J. Indian Inst. Sci., May–June 1990, **70**, 197–212.

Two-dimensional object recognition using simulated annealing

N. K. SANCHETI*, Y. V. VENKATESH† AND Y. N. SRIKANT*
Indian Institute of Science, Bangalore 560 012, India.

Received on July 20, 1989; Revised on March 28, 1990.

Abstract

The following problem, which is intrinsic to computer vision, is considered in the paper: How to recognize, in a given scene (available in the form of a digital image), an object that is one among many possible items found in a certain knowledge (or model) base? The object in the scene could have been subjected to the following 'distortions': scale and orientation changes, and a shifting of position (with respect to the corresponding model in the knowledge-base). It is assumed that the given image is segmented and the objects are not subject to occlusion, *i.e.*, each object in the scene is isolated from the others, and can be located independent of the rest by examining the appropriate regions of the image. We present a new method for solving this problem. In contrast with the results of the literature, the objects' line segments, arcs and the like, *are not* stored in the knowledge-base. The price paid for this is that the matching strategy becomes more complicated. The novelty in the proposed method, however, is believed to be in the reformulation of the recognition problem as one of minimizing an error function, which, being non-convex, cannot be handled by standard optimization techniques. Therefore, simulated annealing is employed to find its global minimum. Nevertheless, this method *is not* recommended for real-time manipulation of robotic arms controlled by a vision system.

Key words: Object recognition, vision system, computer vision, simulated annealing.

1. Introduction

How to enable a robot to 'see' the environment in which it operates is a fundamental problem of robotics and automation. Technically, this is equivalent to the problem of object detection and recognition from a digital image. The object in the scene of the digital image is an element of a pre-specified set of objects constituting the knowledge (or model) base.

There are three basic problems in object detection and recognition:

1) How do we represent the *a-priori* knowledge (World Model) available about the objects under study, as obtained from their digital images?

* Department of Computer Science and Automation; † Computer Vision and Artificial Intelligence Laboratory, Department of Electrical Engineering.
† For correspondence.

2) Given an image of a new (unknown) scene, how can one extract a suitable description of the objects in it?
3) Using the World Model, is it possible to detect and recognize objects in the new scene?

When we are dealing with a robot of which the robotic arm is a part of the Vision System, we need to not only detect and recognize an object, but also compute its position and orientation. Moreover, the algorithm for recognition must also be scale invariant. This is useful if an automatic zooming of the camera were necessitated by variations in the sizes of the objects in a natural environment.

In brief, an object recognition system (ORS) is expected to 1) match an ideal template (or model) in the knowledge-base with its (possibly) distorted version as found in the image; and 2) extract the transformation from the model to the object in the scene.

In this paper, these requirements are duly considered. First, a knowledge-base for the objects is set up using certain invariant (or near-invariant) properties of the objects. Next, a procedure is outlined to match the model in the knowledge-base with the object in the scene. And, lastly, the transformation from the object in the scene to the model is computed.

2. Existing techniques

The problem of object recognition is inherently complex. It is therefore not surprising that most of the techniques found in the literature are combinatorially explosive when the contour of the object requires a number of primitives (like line segments, corners, and circular holes) for its representation. For a survey of earlier work, see, for instance, Pavlidis[1].

In what follows, we examine only a few of the recent papers, and comment on their technical content. Apparently, most of these papers are restricted in scope, and limited in application.

Mero[2] approximates the boundaries of objects by straight lines (obtained by applying the Hueckel operator) and arcs. After tracing the boundary of the object in the picture, the recognition algorithm first checks the boundary lines against the object models, and then performs a model-directed search for the other details of the object. Different objects are taught to the system, and recognized in several positions and sizes. In view of the need to store several views of the object (in different positions and sizes), it is not clear how efficient the matching strategy is, and how ambiguities are resolved.

It is known that the Fourier descriptors (FD)[3] can be used to describe the shape of a closed figure. While, theoretically, the operations of rotation, scaling, and moving the starting point are easily implemented in the Fourier domain, the procedures required in practice are known to be difficult[3]. In fact, Fourier descriptor[4,5] algorithms have apparently obscured the problem of representing a contour taken from a *sampled* image.

