Pattern Recognition, Vol. 20, No. 2, pp. 199–211, 1987.
Printed in Great Britain.

0031-3203/87 $3.00+ .00
Pergamon Journals Ltd.

RECOGNITION OF OCCLUDED OBJECTS: A CLUSTER-STRUCTURE ALGORITHM*

BIR BHANU† AND JOHN C. MING
Department of Computer Science, University of Utah, Salt Lake City, UT 84112, U.S.A.

(*Received* 3 *December* 1985; *in revised form* 9 *July* 1986)

Abstract—Clustering techniques have been used to perform image segmentation, to detect lines and curves in images and to solve several other problems in pattern recognition and image analysis. In this paper we apply clustering methods to a new problem domain and present a new method based on a cluster-structure approach for the recognition of 2-D partially occluded objects. Basically, the technique consists of three steps: clustering of border segment transformations; finding continuous sequences of segments in appropriately chosen clusters; and clustering of sequence average transformation values. As compared to some of the earlier methods, which identify an object based on only one sequence of matched segments, the new approach allows the identification of all parts of the model which match in the occluded scene. We also discuss the application of the clustering techniques to 3-D scene analysis. In both cases, the cluster-structure algorithm entails the application of clustering concepts in a hierarchical manner, resulting in a decrease in the computational effort as the recognition algorithm progresses. The implementation of the techniques discussed for the 2-D case has been completed and the algorithm has been evaluated with respect to a large number of examples where several objects partially occlude one another. The method is able to tolerate a moderate change in scale and a significant amount of shape distortion arising as a result of segmentation and/or the polygonal approximation of the boundary of the object. A summary of the results is presented.

Clustering Occlusion Recognition Segment matching Sequencing Shape matching

I. INTRODUCTION

The problem of occlusion in a two-dimensional scene introduces errors into many existing 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. Since occlusion will be present in all but the most constrained environments, the recognition of partially occluded objects is of prime importance for industrial machine vision applications and to solve real problems in the military domain and factory automation.[6] Several methods have been developed[1,3,4,9,17,23] which do not rely on global shape features. The approaches to solve the occlusion problem can be classified either as boundary-based[3,10,17,23] or local-feature-based.[9] The former approaches use methods such as relaxation or boundary-matching while the latter approach makes use of the available local features such as holes or corners and their relationships. However, these techniques are computationally intensive. They cannot handle minor 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[19] to use a conceptually simple technique to solve the occlusion problem by following the *order* of matched line segments in the model and the apparent object; the apparent object is formed as a result of occlusion of two or more objects. Price uses a device called a disparity matrix. Initially, the algorithm assumes that we have a linear border approximation of a given model and an image (apparent object). Price's method then compares every line segment in the model with every line 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, an error code is placed at the appropriate location in the matrix. After all line segments have been compared, the matrix contains the offsets, or disparities, for all line segments. By

*This work was supported in part by NSF Grants DCR-8506393, DMC-8502115, ECS-8307483 and MCS-8221750.
† Present address of the corresponding author: Department of Computer Science, 3160 Merrill Eng. Building, University of Utah, Salt Lake City, UT 84112, U.S.A.

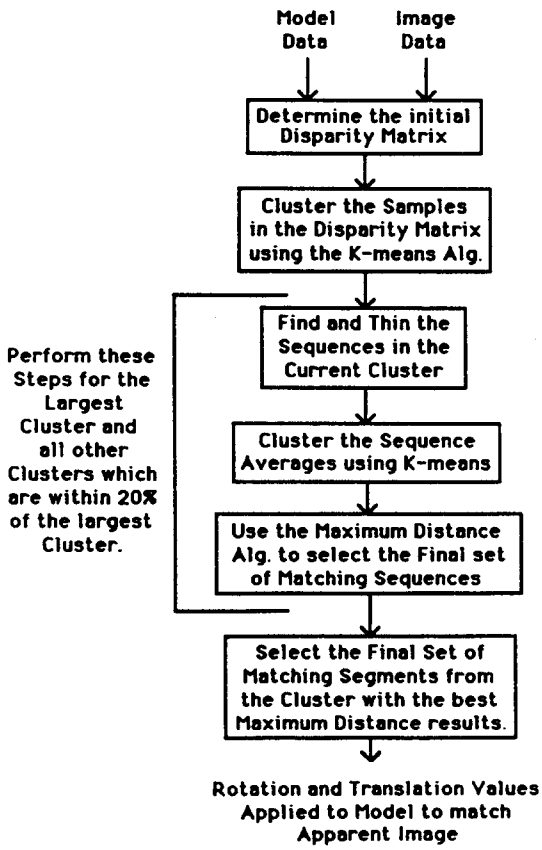

Fig. 1. Block diagram of the clustering-based occlusion algorithm.

traversing this newly formed matrix diagonally, the longest sequence in the matrix that contains compatible entries is found. 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 also very expensive computationally. Since it must treat every entry in the disparity matrix as a possible starting location of a sequence, it traverses the matrix once for every entry that exists. While this fact does not pose a problem in the simplest cases, matching takes a long time for models and images with more than about 20–30 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. It uses only the longest sequence found in the traversal because it cannot determine the compatibility between multiple sequences. 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.

In order to overcome the factors which caused the Price method to fail, we have used a cluster-structure approach which allows the recognition of a given object and provides information about the orientation and position of the objects in the image.[7] Some techniques and structures used by Price have been retained, but the method of matching is entirely new. In the past, Stockman *et al.*[22] have used clustering to recognize 2-D objects. They identify several types of local features and perform clustering on the rotation and translation values of similar features present in both the model and image. However, our cluster-structure algorithm applies the clustering concepts in a hierarchical fashion, which reduces the amount of computational effort and time required by the recognition process. The clustering algorithm contains three major steps: clustering of line segment transformations; locating sequences of line segments in specific clusters; and secondary clustering of sequence average transformations. Sequencing is efficient since the sequences are found in one pass over the data. In addition, the method only relies on boundary segments, eliminating the need for feature extraction within the images. The motivation behind the cluster-structure approach and how the clustering techniques can be useful in solving the occlusion problem is given in Section II. Section III presents the algorithmic description of the new method. Section IV provides a number of examples illustrating the capabilities of the technique. A comparison with the Price method is also presented here. Section V discusses the applications of the clustering methods in three dimensions. Finally Section VI presents the conclusions of the paper.

