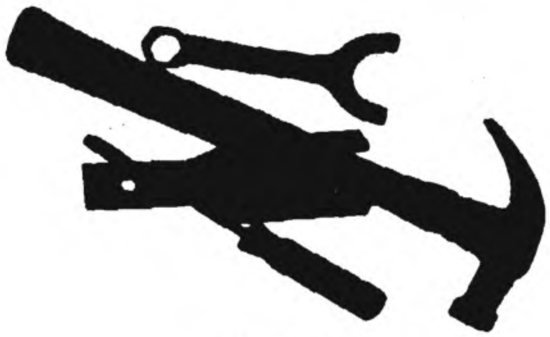
(o)



(p)

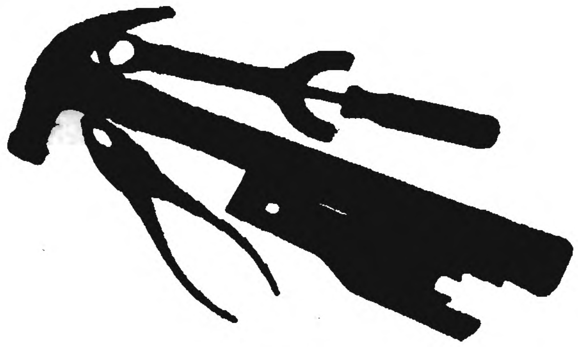


(q)

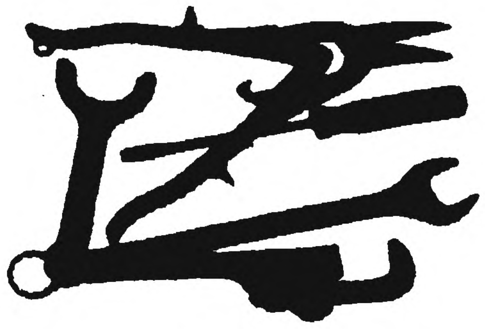


(r)

Fig. 7 (Continued)

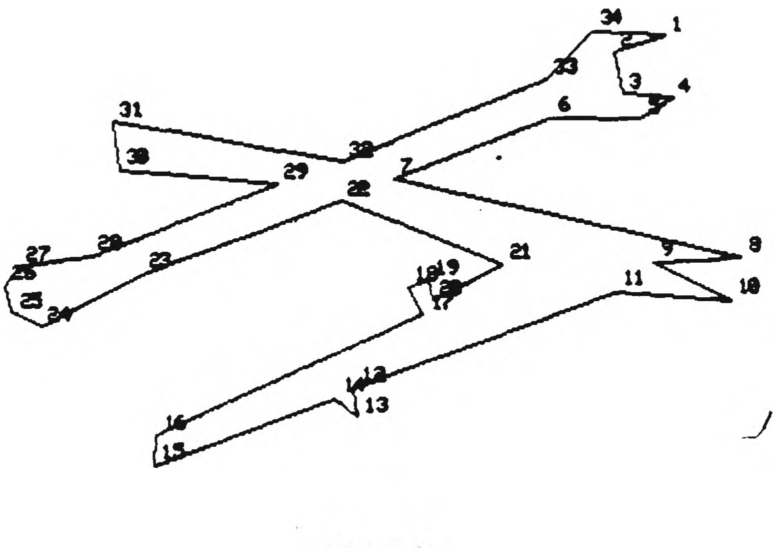


(s)

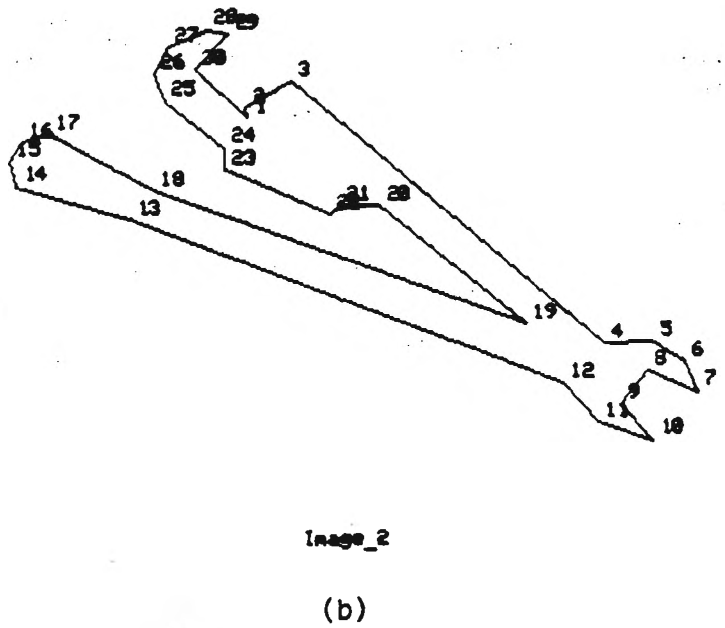


(t)

Fig. 7 (Continued)



(a)



(b)

Fig. 8 Polygonal approximation of the occluded objects (a to t).

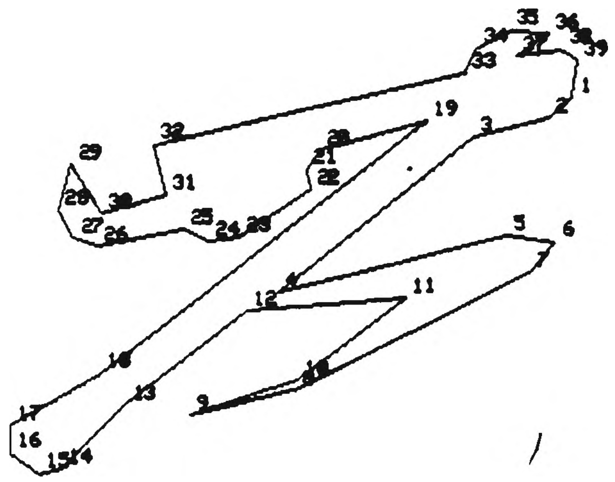


Image_3

(c)

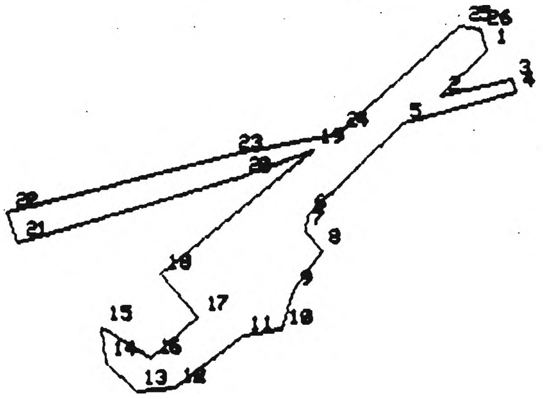


Image_4

(d)

Fig. 8 (Continued)

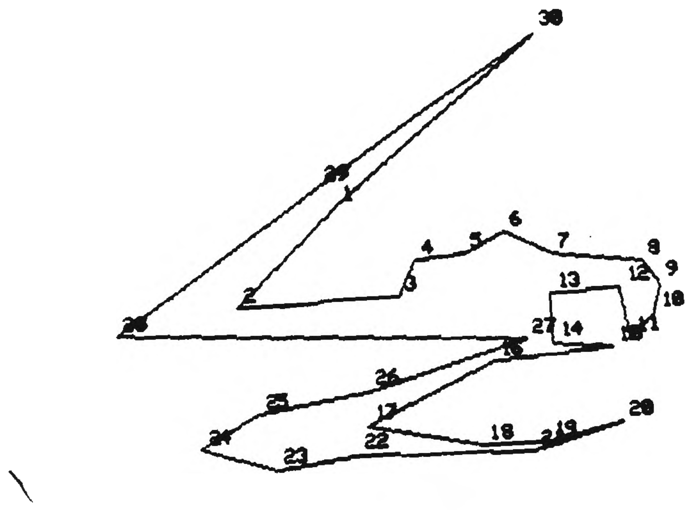


Image_5

(e)

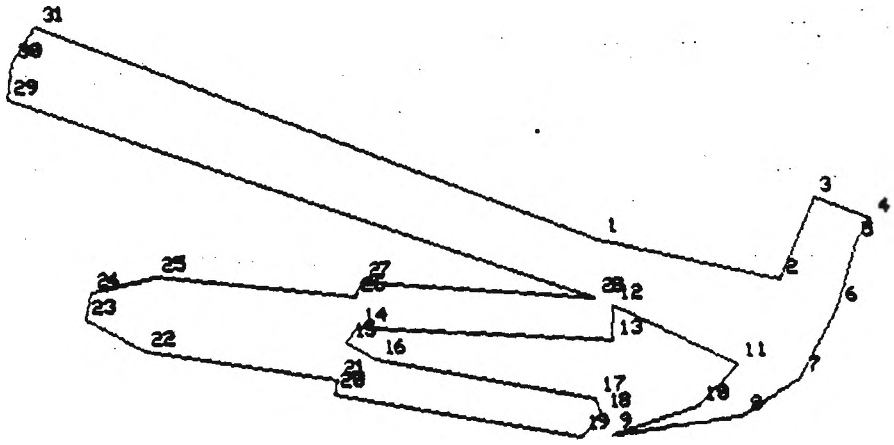
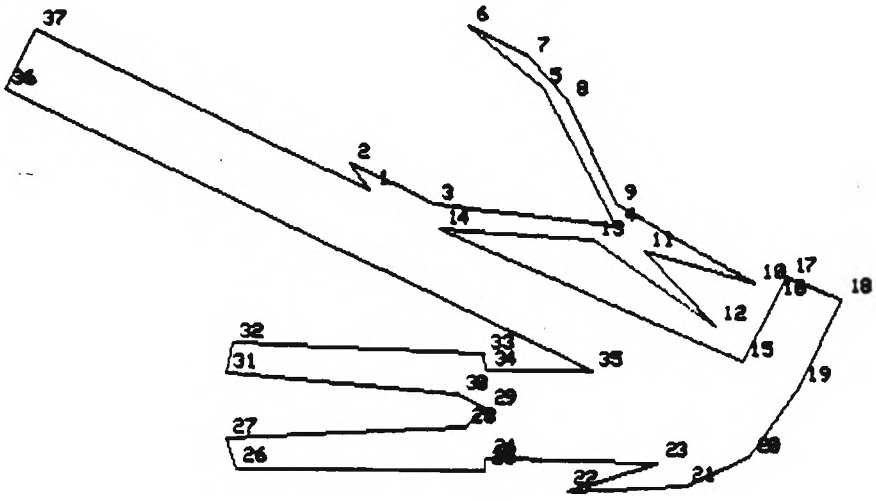


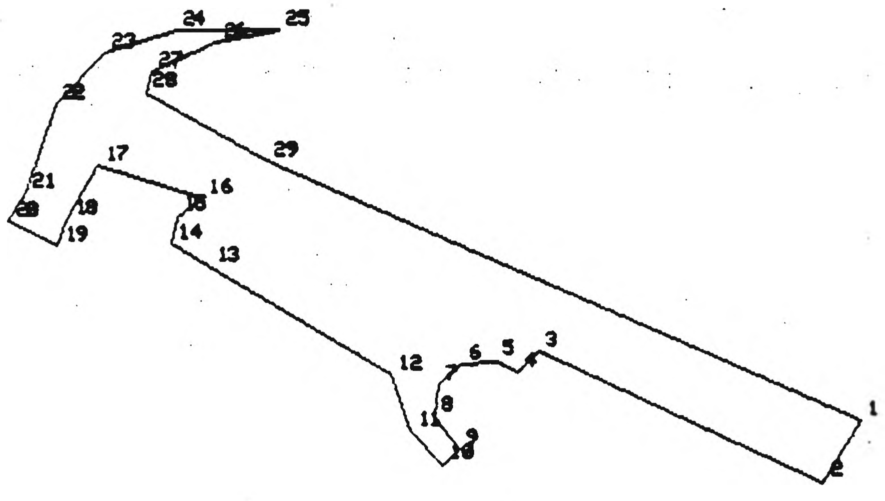
Image_6

(f)

Fig. 8 (Continued)



(g)



(h)

Fig. 8 (Continued)

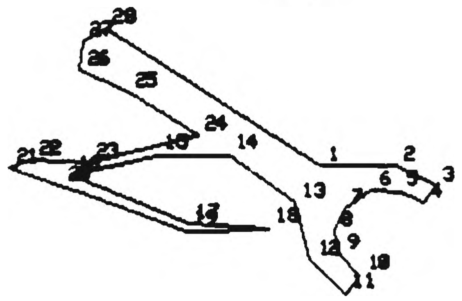


Image_9

(i)