In Hu[6], two-dimensional moments which are *theoretically* invariant with respect to changes in size, orientation and position are used for recognition of objects. In practice,

however, these moments, as extracted from digital images, are *not* invariant. Further, it is not clear how ambiguities in the match operation are overcome.

Ballard[7] maps the object boundary space to the Hough transform space. Variations in the shape such as rotations, scale changes or figure-ground reversals correspond to transformations of this mapping. Ballard uses the Prewitt and Hueckel masks which model the local gray level changes as a ramp and a step, respectively. The mapping from the edge space to the Hough space (called the 'accumulator space' by Ballard) is such that instances of a certain shape in the given image scene produce local maxima in the accumulator space. This mapping is described as a table of edge-orientation reference-point correspondences, termed the R-table. Shapes are stored as canonical forms, and instances of shapes are detected by knowing the transformation from the canonical form to the instances. If this transformation is not known, the *all* plausible transformations must be tried. Moreover, it is not obvious how ambiguities in the match can be overcome.

Gupta and Srinath[8] use information in the closed boundary as a basis for shape recognition. The boundary is characterized by an ordered sequence that represents the Euclidean distance between the centroid and all boundary pixels in the discrete plane. They propose a translation, rotation and scale-invariant methodology for planar shapes having a closed boundary and *NO HOLES*. In their scheme, similarity between two contour sequences is established by a (i) nonlinear alignment scheme (that expands segments of a contour sequence in an optimum fashion), (ii) suitable choice of the starting point for alignment based on the knowledge of the shape's principal major axis, and (iii) by exploiting the circularity property of the contour sequence. The scaling factor for expansion is estimated by noting that if a function is scaled by s, then the second central moment of the function is scaled by s^2. The recognition algorithm is not applicable to objects with holes, and ambiguities in the matching procedure cannot be handled satisfactorily.

Bhanu and Faugeras[9] represent objects and scenes by relational graphs, and use a relaxation technique to search for subgraph isomorphisms. The algorithm is very complex, and its efficiency cannot be predicted when the knowledge-base is large.

Tsui and Chan[10] represent objects using primitive templates, and bring in contextual information explicitly for disambiguation during matching. The problem of recognition is reduced to one of sub-template matching of boundary images. Dynamic programming is then employed to find an optimal solution. In the course of searching for an optimal solution, weights are assigned to each sub-template to reflect their relative importance. Sub-templates which are distinctive among others are given more weight. The main difficulty seems to be in the creation of the library of sub-templates for the objects to be recognized.

A more recent paper is due to Ayache and Faugeras[11] wherein object shapes are represented by polygonal approximations of their borders, and the (stored) model description is by a set of linear segments. The given image is first subjected to an operation (like thresholding, Sobel mask convolution, low-pass filtering with different mask sizes, and zero-cross detection) which would yield a binary picture of the scene.

In their terminology[11], to generate a hypothesis is to predict the position of the model in the scene. This prediction is made by matching a 'privileged' segment in the model

description (MD) with a segment in the scene description (SD) by comparing local intrinsic features. Further, to evaluate a hypothesis is to take advantage of the predicted position of the model to identify additional segments between the two descriptions, and also refine the predicted position of the model (by a Kalman filter). Using this terminology, their basic idea is as follows: For each possible model, generate (predict) and evaluate a number of hypotheses. "Typically, a few hundred hypothese are generated and ranked on the basis of a local criterion of merit." Only the best first hypotheses are evaluated (typically a few tens), and the result of each evaluation is a final position estimate and a quality measure which accounts for the relative length of the identified segments. The matching ends when a sufficient number of hypotheses have been evaluated or when a very high-quality measure is reached. The hypothesis with the highest score is then reexamined before being validated or rejected. Even though this method of hypothesis testing is theoretically very attractive, its implementation is found to be extremely complex. As a result, no comments can be made about its efficiency.