II. PRINCIPLE OF THE TECHNIQUE

In an unstructured environment, for example parts going over a conveyor belt, occlusion may occur in a large number of images that have to be processed. This problem requires the development of an algorithm which can easily handle occluded scenes. Figure 1 shows the block diagram of the clustering-based occlusion algorithm. It is based on the premise of successive applications of clustering to the model and image data and the use of structural information in an environment in which the data reduction takes place as the recognition process progresses and the confidence in recognition is increased.

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.[13,14] Clustering techniques are commonly used in pattern recognition and image processing. [For a recent review see Ref.(16).] A significant problem inherent in using the clustering techniques involves the choice of the number of clusters to be used at any given time. Since the choice depends on the structure of the data that is being clustered, the number of clusters cannot usually be a constant value in a given application. As an example, when applying clustering techniques to solve the occlusion problem, the number of clusters must be altered, depending on the lighting conditions, the segmentation techniques used, the amount of occlusion present, and many other factors. Fortunate-

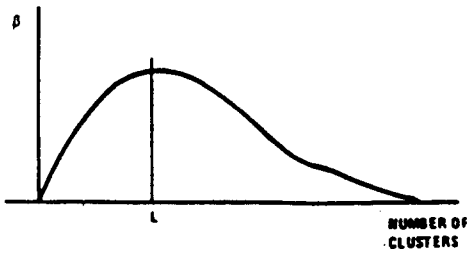

Fig. 2. A clustering quality measure.

ly, there are measures which can be used to find the intrinsic number of clusters.[8,11,12,16] 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.[14] 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 β 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. $\beta = \mathrm{Tr}(S_w)\mathrm{TR}(S_b)$. As the number of clusters increases, the value of β will reach a maximum and then slope towards 0. The number of clusters at which the value of β is a maximum is the desired value and gives the best results. Figure 2 shows the behavior of β with respect to the number of clusters. Thus, by setting the number of clusters to be one, clustering the samples, computing the β value, comparing the β value with its last value, and continuing until the maximal β value is reached, a program can find the best number of clusters in a given data set.

In determining the number of clusters using the above method, a clustering technique such as the K-Means Algorithm could be used. After every sample has its feature vector computed, the algorithm creates an arbitrary set of K cluster centers into which all samples will be placed. In order to determine which center to place a given sample in, the algorithm computes the distance from the sample to each of the cluster centers. This distance is merely the Euclidean distance from the sample to the cluster center in the feature space. The sample belongs to the cluster which is closest to it. When every sample has been assigned to a unique cluster, the algorithm 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 algorithm 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 said to be compatible with each other. Thresholds help to determine when cluster centers have become stable. Since the Euclidean distance may be affected by the choice of the features present in the feature vector, the feature values are normalized so that each feature contributes equally to the overall distance.

The above techniques are computationally efficient, even when the number of samples is very high, thus allowing the vision process to be fast as well as accurate. Since the clustering method groups all sets of compatible matches into a single cluster, regardless of their position in the image, it can find multiple sequences in a model which may match in the given image. While the Price method was only able to find a single best matching sequence, the new procedure will find as many sequences in the image as possible. Also, since the traversal of matrices in Price's method is computationally expensive, the new technique improves the speed of the matching as well. The use of clustering and the performance measures in the body of the algorithm are discussed in the next section.

III. ALGORITHM DESCRIPTION

As shown in Fig.1, the algorithm consists of the following main computational steps:

(1) Disparity matrix
(2) Initial clustering
(3) Sequencing
(4) Final clustering
(5) Transform computation.

Each of these steps will be described individually in this section. Two sets of data are assumed to be given. 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.

Disparity matrix

The first step of the algorithm consists of the formation of the disparity matrix. From the set of vertices for the object and the image, the algorithm determines the length of each border segment and the interior angles between successive segments. These interior angles are referred to as successor angles for each segment. The length of each of these segments as well as the angle between successive segments comprise the only information needed by the algorithm to recognize objects and find their position and orientation. At this point, every segment in the object model will be compared with every segment in the image in order to find all possible matches of the model segments in the image. This comparison utilizes the segment length and successor angle for each model segment. If a segment in the image has the same length and successor angle, within thresholds, as a segment in the model, the pair of segments are considered compatible and the rotational and translational disparity between the pair of segments is computed. These disparity values are stored in the disparity matrix, indexing the entry according to the segment

number in the model and the image.

This process continues until all segments have been compared. Now the range of rotational and translational values present in the matrix are determined, and the disparity matrix values are normalized so that the rotation and translation values contribute equally in the clustering step described next. The normalized values are kept in a normalized disparity matrix, since the initial disparity matrix needs to be retained for later use. This matrix is similar in structure to the matrix used by Price.[19] However, in addition to the rotational offsets, we also place translational offsets in the disparity matrix.

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

Initial clustering

After all of the normalized values have been placed into the disparity matrix, the algorithm clusters these values where the feature vector is merely the normalized rotational and translational offsets for each of the pairs of line segments. The initial number of clusters is set equal to one and in the application of K-means algorithm the first sample becomes the first cluster center. The clustering proceeds as described in the previous section. At each step, all of the samples are clustered, the value of the new cluster centers are recomputed, and this process continues until none of the cluster centers change their positions. Now for the current cluster results, scatter matrices are computed and the value of β is determined. The algorithm then compares the current β value with the last β value. If the value has decreased, then the previous β value and the number of clusters become the final result of this processing step. Note that the initial clustering step is similar to the generalized Hough transform[2] in its ability to detect similarities in local border shapes of objects. The generalized Hough transform might provide a faster alternative to the initial clustering step.

This step of the algorithm takes the most computational time of all of the steps due to the large number 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, 500 samples may be present in the disparity matrix which satisfy the length and angle thresholds. If the program has to cluster these samples three times until β is maximized, 1500 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 remaining steps in the algorithm to determine the location and orientation of the model in the image. 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. The confidence metric is discussed below. The program now passes each cluster that has been selected to the following algorithm steps, one at a time.

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 previous sample indicates that segment one 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. If [2,28] is found, we then check for [3,29] and so on. Each of these samples is marked with the current sequence number. Once the current sequence can no longer be continued, we find the next unmarked sample in the cluster data and repeat the process with this new sample. Since there may be some missing and extra segments in the model and the image as a result of segmentation, polygonal approximation, and various other reasons, we allow up to two extra or two missing segments when finding the sequences. This procedure continues until all possible sequences have been located in the data of the current cluster. All sequences of border segments are found in one pass over the cluster data so this step is quite efficient. Sequencing 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 are discarded. Since these samples are not members of any sequence, they usually represent the extraneous data in the cluster. The program also removes any sequences which have a segment count of less than three. Three segments make 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 and have low confidence. Even if the sequences indicated 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 of the algorithm is to compute the rotational and translational average 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 sample, the program must check every sample in order to locate the best choices. However, because the clustering results have greatly reduced that number 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.