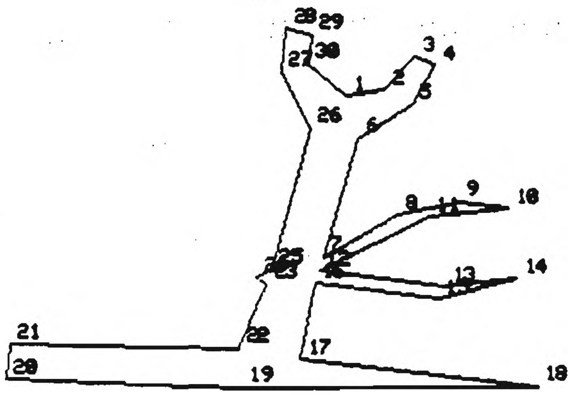
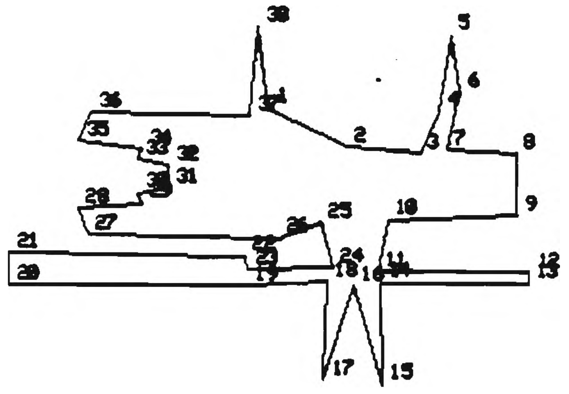


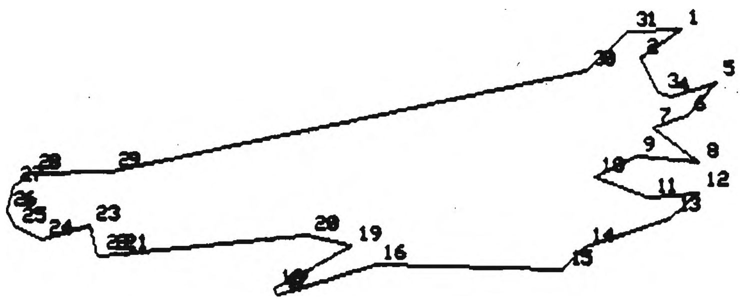
Image_10

(j)

Fig. 8 (Continued)

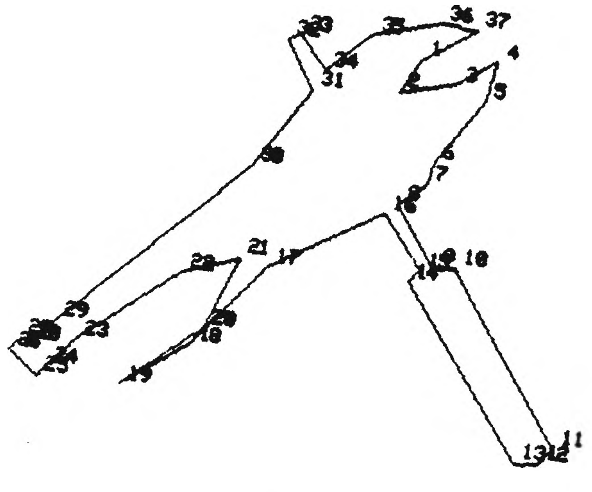


(k)

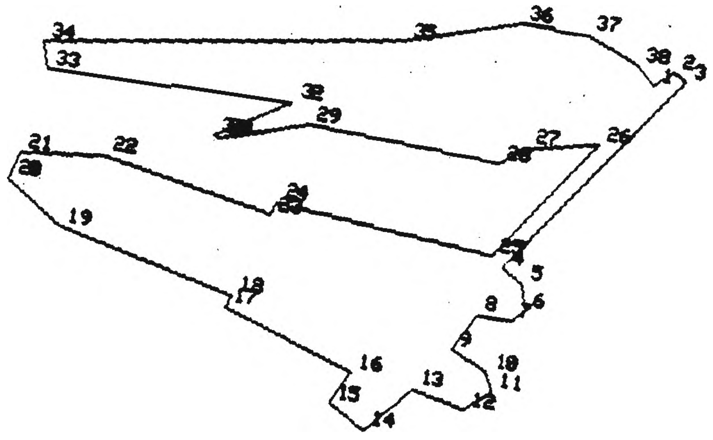


(l)

Fig. 8 (Continued)



(m)



(n)

Fig. 8 (Continued)

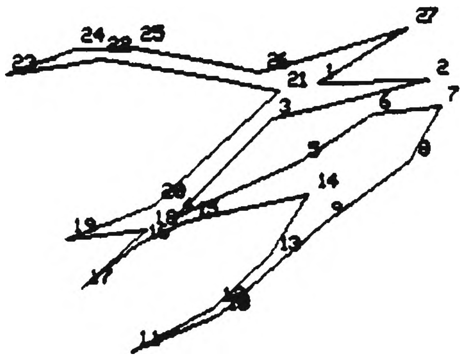


Image_15

(o)

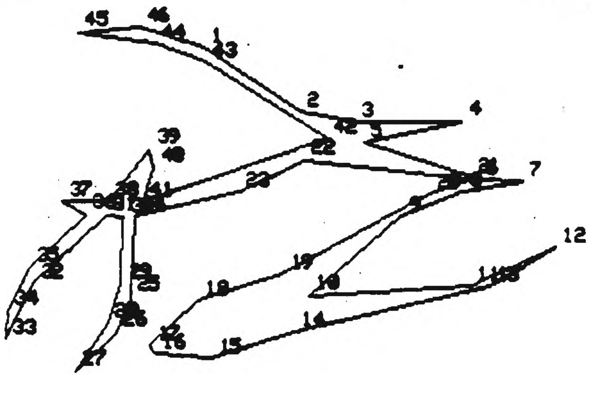
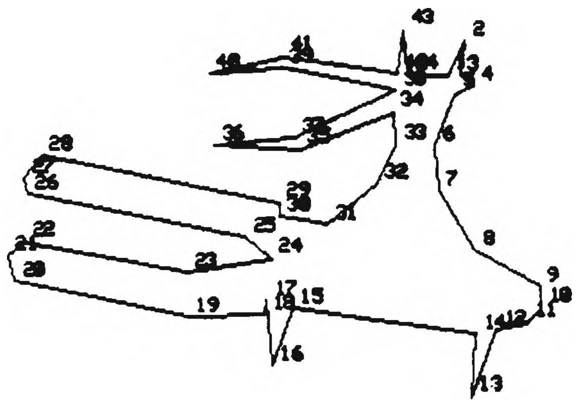


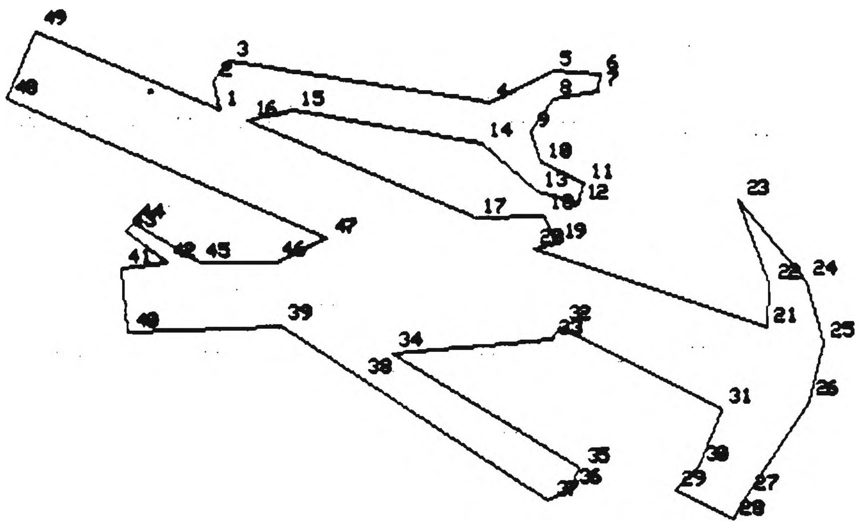
Image_16

(p)

Fig. 8 (Continued)



(q)



(r)

Fig. 8 (Continued)

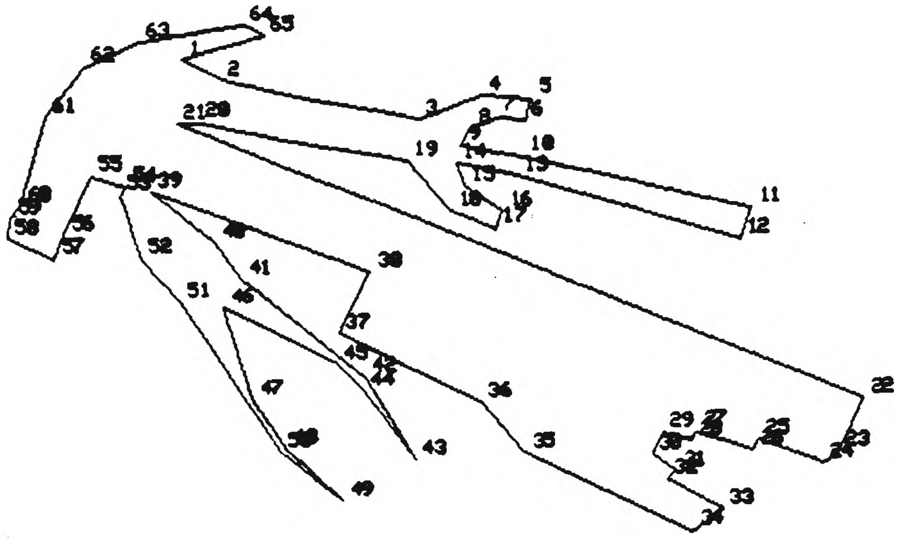


Image_19

(s)

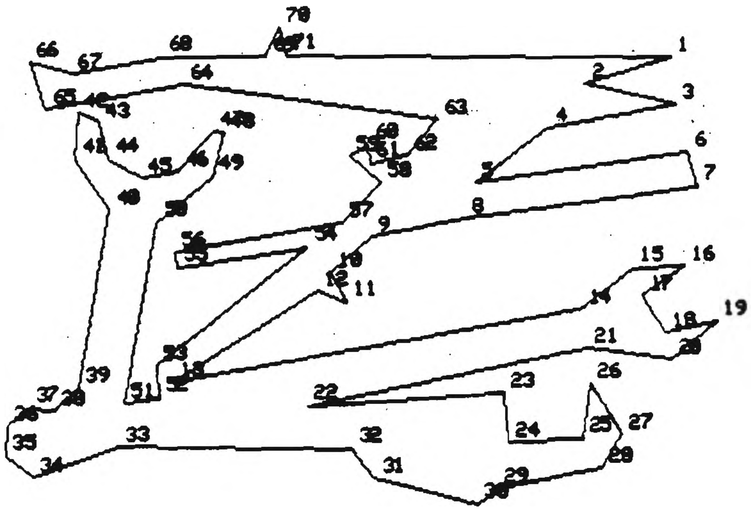
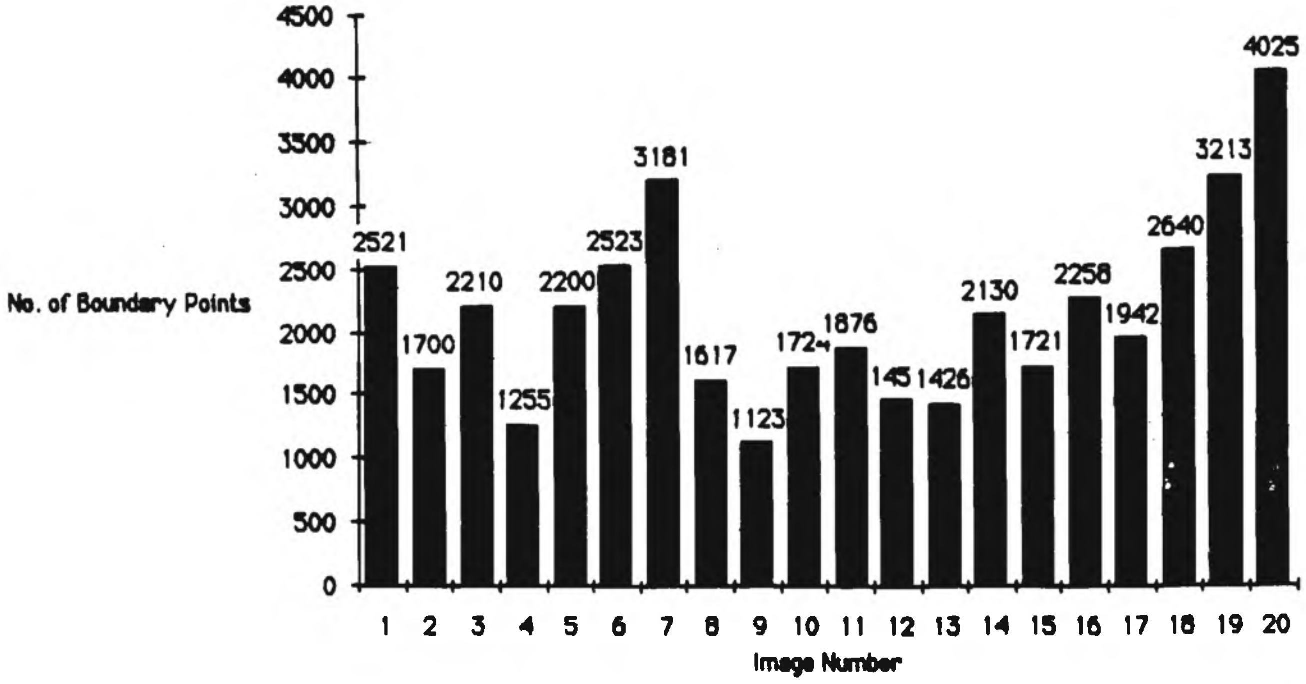