Reitboeck and Altmann[12,13] employ log-polar transforms of templates and of the image which is assumed to contain *only the object* to be matched with the knowledge-base. This enables them to use cross-correlational processes which are rotation and size invariant. However, these are not invariant to the position of the centre of the log-polar transform. Therefore, it is desirable to achieve the invariance of a recognition algorithm with respect to the transformations, and at the same time retain the uniqueness of transformational states. To this end, Reitboeck and Altman found it necessary to develop a four-dimensional pattern representation where all the transformations are translated into shifts. And correlational processes must be enacted in a 4-D space. Given the pattern P, transform T, invariance property set I, and geometrical transform Tg, such that

$$T: P \rightarrow P_f, P_f \varepsilon I;$$

and

$$Tg: P \rightarrow Pg,$$

if T is invariant to Tg, then

$$T: Pg \rightarrow P_f'$$

and

$$P_f' \in I.$$

The restrictions are that the image should contain only one object without holes.

In the paper of Hummel and Wolfson[14], curves are represented by local features which are invariant with respect to rigid transformations. The features used are: points of sharp convexities, and deep convexities along the borders. These feature values are used to generate an attribute of a curve called its footprint, which enables us to efficiently index the appropriate local information to the object for recognition purposes. The method uses a hashing scheme indexed on the affine transformation and model type. There are practical difficulties in the extraction of corners and their attributes. As a consequence, the affine transformation is non-unique, and the matching results are ambiguous.

In the paper by Cass[15], objects are represented by their boundary contours, which in turn are expressed in terms of contour features. These features are characterized by a unique position and orientation. Transformation sampling is then used to determine the optimal model feature to image feature transformation. This is accomplished by sampling the space of possible transformations. All the matches of model and image features are considered, and the ranges of valid relative rotations and relative translations for each sample point in the transformation parameter space are computed. For each point in the parameter space, a measure of matching is computed. The parameter space points which give rise to optimal values of this measure are selected as possible parameter values for matches. However, no actual computational results are presented.

In the report of Banerjee *et al*[16], occluded objects are considered in the framework of a probabilistic relaxation scheme. An object is represented by a piece-wise linear approximation of its boundaries. To detect an object in the observed scene, a search is made in the space of functions from the set of line segments in the scene into the set of line segments of *all the models* in the knowledge-base. Each such function gives an interpretation of the scene. An error functional over the space of interpretations is minimized (using a probabilistic relaxation technique) to obtain minima which represent the possible matchings. It turns out that objects with circular arcs as contours or with many serrations cannot be handled satisfactorily by this method. This is especially the case when the knowledge-base of the model objects has to be quite large.

In contrast with the results of the literature, the main contributions of the present paper are: a) an optimization strategy (based on simulated annealing (SA)[17]) to arrive at the matching transformation; and b) a hierarchical method for minimizing the ambiguities in the match operation. The main disadvantages, however, are: a) occluded objects cannnot be analysed with the present knowledge-base; and b) the time taken for matching is enormous, thereby precluding its application to the real-time control of a robotic arm, *unless*, of course, *a parallel version of the algorithm can be run on a multi-processor machine.*

Remark 2.1. Even though there are quite a few papers on the application of SA to image processing and vision (see, for instance, Carnevali *et al*[18] and Smith *et al*[19]), it should be added that *the present paper appears to be the first to apply simulated annealing (SA) to object recognition.*

3. Proposed method

We focus on model matching, and assume that the earlier stages (namely, pre-processing and segmentation) have been accomplished by some means. Unlike most of the results of the literature, we do not use any algorithm for approximating the boundaries of objects. Objects can consist of (i) both polygonal and curved segments, (ii) many concavities, and contain various degrees of detailed features. We do not match the object with each model in the library in a linear fashion. On the contrary, we store enough shape information in order to uniquely define subsets of likely candidates after extracting the scene features.