Final clustering

Using the sequence averages obtained from the previous step, the algorithm clusters these values to find those sequences which lead to the same rotational and translational results. Note that in this application of clustering we are clustering sequence average rotations and translations. In other words, we are now working at a higher symbolic level of recognition. As with the initial clustering, the program uses the iterative technique of clustering, evaluating, clustering, etc. After the value of β 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 present in the disparity matrix, this step is much faster since we have eliminated so much data in the earlier steps. In all the examples discussed here, the number of sequences is less than 100, with an average somewhere near 30 – 40. Also, since the sequencing step has eliminated a good deal of the erroneous data, the β value quickly reaches its maximum and this steps ends.

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 are 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% or more usually lead to the correct transformation. Of course, its value depends upon the degree of occlusion within the image.

IV. RESULTS

Image acquisition and polygonal approximation

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 model for each of these tools was created by first acquiring an image of the object lying by itself on a backlit table. The image is obtained with a camera and a commercially available digitizer. The digitized picture of the scene is 576 pixels wide by 720 pixels high. These dimensions correspond to about 68 pixels/in. After the image has been digitized, it is then transferred to a Vax 11/780 computer for the remainder of the processing. Several of the images of the models that were collected appear in Fig.3.

Once the image has been obtained, the program finds the border of the tool using a simple border-follow algorithm. This procedure traverses the boundary of the object in the image and marks the pixels which lie on the border. The number of boundary points for the tools range from 567 to 1425 pixels. After locating the boundary of the object, the program uses a curvature-maxima algorithm to compute an initial border approximation of the object. This procedure uses the local curvature at every border pixel in conjunction with a smoothing factor to find the approximation of the object. The smoothing factor controls the quality of the initial polygonal approximation of the object.[20] Using a smoothing factor of eight results in a range of 18 – 52 segments in the models.

Using both the initial border approximation and the border pixels themselves, the program then uses the split-merge technique[18] to determine the final border approximation for each of the objects. This method splits all border segments with bad approximations and combines all pairs of adjacent segments that do not cause a dramatic change in the model representation. The number of final border segments varies from five to 33 for the 14 models that were used. The final polygonal approximations of the model images which appear in Fig. 3 are shown in Fig. 4.

The task of obtaining images to be processed proceeds in exactly the same manner as in the model acquisition. The images are obtained using the same hardware and are then moved to the Vax 11/780. These images are then processed with the border-follow, curvature-maxima, and split-merge algorithms. Some examples of the images collected in this test run appear in Fig. 5. For this particular experiment, 20 images were collected and then processed. In these 20 images, 56 instances of the 14 models are present. The number of boundary points varied from 1123 to 4025 pixels.

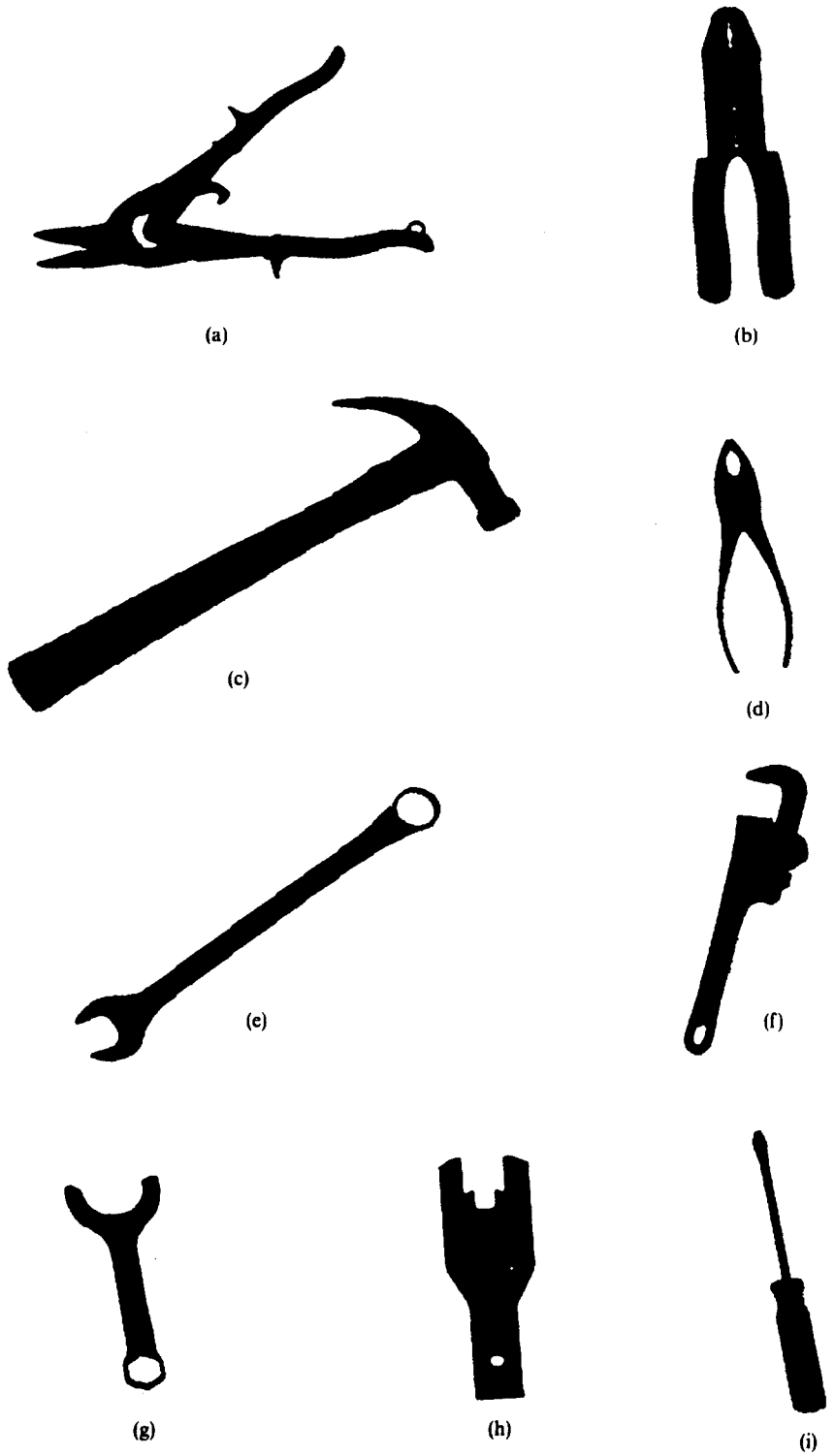

Fig. 3. Images of the object models (a–i).