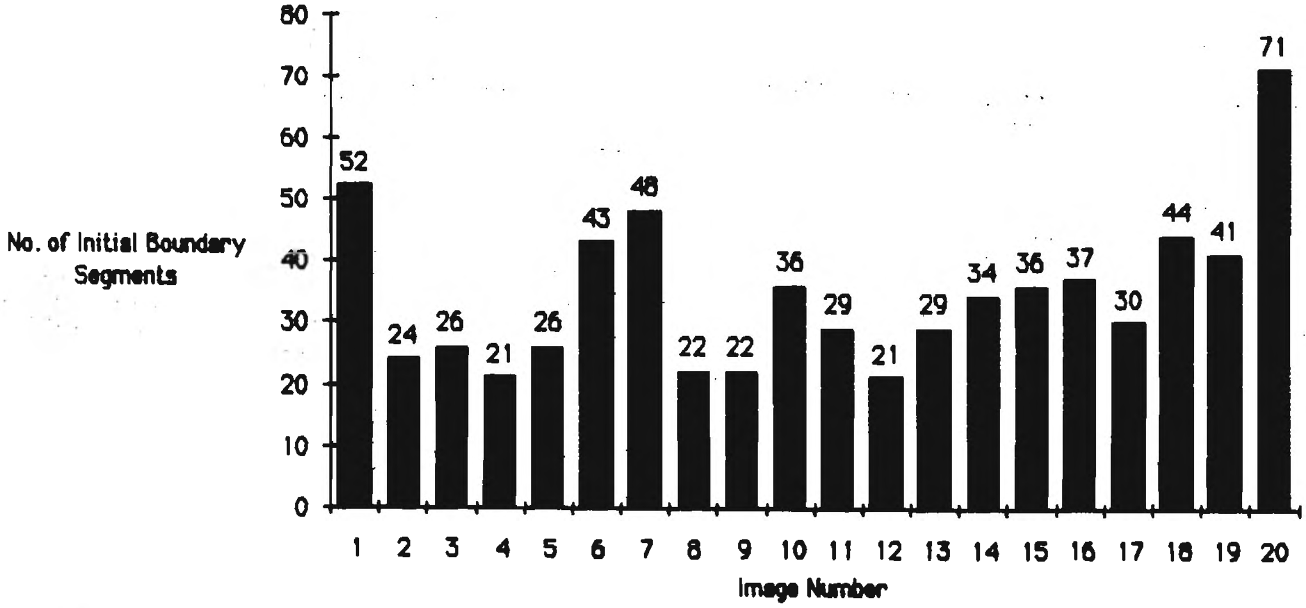
Image_20

(t)

Fig. 8 (Continued)

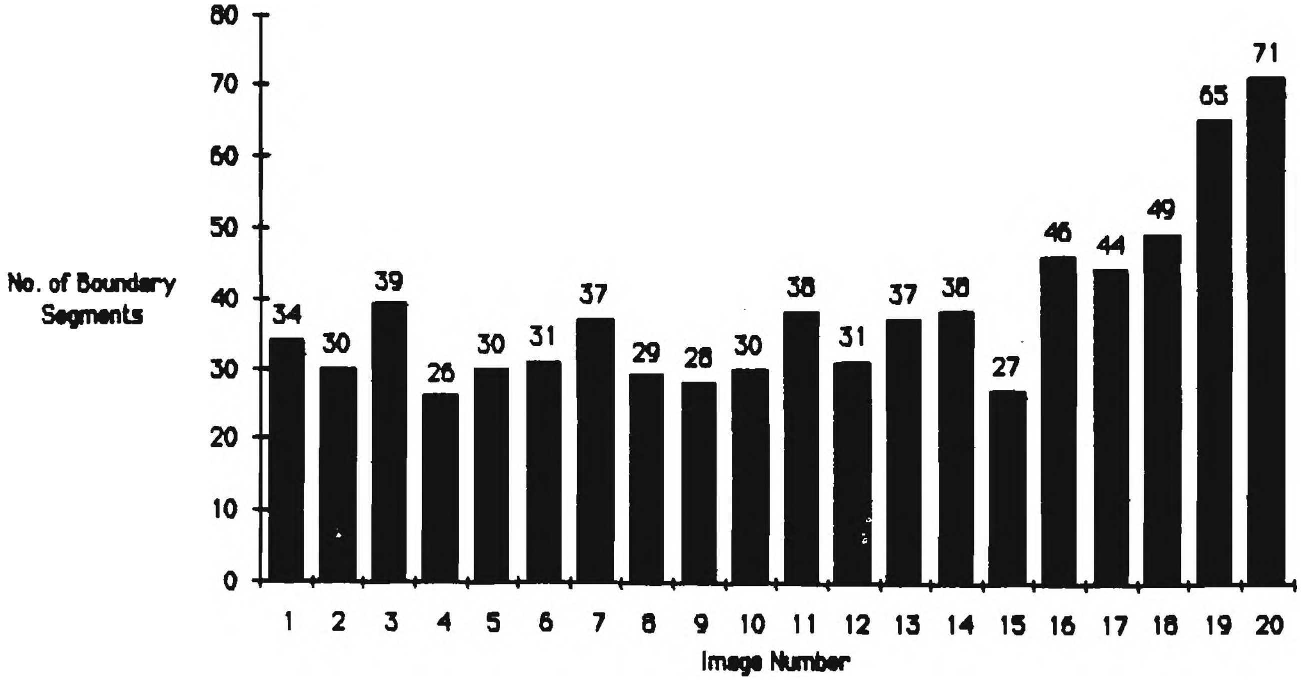


(a) Image boundaries

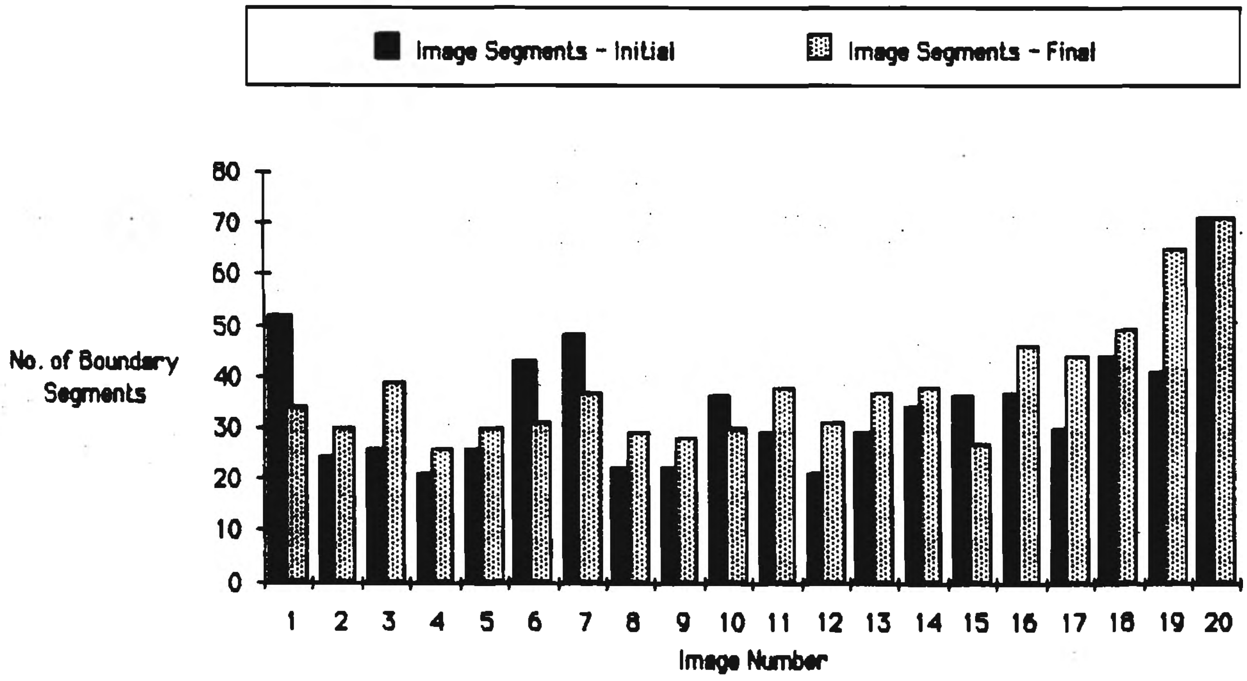


(b) Image segments - initial

Fig. 9 Polygonal approximation performance charts for the occluded objects (a to j).