Table I

Object	Area	PERM	SHPFAC	CGX	CGY	INVFAC1	INVFAC2
Model parameters (Typical)							
Plier	868	231	0.12754	51.409	82.85	2.9057	8.44
Spanner	1153	245	0.13859	60.34	74.77	2.229	4.970
Screw	691	177	0.14851	47.50	74.21	5.716	32.678
Object 1	2450	248	0.19958	62.65	65.81	0.54	0.292
Object 2	2863	243	0.22019	66.48	63.53	0.498	0.248
Scene objects' parameters (Typical)							
Scene 1	2715	272	0.19156	58.31	61.63	0.527	0.278
Scene 2	4479	329	0.20342	63.19	60.01	0.506	0.256
Scene 3	1167	270	0.12652	57.54	40.23	2.466	6.0814
Scene 4	710	211	0.12628	42.49	69.23	6.078	36.943
Scene 5	866	272	0.10819	62.77	96.69	2.412	5.816

As far as the library (or the knowledge-base) is concerned, information about the shape of known objects is stored in a hierarchical way so as to minimize search in the course of matching of shape representations produced from the scene. A typical set of values used in this step is shown in Table I, corresponding to some of the objects shown in figs 1 and 2 (for details, see below).

A hierarchical technique is used for matching, with the possible match location starting at a low resolution. Two sets of these images are required: one for the scene and the other for the model. A threshold sequence and decision rules are established to guide the search from a low-resolution level to the next higher resolution level.

At the end of the match, it is required that the orientation, translation, and scaling factor of the identified object should be available as accurately as possible.

We now give a brief description of the various steps used in the proposed algorithm. The modifications needed to handle occluded objects are outlined at the end of the section.

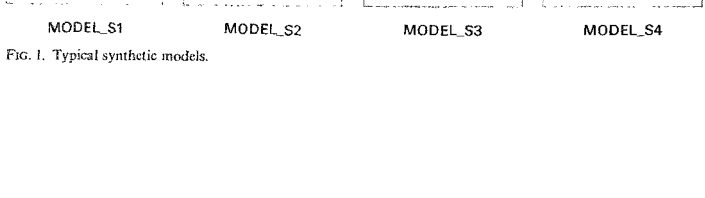

FIG. 1. Typical synthetic models.

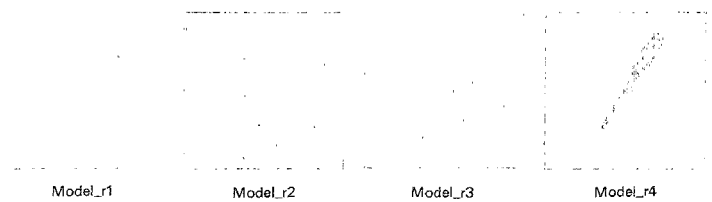

FIG. 2. Typical real models: Models $r1$ to $r3$ are shown as gray level images (halftone), and Model $r4$ as a thresholded image.

3.1. *Essential details of the proposed method*

The first step (Step 1) consists of the creation of the knowledge- (or model-) base (MB) with a pre-specified number, N, of objects. Each object, with a label, is supposed to have been given in the form of a binary image. Find its area (A) and perimeter (P) by any standard procedure. Find its (i) shape factor, S_o, defined by $S_o = SQRT(A)/P$, (ii) its centroid (X_c, Y_c), and (iii) its first and second invariants, τ_1 and τ_2 which are defined as follows:

Let $f(x, y)$ denote the digital image function describing the object (in the model set or in the scene), and let Σ denote summation with respect to both x and y coordinates.

$$m_{00} = \Sigma f(x, y); m_{10} = \Sigma x^* f(x, y); m_{01} = \Sigma y^* f(x, y);$$

$$X_c = m_{10}/m_{00};\ Y_c = m_{01}/m_{00};$$

$$\beta_{11} = \Sigma(x - X_c)^*(y - Y_c)^* f(x, y);$$

$$\beta_{20} = \Sigma(x - X_c)^{2*} f(x, y); \beta_{02} = \Sigma(y - Y_c)^{2*} f(x, y);$$

$$\tau_1 = (\beta_{02} + \beta_{20})/m_{00};$$

and

$$\tau_2 = ((\beta_{02} - \beta_{20})^2/m_{00}{}^2) + 4^*(\beta_{11}/m_{00})^2.$$

The orientation of the model/object can be obtained by computing the following angle:

$$\varphi = (1/2)\tan^{-1}(2^*\tau_{11}/(\tau_{20} - \tau_{02}).$$

This can be used to minimize the search for the correct orientation in the optimization procedure. (See Hu[5] and Weiss[20] for an explanation of these parameters.)