The smoothing factor used to obtain the initial image approximation is 24 for all images. The number of curvature maxima segments ranges from 21 to 71. The final number of border segments received from the split-merge procedure range from 26 to 71 segments. The polygonal approximations of these images are presented in Fig. 6.

Model based recognition

Once all of the models and images have been collected, the clustering algorithm is 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, 47 (84%) models were correctly matched.

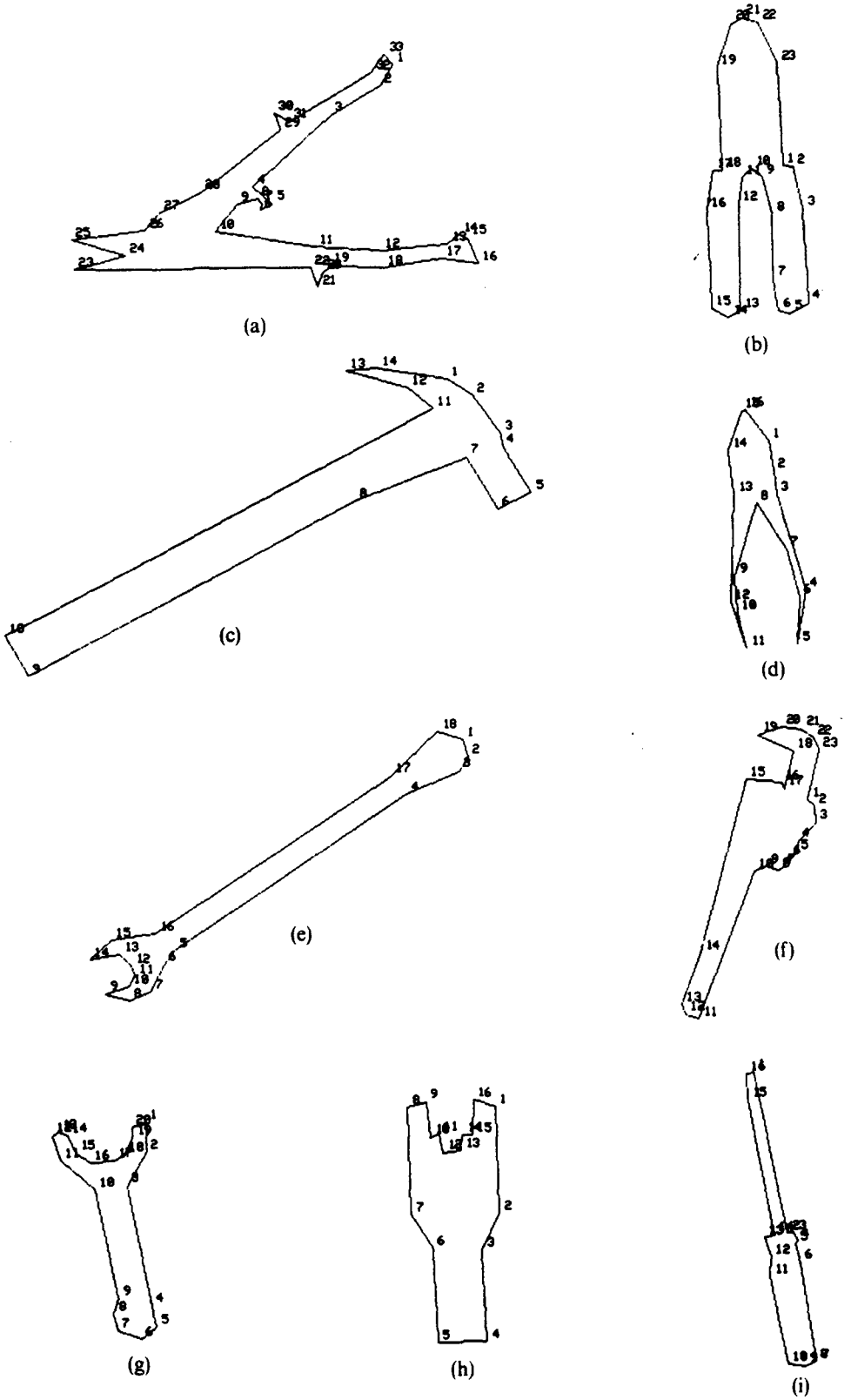

Fig. 4. Polygonal approximation of the object models (a–i).

Four of the 56 models were mistakenly matched to a different model. The remaining five model instances could not be matched. Figure 7 shows the matching results for the five images that were shown in Fig. 5. Solid lines show the polygonal approximation of the images using the split-merge algorithm. The dashed and dot-dashed lines show the polygonal approximation of the model at its matched location in the image.