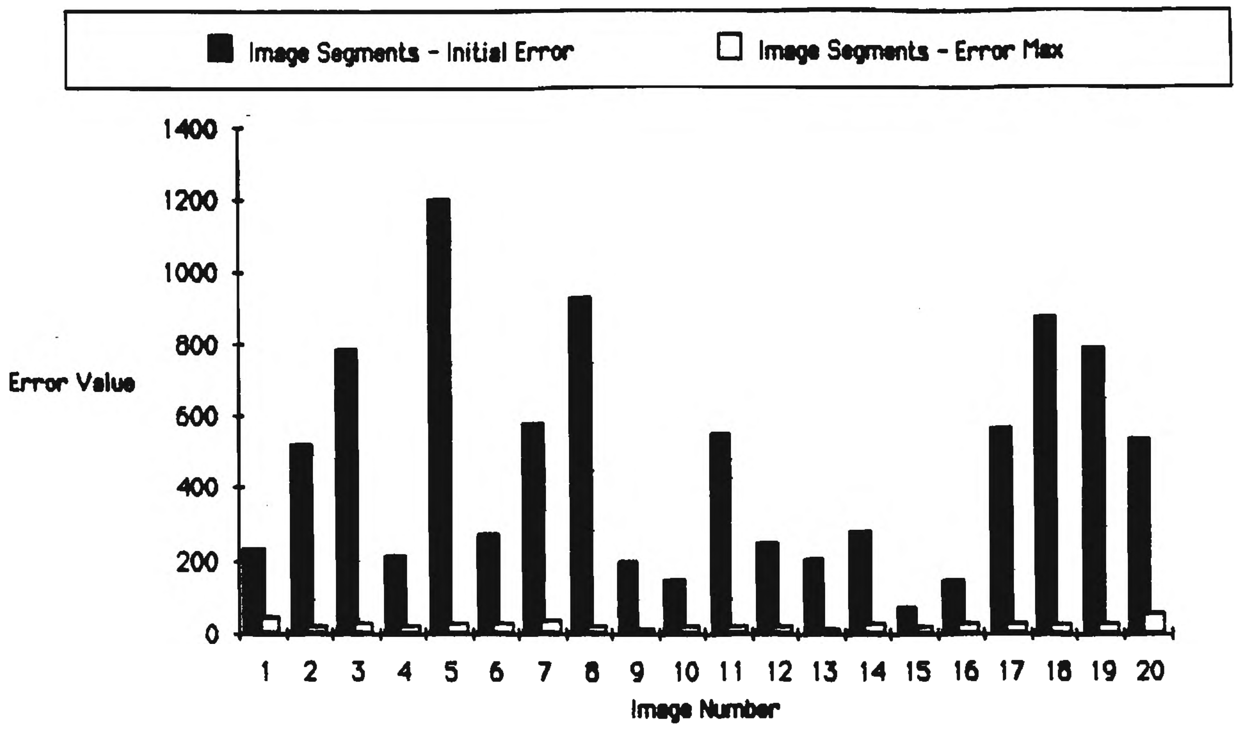


(c) Image segments - final

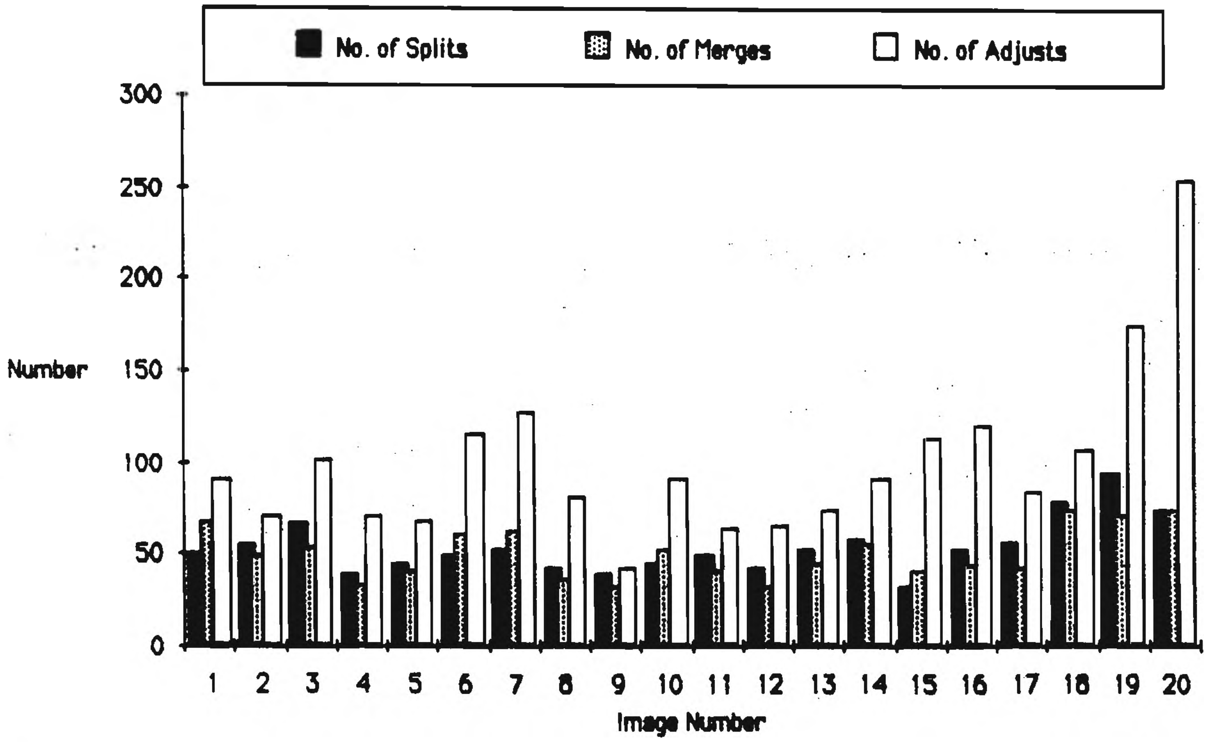


(d) Image segments - initial vs final

Fig. 9 (Continued)

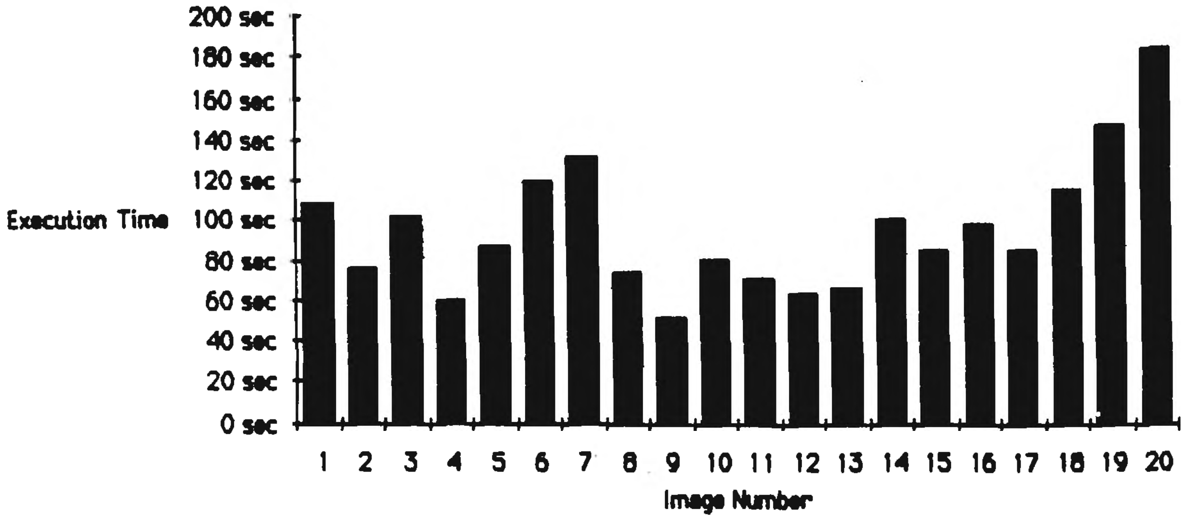


(e) Image segments - initial vs error maximum

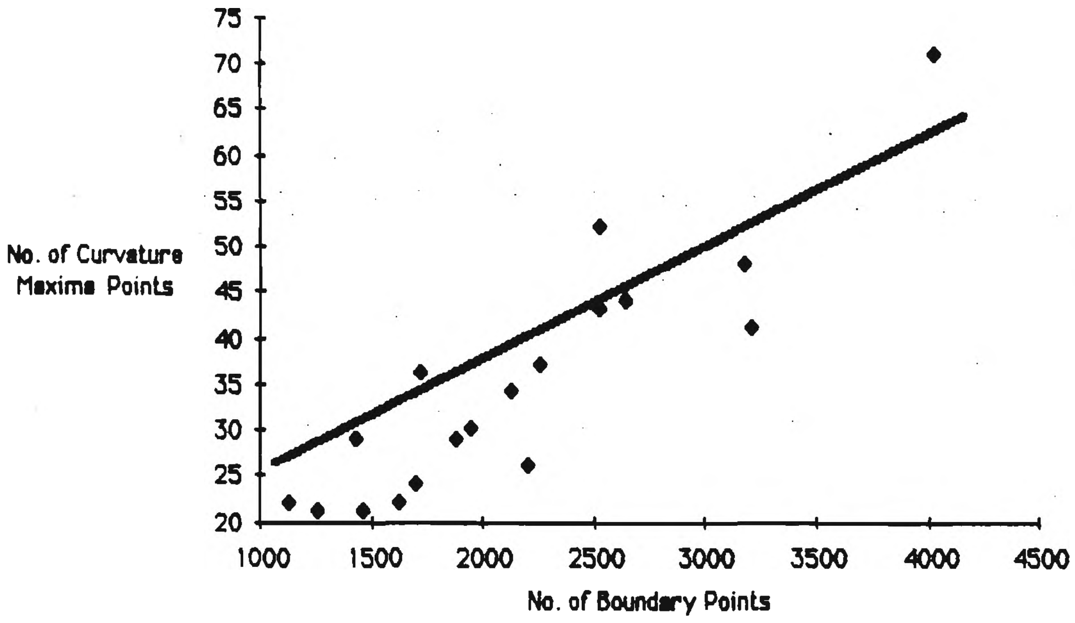


(f) Images - No. of splits, merges, and adjusts

Fig. 9 (Continued)

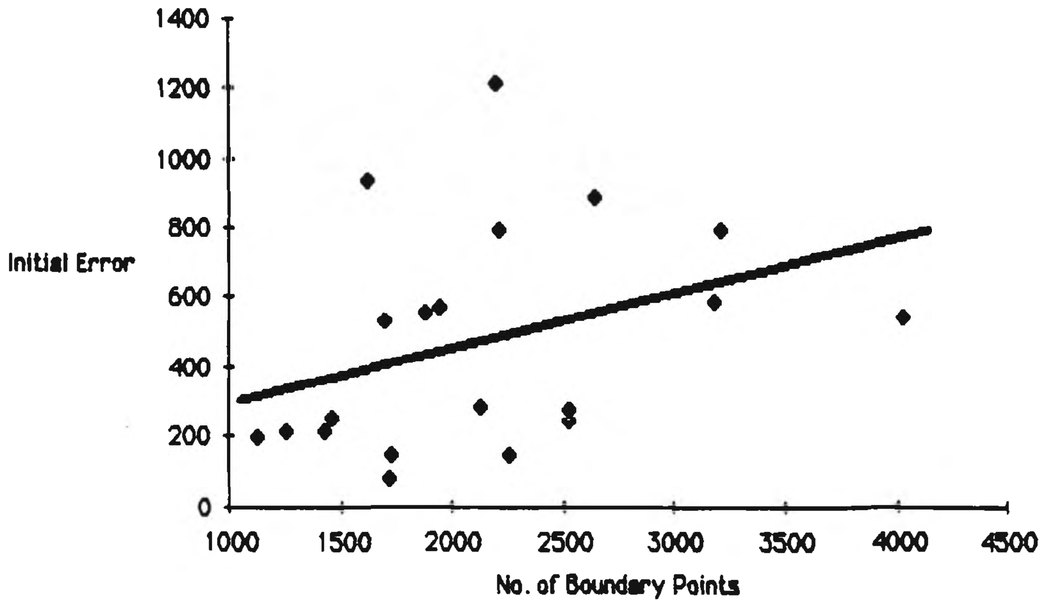


(g) Images - execution time

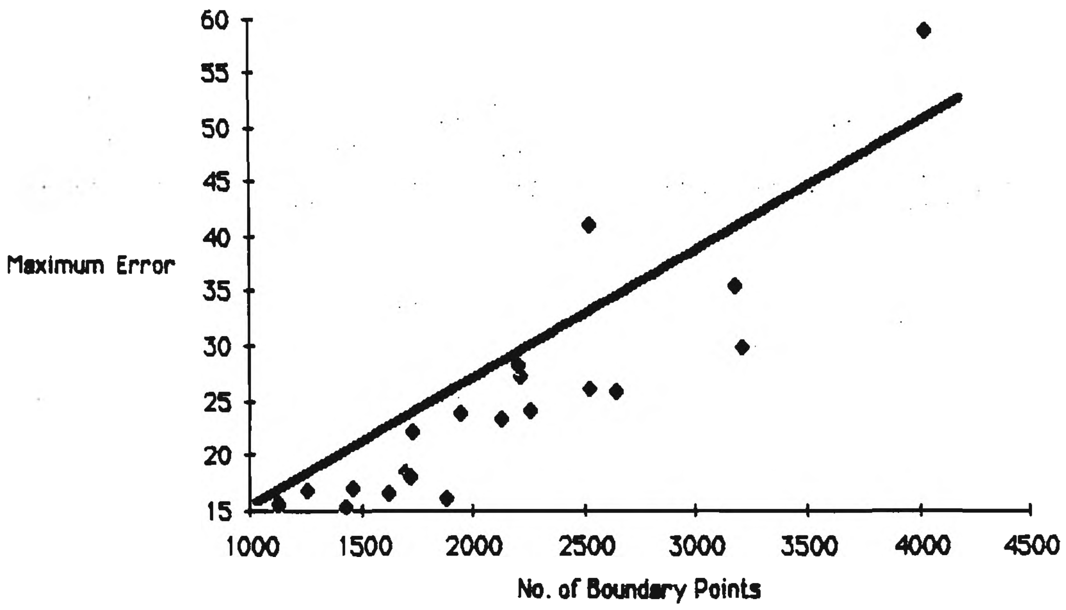


(h) Images - curvature data

Fig. 9 (Continued)



(i) Images - initial error



(j) Images - maximum error

Fig. 9 (Continued)

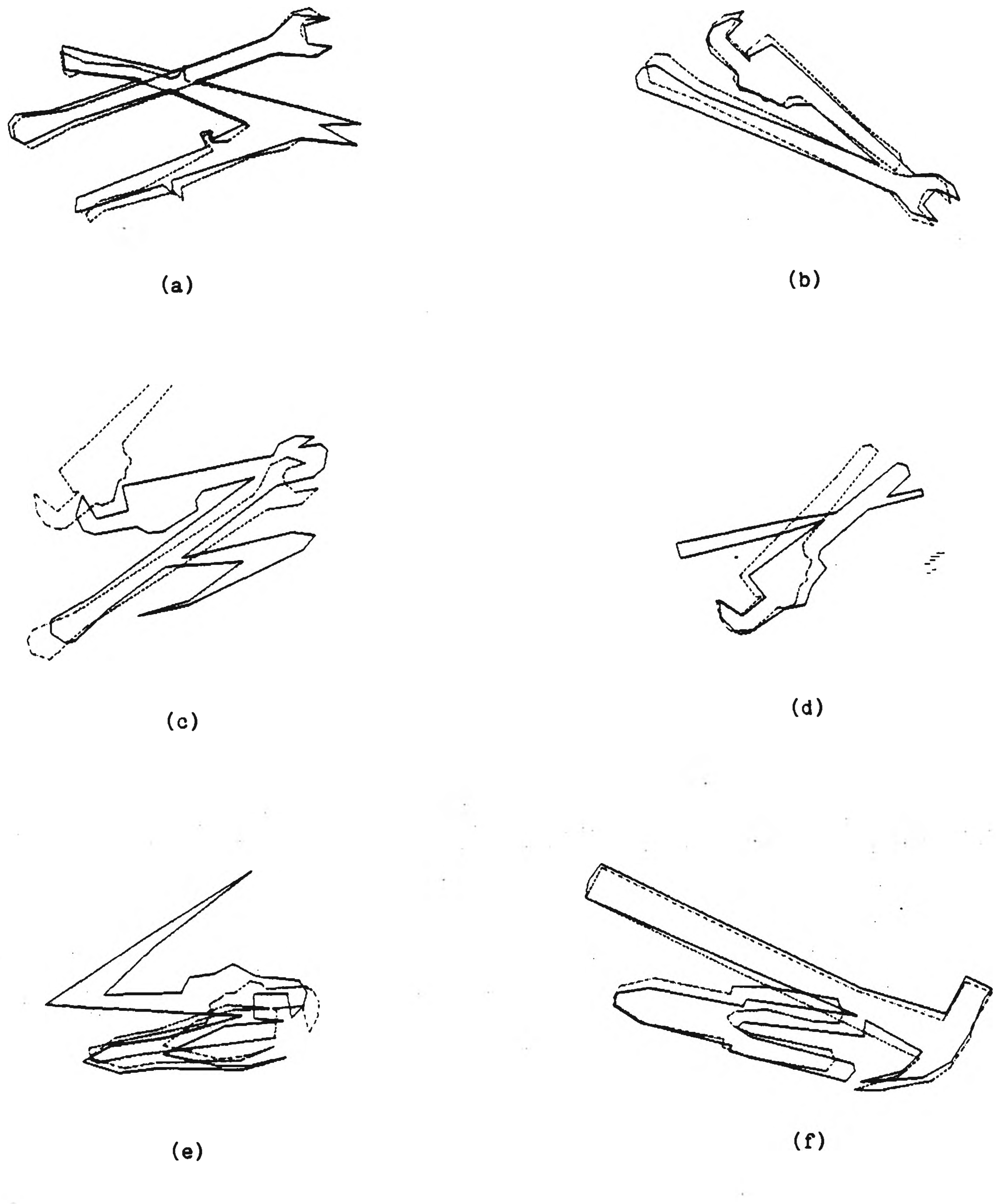
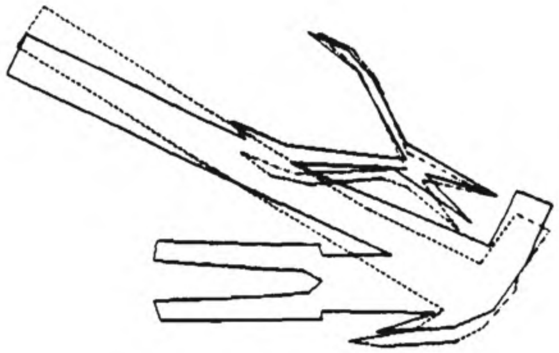
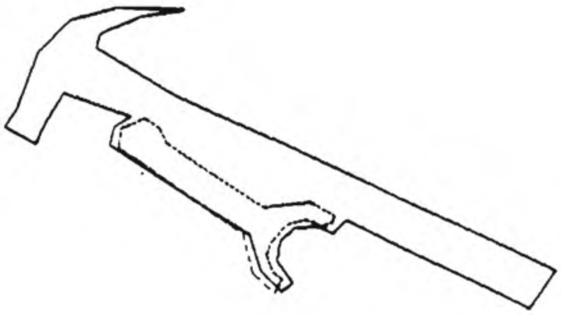
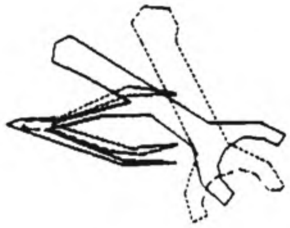
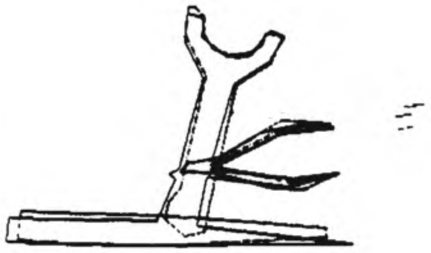


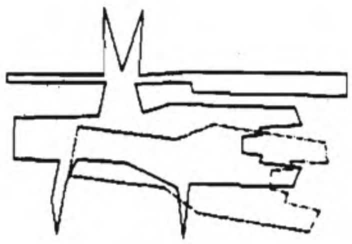
Fig. 10 Results of matching for the occluded images shown in Fig. 7 (a to t). Solid red lines show the polygonal approximation of the images using a split-merge algorithm. Dotted lines (in blue and green colors) show the model (polygonal approximation), when rotational and translational transforms computed by the algorithm are applied to it. Green color of the dotted lines shows the actual segments which matched.


(g)


(h)


(i)


(j)


(k)

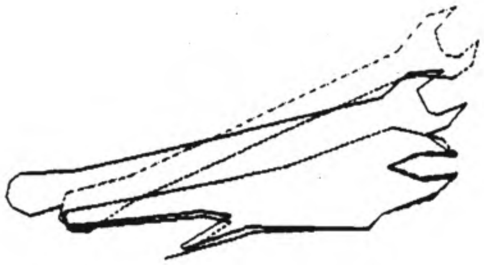
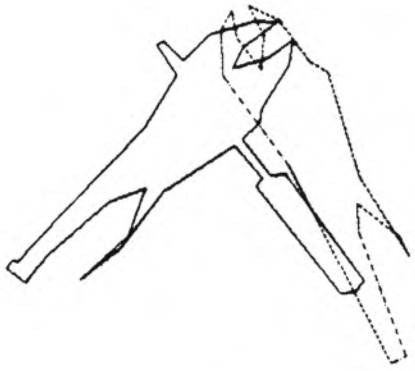
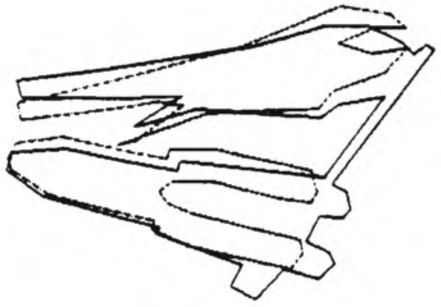

(l)

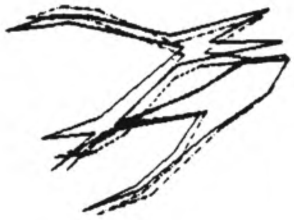
Fig. 10 (Continued)



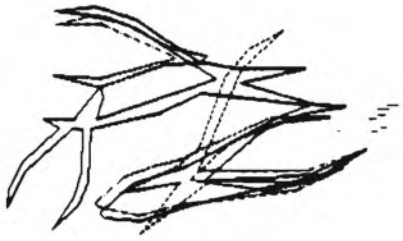
(m)



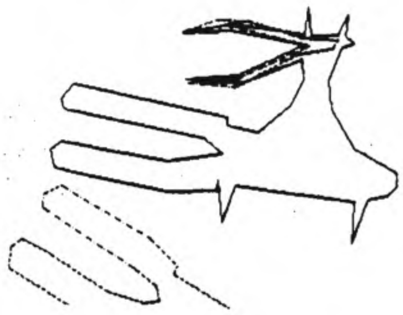
(n)



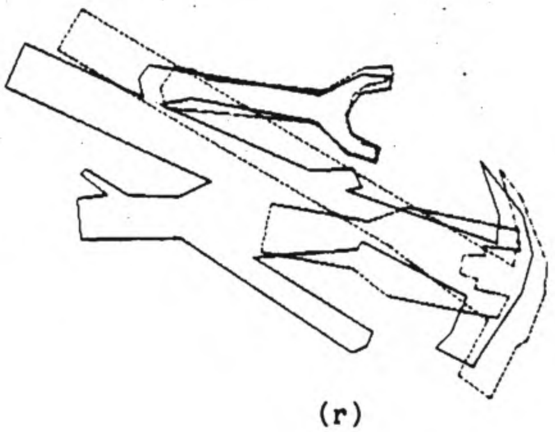
(o)



(p)

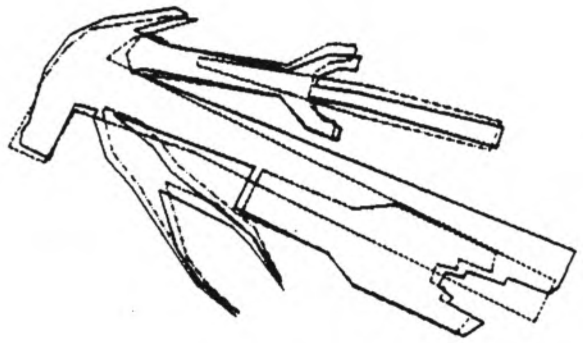


(q)

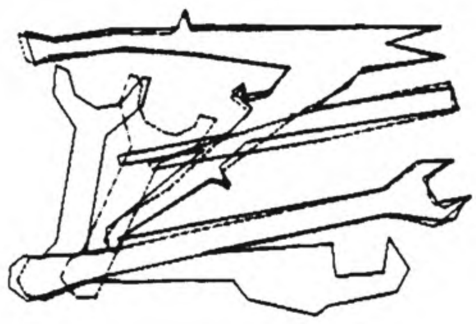


(r)

Fig. 10 (Continued)

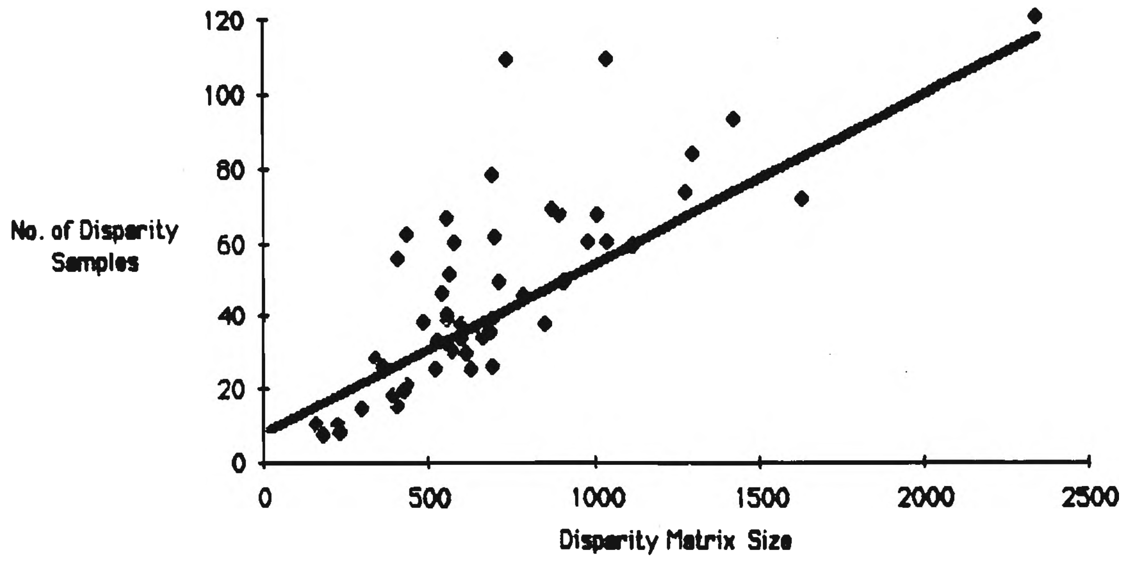


(s)