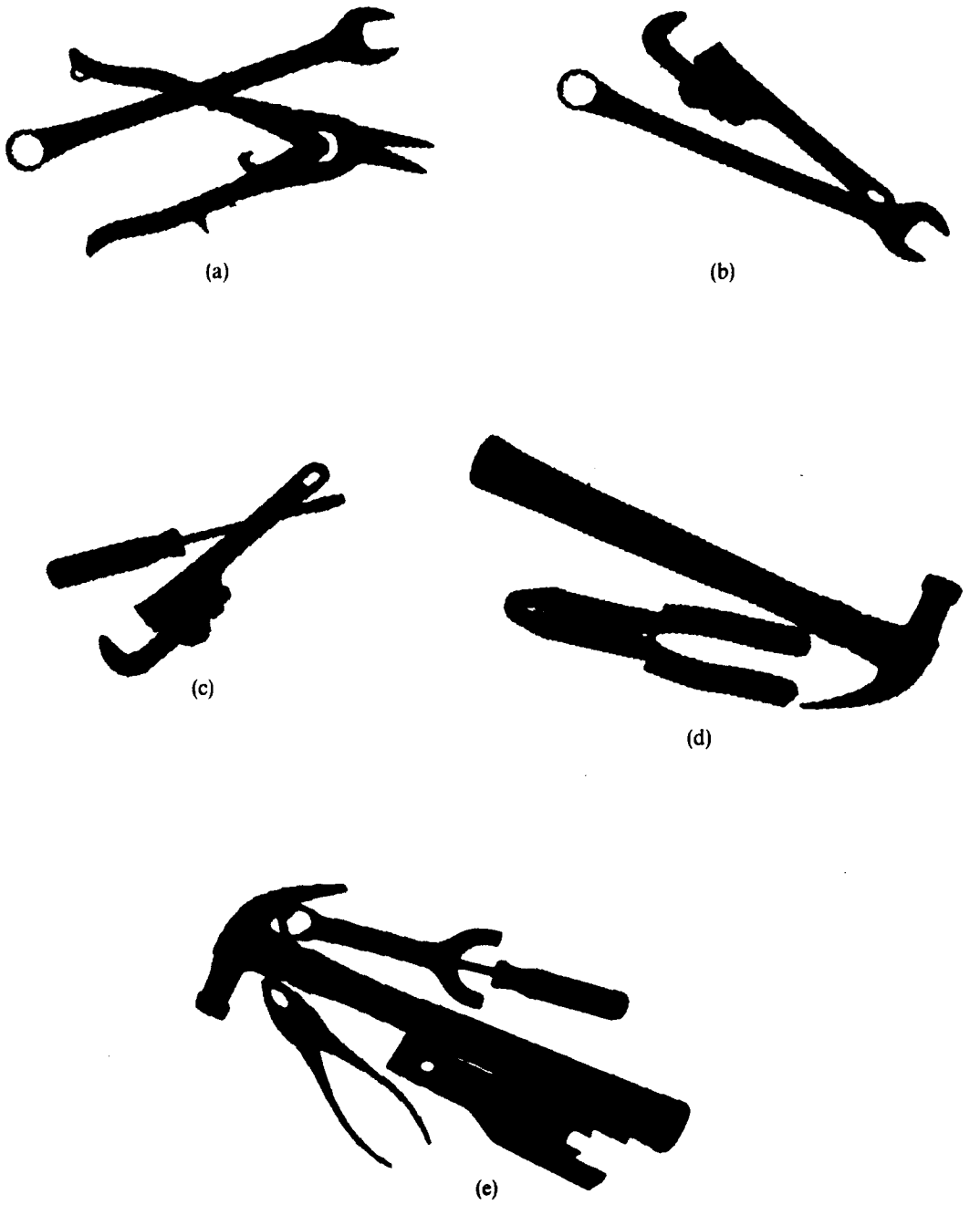

Fig. 5. Images of the occluded objects (a–e).

Dot-dashed lines indicate the segments which were matched while the dashed lines show the segments which did not contribute to the matching.

Out of the 47 models that were correctly matched in the images, seven of these matches were not successfully found until the polygonal approximation of the appropriate model was improved. Of the five model instances 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 occurs and the transformation determined by the program is better. For example, in Fig 7(c), the pipe wrench has been properly matched, but the transformation of the model is not very good due to the large difference in polygonal approximation. Figure 8 shows the results after the representation of the model has been substantially improved within the image. Now the new transformation is only slightly improved. However, more model segments were matched in this case which led to a higher confidence level. Note that a small visible change in scale occurred between the models

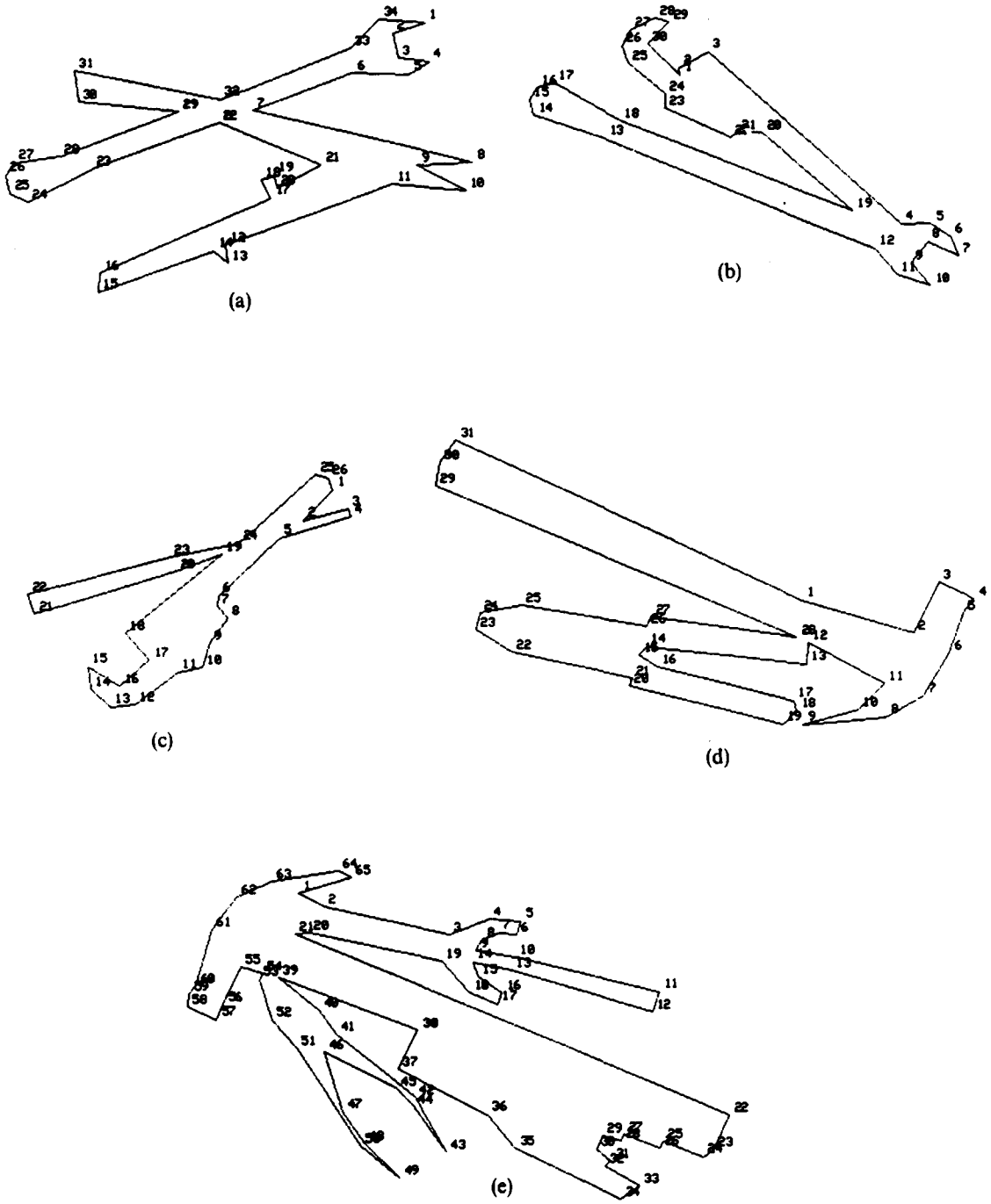

Fig. 6. Polygonal approximation of the occluded objects (a–e).

and their representations in the image during the process of image acquisition.