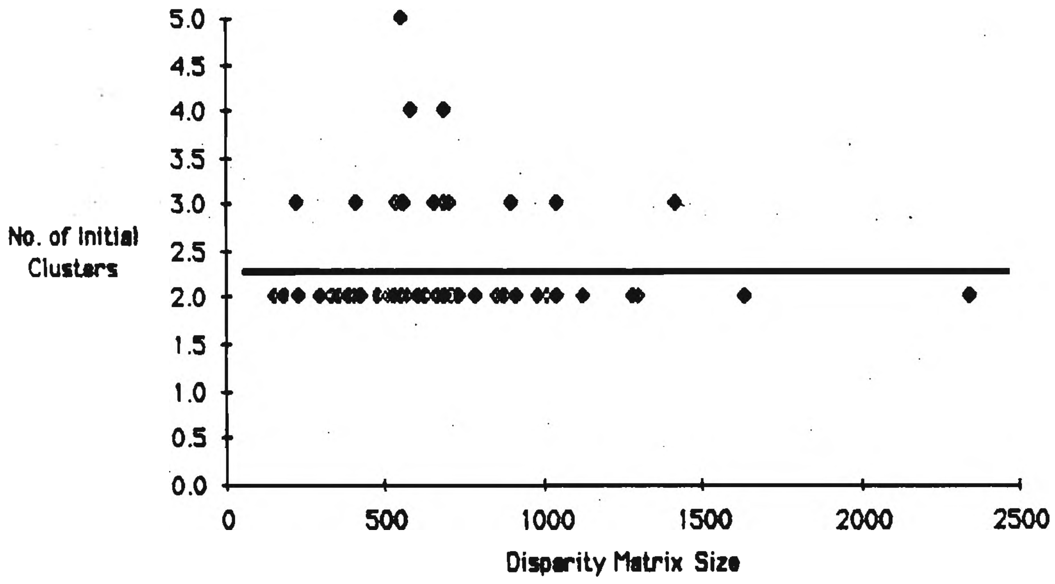


(t)

Fig. 10 (Continued)

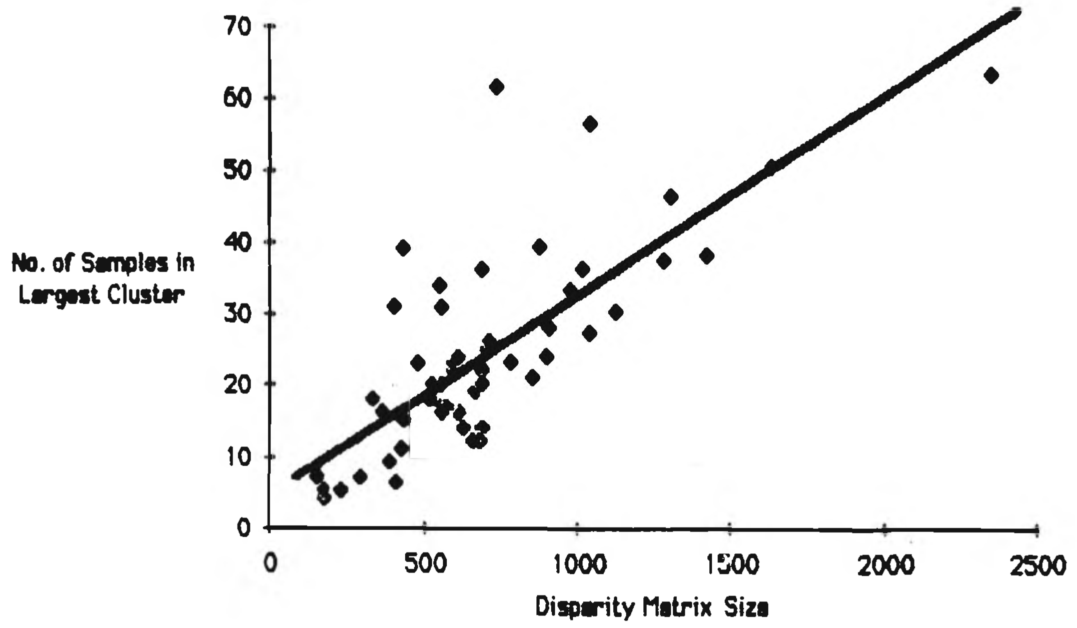


(a) Disparity matrix sparsity

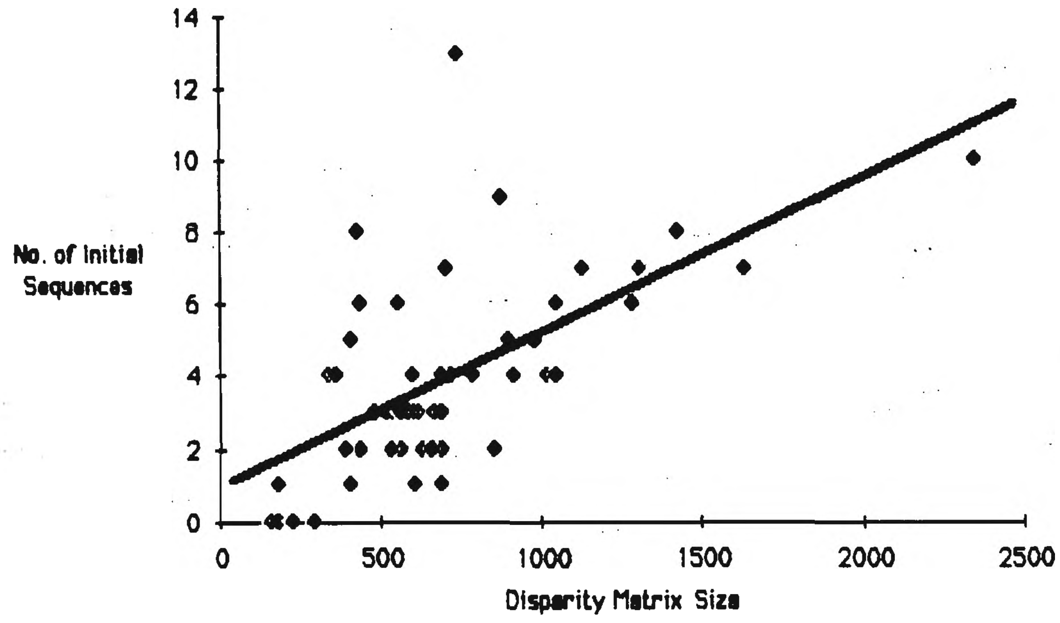


(b) Disparity matrix vs. number of initial clusters

Fig. 11 Performance charts for the recognition algorithm (a to g).

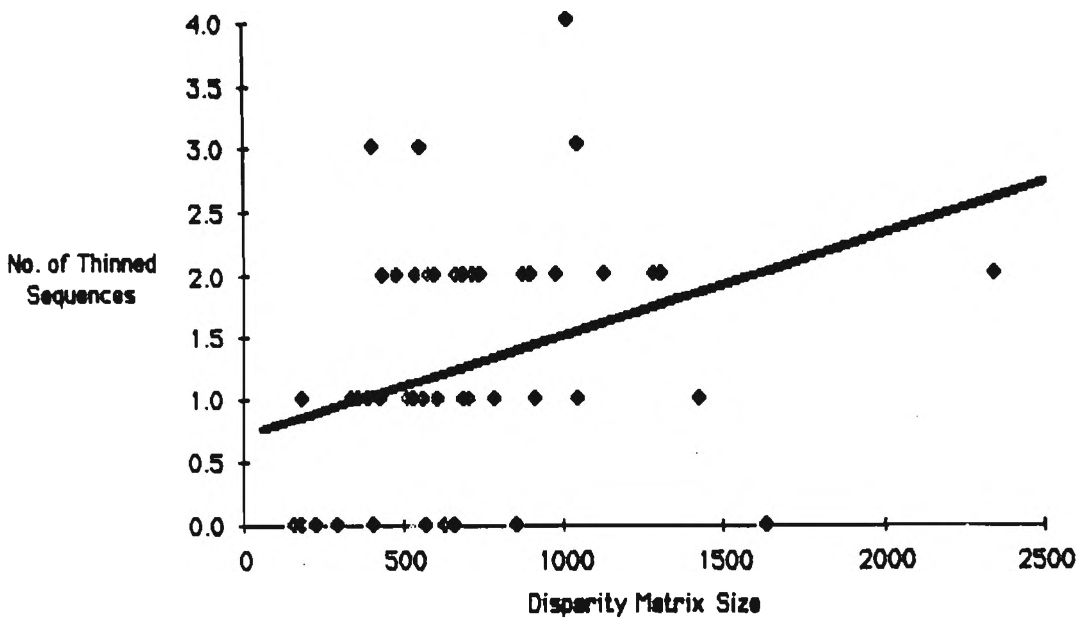


(c) Disparity matrix vs. cluster size

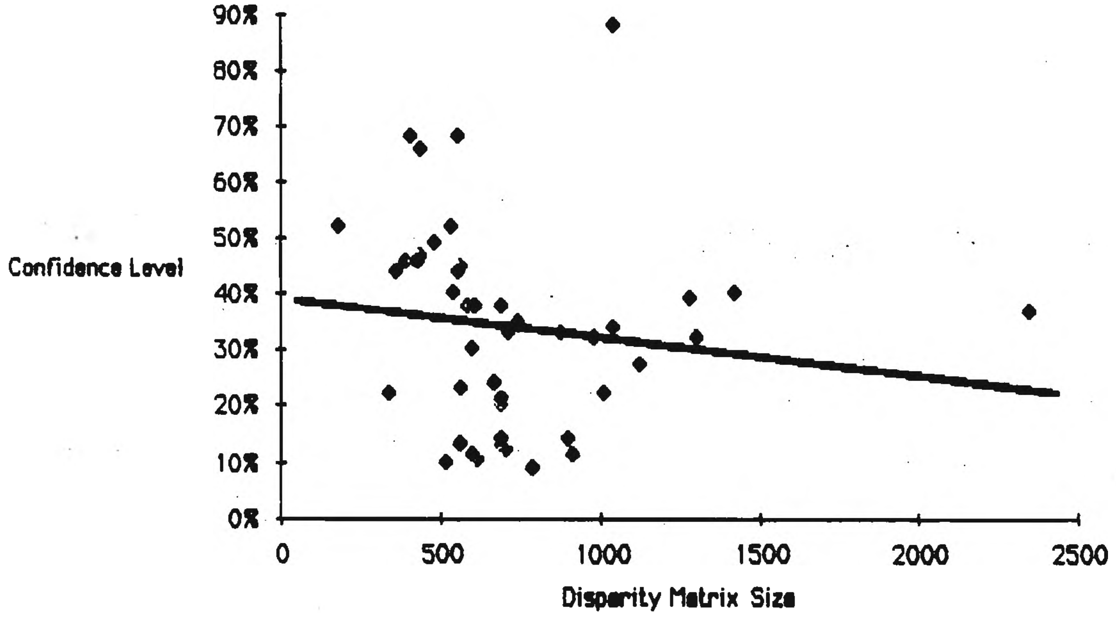


(d) Disparity matrix vs. number of initial sequences

Fig. 11 (Continued)

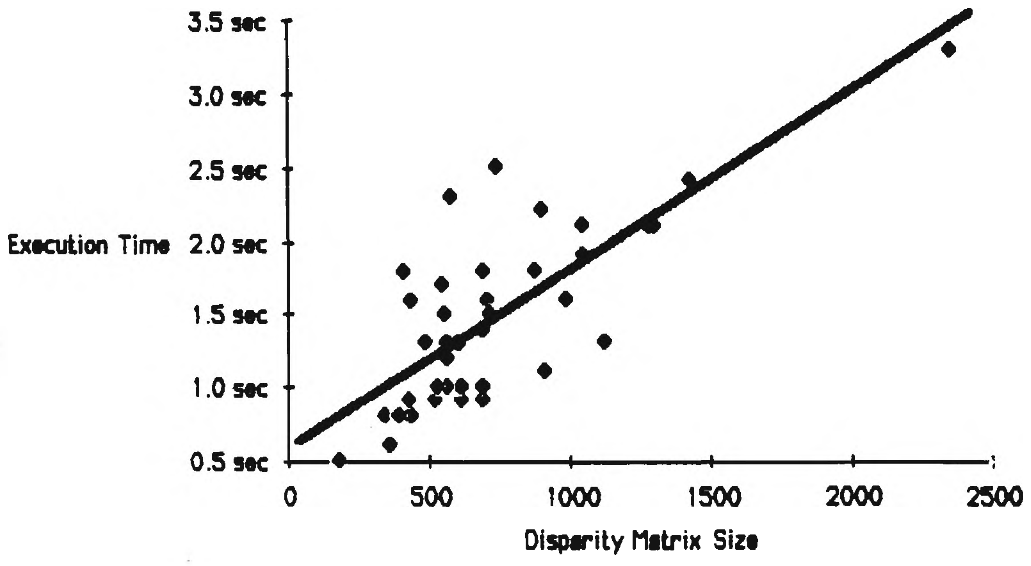


(e) Disparity matrix vs. number of thinned sequences



(f) Disparity matrix vs. confidence level

Fig. 11 (Continued)