Execution times for the clustering method range from 0.5 to 3.2s on a Vax 11/780. No attempt was made to optimize the C code. After each matching has been determined, the program also computes a confidence level based on the segments which contributed to the matching. This value is computed by dividing the cumulative length of all matched segments by the cumulative length of all segments in the model. For this set of images, confidence levels vary from 0 to 98%. The error analysis of the matchings which were correct yield a mean rotational error of $-0.14°$ and a standard deviation of 8.68°. The mean translational errors are -11.83 and -7.65 pixels for translation in x and y, respectively. These translational error values correspond to -0.174 in. in x and -0.1125 in. in y.

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 9 shows the results

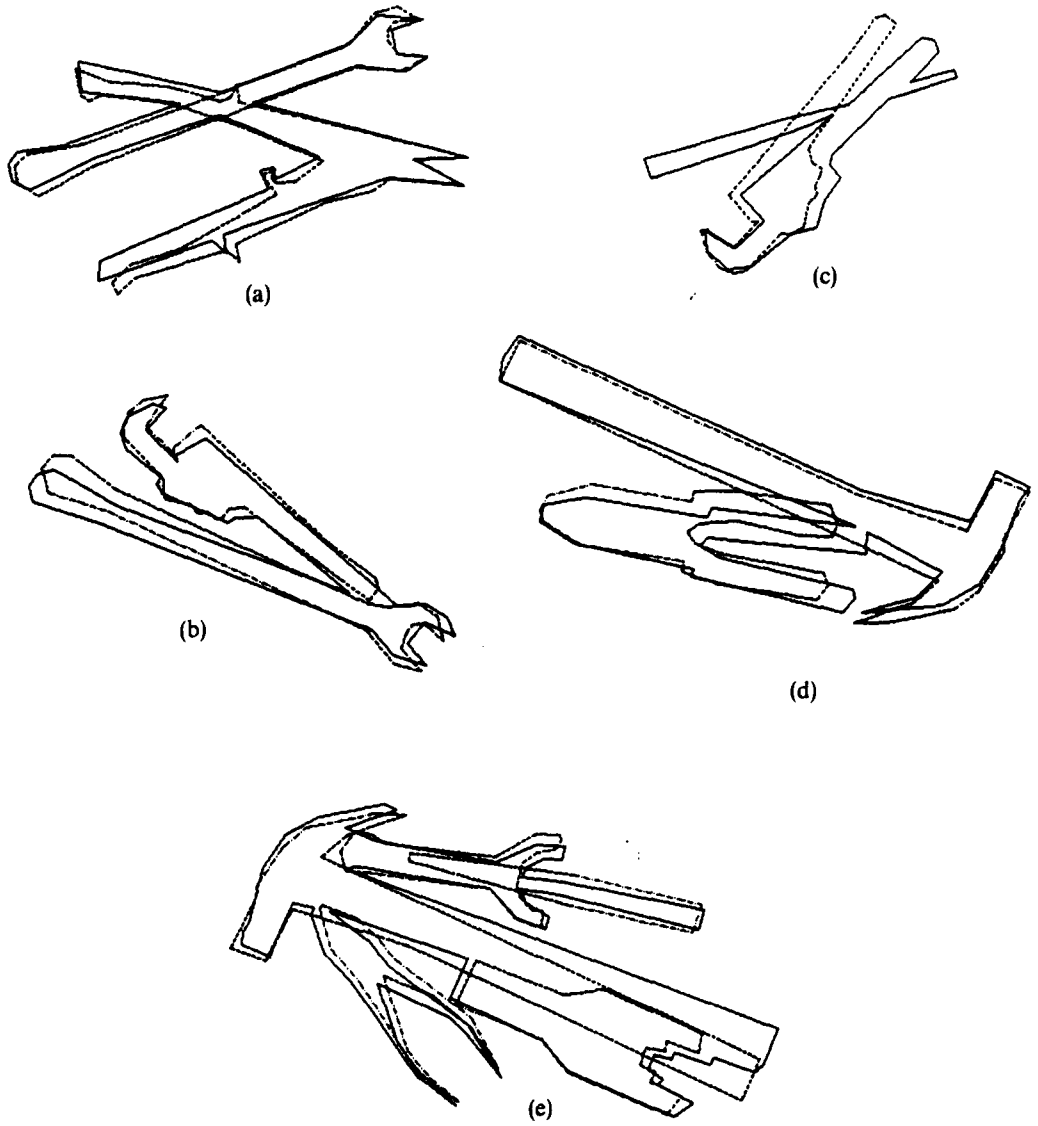

Fig. 7. Results of matching for the occluded images shown in Fig. 5. Solid lines show the polygonal approximation of the image. Dashed lines indicate the model segments that did not match while dot-dashed lines represent the model segments which were matched in the image.

of this matching. Out of the seven models present in Fig. 9, only two were properly matched. These results can be compared with the clustering results in Figs 7(a) and (e). 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, only selects the longest sequence.

V. 3-D APPLICATIONS

We have identified several problems in three dimensional scene analysis which can be solved using clustering techniques.[5] The recognition method described in the previous sections can readily be extended into three dimensions. Stockman and Esteva[21] have used clustering techniques in the past to determine 3-D object pose from two dimensional image points. However, we extend our two dimensional methods to 3-D under the assumption that our data set is the position and orientation of the planar faces of the objects in a range image. If we have a three dimensional description of the model as well, the 2-D clustering technique can be directly extended into the 3-D realm. In addition, we have investigated applications of clustering on range data to find planar patches. As with the 2-D algorithm, the clustering techniques are applied in a hierarchical fashion to reduce the amount of data computation. Each of these areas will now be described briefly.

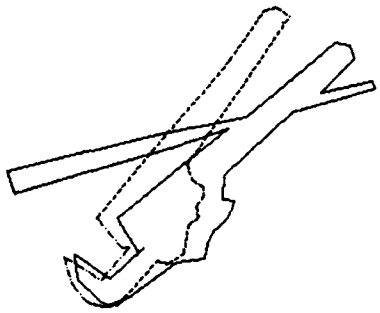

Fig. 8. Results of matching using the improved polygonal approximation for the model wrench [in Fig. 5(c)] only. By comparing this figure with Fig. 7(c) note that the transformation results did not change substantially. However, more segments of the model were matched, which provided a higher confidence level in the matching.