(g) Disparity matrix vs. execution time

Fig. 11 (Continued)

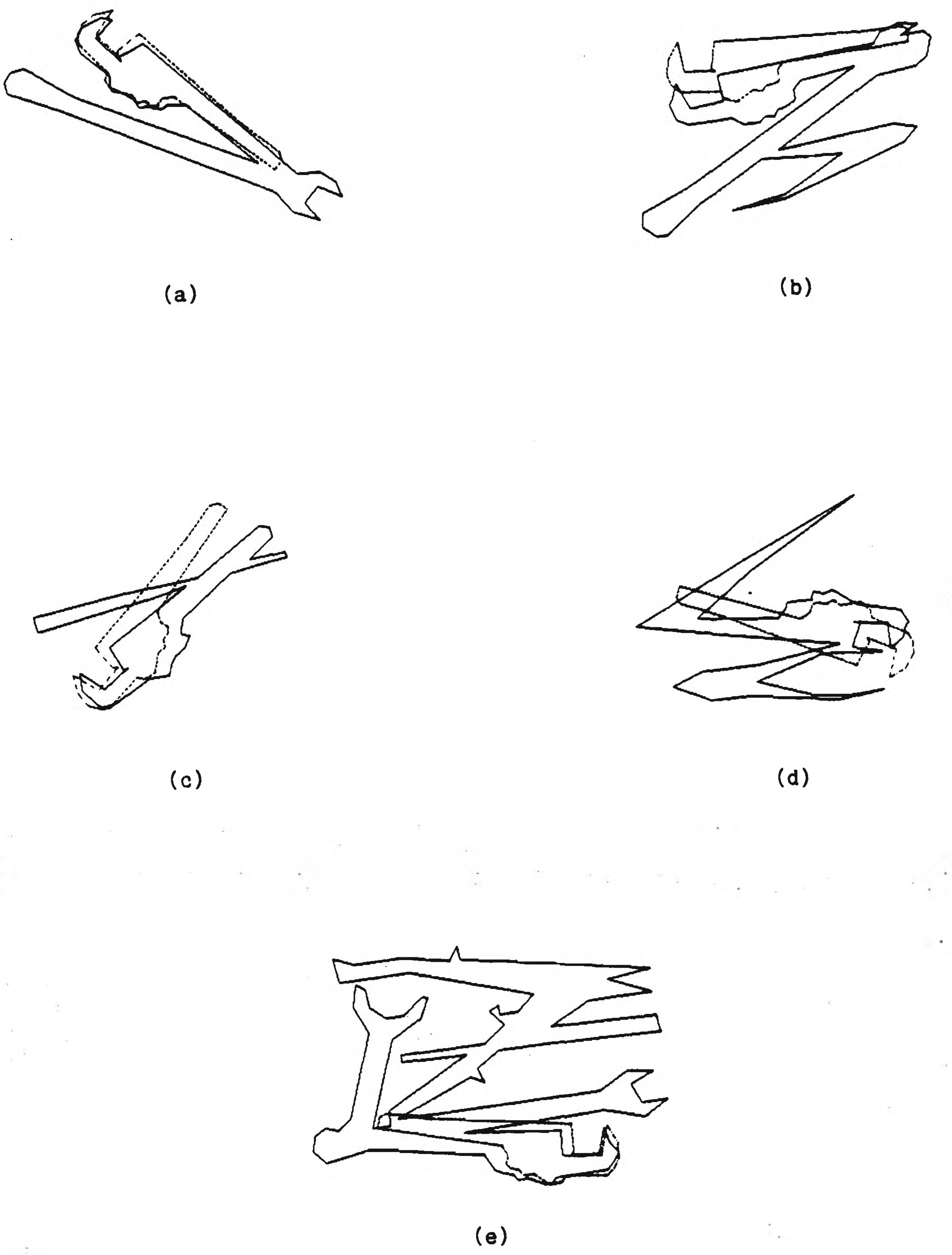
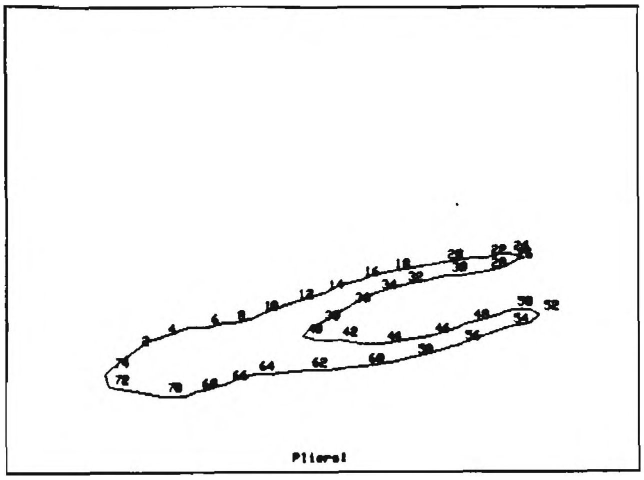
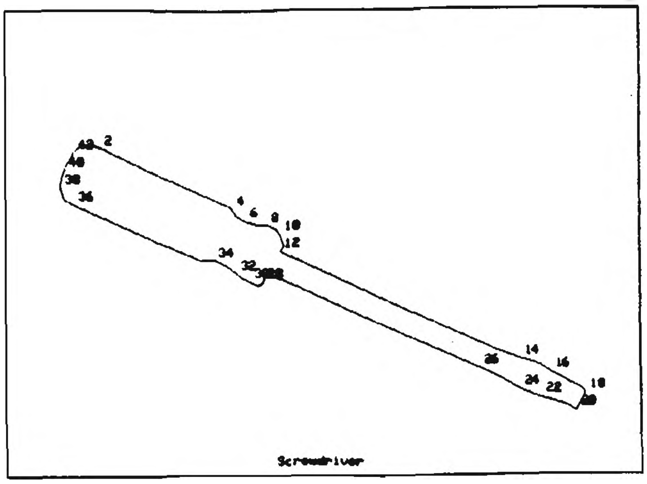


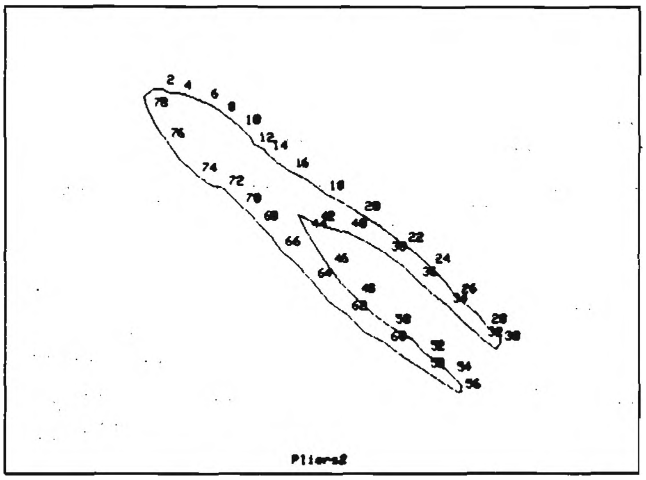
Fig. 12 Results of matching using the improved polygonal approximation for the model wrench in Fig. 4(g). Figs. 12(a), 12(b), 12(c), 12(d), and 12(e) correspond to Figs. 10(b), 10(c), 10(d), 10(e), and 10(t) respectively. Note that the matching results are superior to the results in Fig. 10.



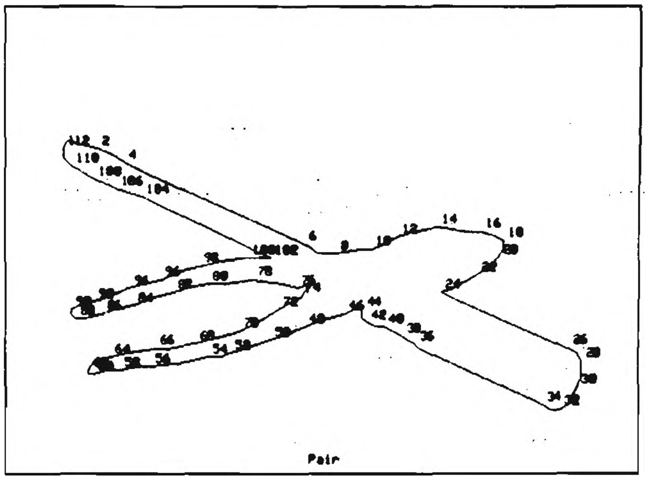
(a)



(b)

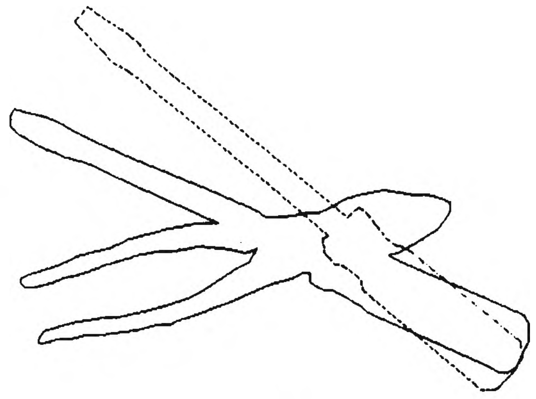


(c)

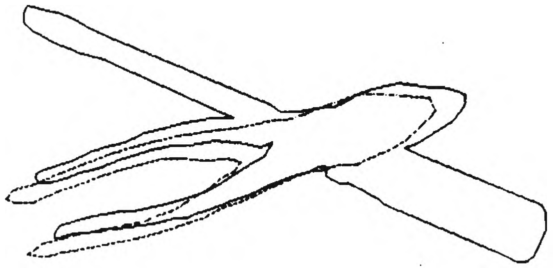


(d)

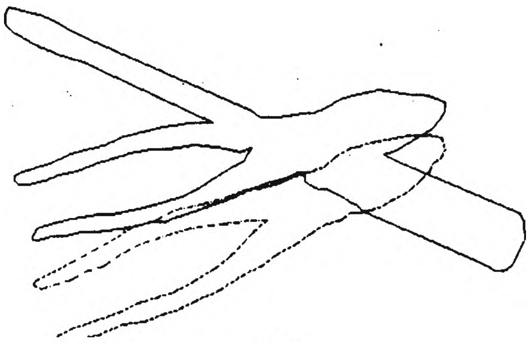
Fig. 13 Three models (a-c) and occluded image (d) generated by hand to demonstrate algorithm performance using models with a large number of boundary segments.



(a)

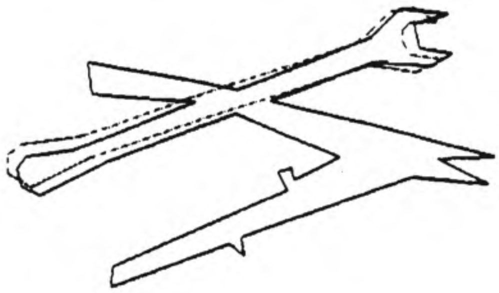


(b)

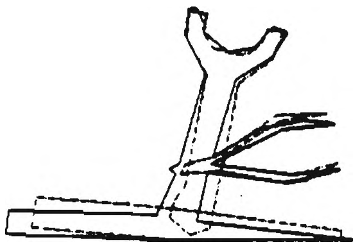


(c)

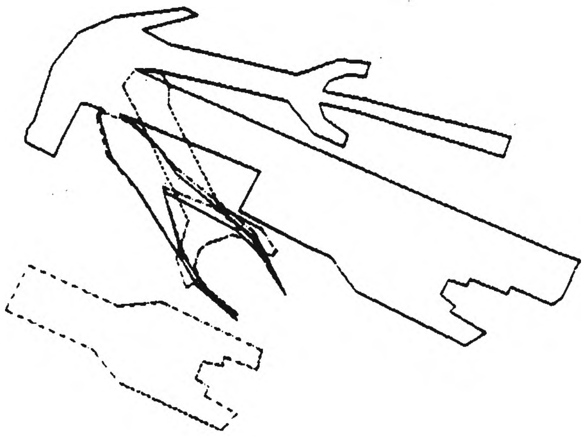
Fig. 14 Results of matching models in Fig. 13. Figs. (a) -(c) correspond to Figs. 13(a) -13(c).



(a)

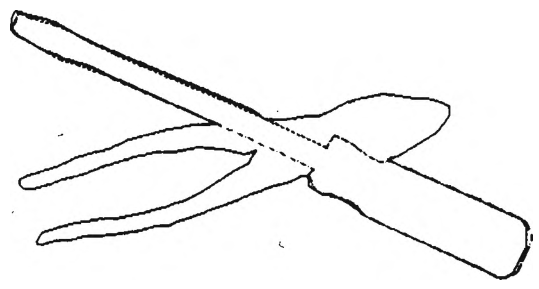


(b)

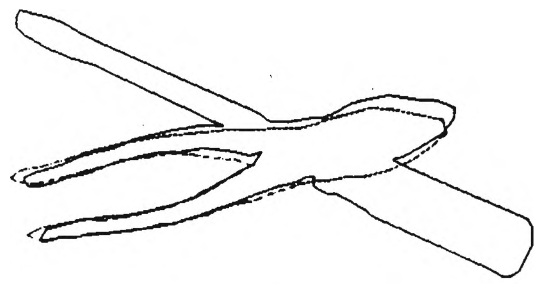


(c)

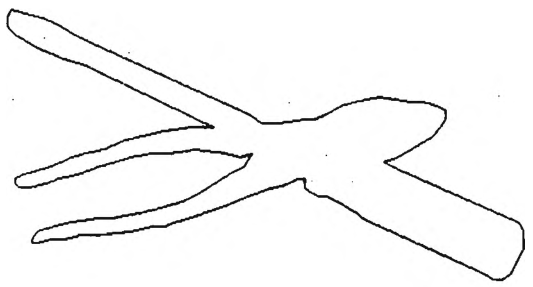
Fig. 15 Results of matching using the Price algorithm. Figs. 15(a), 15(b), and 15(c) correspond to Figs. 10(a), 10(j), and 10(s) respectively.



(a)



(b)



(c)

Fig. 16 Results of matching using the Price algorithm on the hand generated examples in Fig. 13. Figs. (a) -(c) correspond to Figs. 13(a) -13(c).

List of Figures

Fig. 11 Performance charts for the recognition algorithm (a to g).

(a) Disparity matrix sparsity
(b) Disparity matrix vs. number of initial clusters
(c) Disparity matrix vs. cluster size
(d) Disparity matrix vs. number of initial sequences
(e) Disparity matrix vs. number of thinned sequences
(f) Disparity matrix vs. confidence level
(g) Disparity matrix vs. execution time

Fig. 12 Results of matching using the improved polygonal approximation for the model wrench in Fig. 4(g). Figs. 12(a), 12(b), 12(c), 12(d), and 12(e) correspond to Figs. 10(b), 10(c), 10(d), 10(e), and 10(t) respectively. Note that the matching results are superior to the results in Fig. 10.

Fig. 13 Three models (a-c) and occluded image (d) generated by hand to demonstrate algorithm performance using models with a large number of boundary segments.

Fig. 14 Results of matching models in Fig. 13. Figs. (a) -(c) correspond to Figs. 13(a) -13(c).

Fig. 15 Results of matching using the Price algorithm. Figs. 15(a), 15(b), and 15(c) correspond to Figs. 10(a), 10(j), and 10(s) respectively.

Fig. 16 Results of matching using the Price algorithm on the hand generated examples in Fig. 13. Figs. (a) -(c) correspond to Figs. 13(a) -13(c).

Table 1 Model polygonal approximation summary.

Border Follow Results:

Boundary Points:	Minimum:	567
	Maximum:	1425
	Average:	870
	Median:	867
	Std. Deviation:	287

Curvature Maxima Results:

No. of Initial Segments:	Minimum:	18
	Maximum:	52
	Average:	32
	Median:	34
	Std. Deviation:	12
Initial Approximation Error:	Minimum:	20.9
	Maximum:	151.0
	Average:	53.0
	Median:	43.7
	Std. Deviation:	38.3

Split-Merge Results:

Error Maximum Value:	Minimum:	3.7
	Maximum:	17.3
	Average:	9.2
	Median:	7.7
	Std. Deviation:	4.3
Number of Splits:	Minimum:	22
	Maximum:	52
	Average:	39
	Median:	41
	Std. Deviation:	9

Table 1 (Continued)

Number of Merges:	Minimum:	30
	Maximum:	72
	Average:	53
	Median:	50
	Std. Deviation:	14
Number of Adjusts:	Minimum:	5
	Maximum:	86
	Average:	43
	Median:	46
	Std. Deviation:	24
No. of Final Segments:	Minimum:	5
	Maximum:	33
	Average:	18
	Median:	17
	Std. Deviation:	6
Execution Time (seconds):	Minimum:	34.9
	Maximum:	75.5
	Average:	45.8
	Median:	41.9
	Std. Deviation:	12.7

Table 2 Image polygonal approximation summary.

Border Follow Results:

Boundary Points:	Minimum:	1123
	Maximum:	4025
	Average:	2137
	Median:	2130
	Std. Deviation:	728

Curvature Maxima Results:

No. of Initial Segments:	Minimum:	21
	Maximum:	71
	Average:	35
	Median:	34
	Std. Deviation:	13
Initial Approximation Error:	Minimum:	70.6
	Maximum:	1200.7
	Average:	466.3
	Median:	523.8
	Std. Deviation:	317.1

Split-Merge Results:

Error Maximum Value:	Minimum:	15.1
	Maximum:	58.8
	Average:	24.8
	Median:	23.7
	Std. Deviation:	10.6
Number of Splits:	Minimum:	31
	Maximum:	94
	Average:	53
	Median:	51
	Std. Deviation:	15

Table 2 (Continued)

Number of Merges:	Minimum:	32
	Maximum:	73
	Average:	50
	Median:	48
	Std. Deviation:	14
Number of Adjusts:	Minimum:	42
	Maximum:	254
	Average:	100
	Median:	90
	Std. Deviation:	47
No. of Final Segments:	Minimum:	26
	Maximum:	71
	Average:	38
	Median:	37
	Std. Deviation:	12
Execution Time (seconds):	Minimum:	51.2
	Maximum:	182.3
	Average:	94.3
	Median:	85.6
	Std. Deviation:	32.3

Table 3   True verses experimental transformation values.

Image Name	Model Number	True Values Rot.	True Values Translation X	True Values Translation Y	Experimental Values Rot.	Experimental Values Translation X	Experimental Values Translation Y	Error Rot.	Error Translation X	Error Translation Y
Image 1:	Model 1	165.5	638	390	167.0	638	393	-1.5	0	-3
	Model 6	166.25	584	440	167.5	581	445	-1.25	3	-5
Image 2:	Model 6	125.25	624	108	121.8	615	98	3.45	9	10
	Model 7	33.25	169	-87	33.2	166	-82	.05	3	-5
Image 3:	Model 4	283.5	62	436	---	---	---	---	---	---
	Model 6	184.0	545	473	180.7	540	433	3.3	5	40
	Model 7	88.0	508	-1	120.0	549	223	-32.0	-41	-224
Image 4:	Model 7	113.75	674	35	123.3	681	104	-9.55	-7	-69
	Model 12	275.25	88	534	---	---	---	---	---	---
Image 5:	Model 4	91.0	508	-106	92.8	510	-90	-1.8	-2	-16
	Model 7	251.5	175	598	262.1	131	531	-10.6	44	67
	Model 12	297.0	-81	469	---	---	---	---	---	---
Image 6:	Model 2	77.5	402	-144	78.4	395	-133	-0.9	7	-11
	Model 3	184.25	552	523	183.6	553	516	0.65	-1	7
Image 7:	Model 2	266.5	94	471	---	---	---	---	---	---
	Model 3	182.0	557	518	175.5	581	488	6.5	-24	30
	Model 8	240.0	314	689	238.5	321	685	1.5	-7	4
Image 8:	Model 3	3.0	29	-11	---	---	---	---	---	---
	Model 9	228.25	271	594	227.1	271	594	1.15	0	0
Image 9:	Model 9	226.75	358	622	195.2	581	549	31.55	-223	73
	Model 14	85.0	488	-106	88.6	514	-83	-3.6	-26	-23
Image 10:	Model 9	335.0	-63	162	336.9	-56	152	-1.9	-7	10
	Model 12	258.5	115	483	258.6	120	487	-0.1	-5	-4
	Model 14	94.0	618	-74	94.1	622	-72	-0.1	-4	-2
Image 11:	Model 8	181.5	651	484	---	---	---	---	---	---
	Model 10	86.25	540	-64	79.1	459	-67	7.15	81	3
	Model 12	260.0	131	510	---	---	---	---	---	---
Image 12:	Model 6	45.75	223	-78	57.3	328	-62	-11.55	-105	-16
	Model 13	191.75	547	532	192.9	544	533	-1.15	3	-1

Table 3 (Continued)

		True Values			**Experimental Values**			**Error**		
			Translation			Translation			Translation	
Image Name	Model Number	Rot.	X	Y	Rot.	X	Y	Rot.	X	Y
Image 13:	Model 12	27.5	197	-100	---	---	---	---	---	---
	Model 13	229.0	277	607	299.2	73	360	-70.2	204	247
Image 14:	Model 2	75.5	425	-148	75.5	412	-135	0.0	13	-13
	Model 12	315.25	47	280	---	---	---	---	---	---
	Model 13	190.0	548	587	197.4	512	616	-7.4	36	-29
Image 15:	Model 4	297.0	32	362	299.1	32	349	-2.1	0	13
	Model 8	286.75	-2	497	284.7	2	500	2.05	-4	-3
Image 16:	Model 4	103.0	602	-17	98.8	576	-38	4.2	26	21
	Model 8	266.5	61	620	124.2	715	111	142.3	-654	509
	Model 14	329.75	-207	162	263.5	29	653	66.25	-236	-491
Image 17:	Model 2	258.5	102	529	235.8	186	437	22.7	-84	92
	Model 8	351.0	10	27	---	---	---	---	---	---
	Model 14	273.0	55	591	272.3	53	594	0.7	2	-3
Image 18:	Model 3	307.75	-79	299	301.8	-41	347	5.95	-38	-48
	Model 9	251.0	145	657	255.1	135	642	-4.1	10	15
	Model 10	272.25	-18	466	257.1	218	496	5.15	-236	-30
	Model 12	48.75	213	-214	---	---	---	---	---	---
Image 19:	Model 3	5.5	31	-19	1.5	5	-13	4.0	26	-6
	Model 4	35.0	70	-126	33.1	66	-120	1.9	4	-6
	Model 9	246.75	135	699	250.5	115	692	-3.75	20	7
	Model 10	241.0	278	545	242.7	269	546	-1.7	9	-1
	Model 12	66.25	416	-49	66.1	414	-47	0.15	2	-2
Image 20:	Model 1	178.0	599	524	179.0	594	526	-1.0	5	-2
	Model 6	43.75	229	-183	44.6	242	-187	-0.85	-13	4
	Model 7	286.25	18	342	---	---	---	---	---	---
	Model 9	339.0	-196	119	323.8	-174	200	15.2	-12	-81
	Model 12	89.0	565	-26	92.1	591	-19	-3.1	-26	-7

Table 4 Recognition performance summary.

Models	No. of Occurences in Images	Recognition Performance											
		1	2	3	4	6	7	8	9	10	12	13	14
Model 1	2(0)	2	0	0	0	0	0	0	0	0	0	0	0
Model 2	4(2)	0	2	0	0	0	0	0	0	0	0	0	0
Model 3	5(1)	0	0	4	0	0	0	0	0	0	0	0	0
Model 4	5(1)	0	0	0	4	0	0	0	0	0	0	0	0
Model 6	5(0)	0	0	0	0	5	0	0	0	0	0	0	0
Model 7	5(1)	0	0	0	0	0	4	0	0	0	0	0	0
Model 8	5(2)	0	0	0	1	0	0	2	0	0	0	0	0
Model 9	6(0)	0	0	0	0	0	0	0	6	0	0	0	0
Model 10	3(0)	0	0	0	0	0	0	0	1	2	0	0	0
Model 12	9(6)	0	0	0	0	0	0	0	0	0	3	0	0
Model 13	3(0)	0	0	0	0	0	0	0	0	0	1	2	0
Model 14	4(0)	0	0	0	0	0	0	1	0	0	0	0	3

This chart indicates the number of times that each of the models appears in the images. The recognition performance values indicate what each of the models matched in these images. For instance, row 1 indicates that model #1 appears twice in the images, and in both cases it has been recognized as model #1. On the other hand, model #8, which appears in the images 5 times, was mistakenly matched to model #4 in one case, was correctly matched 2 times, and was not matched at all in 2 cases. The values that appear in parentheses in the number of occurences column indicates the number of times that the model was not matched at all.

List of Tables