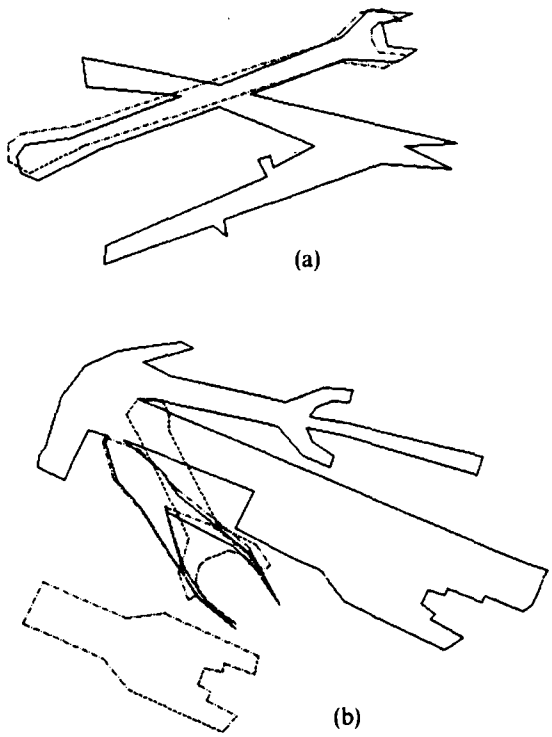

Fig. 9. Results of matching using the Price algorithm. Figures 9(a) and (b) correspond to Figs 7(a) and (c) respectively.

If we know the position and orientation of the planar faces on the surface of an object, we can use this information to aid in the recognition process when we are given only one view or an occluded view of the object and a 3-D model having a similar representation. The clustering method discussed earlier can be naturally extended into three dimensions. The line length and angle between successive segments in 2-D now become the planar area and angles between adjacent planes in the three dimensional domain. These features are used to determine compatibility between planar faces. The disparity matrix is formed by comparing all faces in the known model with the faces in the image. If compatible, the three dimensional rotation and translation disparities are entered in the disparity matrix. Clustering can then be used to group sets of planar faces that have similar transformations. In this case, we are merely clustering with a larger feature vector than the one used in the 2-D technique. Sequencing in two dimensions corresponds to planar region growing in the 3-D algorithm, providing the necessary structural information. In each of the initial clusters, the algorithm marks all compatible groups of adjacent faces. The planar region is allowed to grow in any direction until no further compatible faces can be located. The next unmarked face is then found and the region growing method is repeated on the set of remaining unmarked faces. This process continues until all regions of compatible planar faces have been located. Once this procedure is complete, the algorithm computes the average transformation determined by each of these regions. This step is analogous to the computation of the sequence averages in the two dimensional case. In the second clustering stage, we cluster these average transformations to obtain the groups of planar regions that yield the same rotation and translation. The algorithm now computes the confidence level of the matching determined by the regions in each of the initial clusters. Finally, the matching that leads to the best confidence level is selected as the final model transformation. The development of the 3-D algorithm is under investigation and will be reported in the future.

Identification of planar patches or faces on the surface of a three dimensional object using range data is also one of the current areas of interest. This range data is available from several sources, including range information from a laser range finder as well as data points obtained from a CAD (Computer Aided Design) model of the object. The first step of this process is to compute the surface normals for every point in our data set. We first identify all the neighbors of each point using a k–d tree algorithm[15] and then use a least-squared method to represent the plane which approximates the local surface patch. The surface normal is then obtained from this plane. The surface normals are converted to unit normal vectors for consistency in the clustering step. Once the surface normals are known, the use of clustering allows surface points with similar normal values to be easily grouped together. The K-means algorithm along with the performance measures previously discussed are again used in the clustering process. In this step of the algorithm, the normals at each point are clustered. Once the initial clustering has been completed, we perform an additional clustering on each of these clusters separately. Since the first step only considered the orientation of the surface normals, it is possible that non-adjacent surface patches that have similar orientations on the object have been included in the same cluster. By applying the clustering methods on the x, y and z values of each of the data points, the algorithm separates any samples in these clusters that are not spatially adjacent. Since the performance

measures are used to find the inherent number of clusters, the initial clusters that contain more than one planar face are split up while the clusters that contain only one face are left intact. The algorithm then uses the least-squared procedure on each of the final clusters to determine the theoretical position and orientation of each of the planar faces on the surface of the object. This symbolic representation of the surface of the object can then be used in the cluster-structure algorithm as discussed earlier.

VI. CONCLUSIONS

Based on the results presented in this paper, we conclude that the cluster-structure algorithm is a robust approach to solve the occlusion problem in 2-D. The data and the amount of computation reduce in a systematic and hierarchical manner. Since the technique does not limit itself to a single sequence of line segments on the border of an object, it can locate all of the matched segments of the model, which accounts for the high success rate. The program was not highly successful in the instances of severe occlusion, where a given model has only about five percent of the total number of segments visible in the image. In those situations, even an expert would have problems locating a model within the image. The final matching results of the algorithm are in a form well suited for higher-level analysis, if necessary. The reliability of this technique might be increased by including relationships of prominent object features such as holes, parallels, and perpendiculars.

Future modifications to this algorithm will include the ability of the algorithm to handle cases when the model has a large amount of symmetry and considerations for grasping strategies by a manipulator.[7]

In addition to the success of the 2-D work, we also believe that the application of the clustering methods to the three dimensional domain also holds good promise of success. Note that in 3-D, both symbolic representation and matching can be obtained using the same clustering techniques used in the 2-D cluster-structure approach. Work completed to date suggests that the extension to three dimensions should provide reliable results.

SUMMARY

The recognition of partially occluded objects is of prime importance for industrial machine vision applications and to solve real problems in the military domain and factory automation. The problem of occlusion in a two dimensional scene introduces errors into many existing 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.

Clustering techniques have been used to perform image segmentation, to detect lines and curves in images and to solve several other problems in pattern recognition and image analysis. In this paper we apply clustering methods to a new problem domain and present a new method based on a cluster-structure approach for the recognition of 2-D partially occluded objects. Basically, the technique consists of three steps: clustering of border segment transformations; finding continuous sequences of segments in appropriately chosen clusters; and clustering of sequence average transformation values. As compared to some of the earlier methods, which identify an object based on only one sequence of matched segments, the new approach allows the identification of all parts of the model which match in the occluded scene. We also discuss the application of the clustering techniques to 3-D scene analysis. In both cases, the cluster-structure algorithm entails the application of clustering concepts in a hierarchical manner, resulting in a decrease in the computational effort as the recognition algorithm progresses. The implementation of the techniques discussed for the 2-D case has been completed and the algorithm has been evaluated with respect to a large number of examples where several objects partially occlude one another. The method is able to tolerate a moderate change in scale and a significant amount of shape distortion arising as a result of segmentation and/or the polygonal approximation of the boundary of the object. A summary of the results is presented.

REFERENCES

1. N. Ayache, A model-based vision system to identify and locate partial visible industrial parts, *IEEE Conf. on Computer Vision and Pattern Recognition*, pp. 492–494, June (1983)
2. D.H. Ballard, Generalizing the Hough transform to detect arbitrary shapes, *Pattern Recognition* **13**, pp. 111–122 (1981).
3. B. Bhanu and O.D. Faugeras, Shape matching of two-dimensional objects, *IEEE Trans. Pattern Anal. Mach. Intell.* **PAMI-6**, pp. 137–155 (1984).
4. B. Bhanu, Shape matching of two-dimensional occluded objects, *Proc. 6th Int. Conf. on Pattern Recognition*, Munich, West Germany, pp. 742–744 (1982).
5. B. Bhanu, Representation and shape matching of 3-D objects, *IEEE Trans. Pattern Anal. Mach. Intell.* **PAMI-6**, pp. 340–351 (1984).
6. B. Bhanu, Automatic target recognition: state-of-the-art survey, *IEEE Trans. Aerospace Electron. Syst.* **AES-22**, pp. 364–379 (1986).
7. B. Bhanu and J.C. Ming, Recognition of 2-D occluded objects and their manipulation by PUMA 560 robot. Technical Report UUCS85-111, Department of Computer Science, University of Utah, August (1985).
8. B. Bhanu, A.S. Politopoulos and B.A. Parvin, Intelligent autocueing of tactical targets, *Architecture and Algorithms for Digital Image Processing*, A. Oosterlinck and P.E. Danielsson, eds, pp. 90–97. August (1983).

9. R.C. Bolles and R.A. Cain, Recognizing and locating partially visible objects: the local-feature-focus method. *Int. J. Robot. Res.* **1**, pp. 57–82 (1982).
10. W.K. Chow and J.K. Aggarwal, Computer analysis of planar curvilinear moving images, *IEEE Trans. Comput.* **C-26**, pp. 179–185, February (1977).
11. G.B. Coleman, *Image segmentation by clustering*, Technical Report USCIPI Report 750, Image Processing Institute, University of Southern California, Los Angeles, California (1977).
12. E. Diday, *Problems of Clustering and Recent Advances*, Technical Report 337, IRIA Laboria, Domaine de Voluceau, Rocquencourt, France, January (1979).
13. R. Dubes and A.K. Jain, Clustering techniques: the user's dilemma, *Pattern Recognition* **8**, pp. 247–260 (1976).
14. R.O. Duda and P.E. Hart, *Pattern Classification and Scene Analysis*. John Wiley (1973).
15. J.H. Friedman, J.L. Bentley and R.A. Finkel, An algorithm for finding best matches in logarithmic expected time, *ACM Trans. Math. Soft.* **3**(3), pp. 209–226 (1977).
16. A.K. Jain, Cluster analysis, *Handbook of Pattern Recognition and Image Processing*, K.S. Fu and T.Y. Young, eds. Academic Press, New York (1985).
17. M.W. Koch and R.L. Kashyap, A vision system to identify occluded industrial parts, *IEEE Int. Conf. on Robotics and Automation*, pp. 55–60 (1985).
18. T. Pavlidis and S. Horowitz, Segmentation of plane curves, *IEEE Trans. Comput.* **C-23**, pp. 860–870 (1974).
19. K.E. Price, Matching closed contours, *Proc. 7th Int. Conf. on Pattern Recognition*, pp. 990–992, July–August (1984). Also in Tech. Report 104, Intelligent Systems Group, University of Southern California, Los Angeles, October 19, pp. 29–37 (1983).
20. A. Rosenfeld and E. Johnston, Angle detection on digital curves, *IEEE Trans. Comput.* **C-22**, pp. 875–878, September (1973).
21. G. Stockman and J.C. Esteva, Use of geometrical constraints and clustering to determine 3-D object pose, *Proc. 7th Int. Conf. on Pattern Recognition*, July–August, pp. 742–744 (1984).
22. G. Stockman, S. Kopstein and S. Benett, Matching images to models for registration and object detection via clustering, *IEEE Trans. Pattern Anal. Mach. Intell.* **PAMI-4**, pp. 229–241 (1982).
23. J.L. Turney, T.N. Mudge and R.A. Volz, Recognizing partially occluded parts, *IEEE Trans. Pattern Anal. Mach. Intell.* **PAMI-7**, pp. 410–421 (1985).

About the Author — BIR BHANU received the S.M. and E.E. degrees in Electrical Engineering and Computer Science from the Massachusetts Institute of Technology, the Ph.D. degree in Electrical Engineering from the University of Southern California and the M.B.A. degree from the University of California Irvine.

At MIT Research Laboratory of Electronics he worked on the problem of computation of complex cepstrum. At USC he was at the Image Processing Institute where he worked on the problems of recognition of 2-D and 3-D objects, segmentation and motion analysis. In 1978 he worked with the computer science department of IBM research laboratory at San Jose in the image processing area. During the Winter of 1980 he was a research fellow at INRIA, the French National Research Institute in Computer Science where he worked on 3-D shape representation. From July 1981 to March 1984 he was an Engineering Specialist with the Aeronutronic Division of Ford Aerospace and Communications Corporation, Newport Beach, California. At Ford he worked on various aspects of the automatic target recognition problem and the industrial and tactical applications of image processing, pattern recognition and artificial intelligence. Currently he is working as an Assistant Professor of Computer Science at the University of Utah where he is engaged in teaching and research in the areas of computer vision, robotics, pattern recognition and artificial intelligence.

He is a member of IEEE, IEEE Computer Society, ACM, AAAI, Sigma XI, SPIE and Pattern Recognition Society.

About the Author — JOHN C. MING received the B.S. degree in Computer Science from the University of Utah in June, 1986. He is currently pursuing an M.S. degree in Computer Science at the University of Utah. His research interests are in computer vision, pattern recognition and robotics.

He is a student member of the Institute of Electrical and Electronics Engineers.