For each model object, store its area, perimeter, shape factor, centroid, first and second invariants, in the library (L) of the model base. The library is ordered first in terms of the shape factor, and next in terms of the first and second invariants. This is level #0 in the feature space hierarchy.

For a resolution factor K (integer) of the images of the models, the feature space parameters of library L are transformed to constitute level #1 of the feature space hierarchy. (This level is required in Step 7 for minimizing the error measure of match

between the low-resolution object in the transformed image and the models in the library.) This concludes Step 1.

In Step 2, the given image of the scene containing multiple objects is pre-processed to yield a segmented image in which the objects are assumed to be distinct, *i.e.*, in the present version of the algorithm, the objects are supposed to be *not occluded.*

In Step 3, the perimeter and area of the part of the segmented image (where an object is supposed to exist) are calculated by any standard procedure. Then the shape factor, S_σ, and the two invariant factors, β_1 and β_2, of the object under study are obtained. S_σ is first compared with the shape factors in the library, L. Those objects in the library, not falling within a pre-specified error tolerance for shape factor matching, are discarded.

Let $N1$ be the number of objects in the library accepted for further matching operation. Next, the first and second invariants of the object are compared with the sub-library of $N1$ models, thereby reducing the accepted number of matches to $N1r$. In the same step, the areas of the $N1r$ models (in the library) are compared with the area of the object in the image to estimate corresponding scale factors. Further, the centroids of the model objects and those of the object in the scene are compared in order to estimate corresponding shift parameters (X_{sh}, Y_{sh}) (These values are used in Step 6 to reduce the search space in the minimization procedure. Note that these estimates are at level #0 of the hierarchy in the feature space.).

In Step 4, the image of the object is reduced by a factor K commensurate with the details found in the image, *i.e.*, finer the details in the image, smaller is the reduction factor. With $K > 1$, we are now working at level #1 of the feature space hierarchy. The object in the image of a reduced resolution is then subject to the following linear transformation to transform it to a new coordinate system.

With (x, y) denoting the coordinate system of the original image, let (x_t, y_t) represent the coordinates of the transformed image, α the scale factor, Θ the rotation, (x_{co}, y_{co}) the coordinates of the centroid of the object in the original image, (x_{ct}, y_{ct}) those of the object in the transformed image, and (X_{sh}, Y_{sh}) the shift in the (x, y) coordinate system. The transformation required for the next stage of matching is given by

$$(x_t - x_{ct}) = \alpha \times ((x - x_c) \times \cos(\Theta) + (y - y_c) \times \sin(\Theta)) + X_{sh};$$

$$(y_t - y_{ct}) = \alpha \times ((y - y_c) \times \cos(\Theta) - (x - x_c) \times \sin(\Theta)) + Y_{sh}.$$

(end of Step 4).

Step 5 requires computation of the difference between the two images, the first (f_{ot}) contains the object in the transformed image, and the second $(f^{(\text{model})})$ one contains one of the $N1$ models chosen from the library. This difference is computed as the error function,

$$E(PR) = \Sigma_{i,j} |f_{ot}(i,j) - f^{(\text{model})}(i,j)|,$$

with the parameter set

$$PR = \{\alpha, x_{sh}, y_{sh}, \Theta, \text{model}\},$$

where the difference between transformed object and the model is explicitly shown as a function of the parameter set PR in which the elements are, respectively, the scale factor, coordinate shift, rotation and the model chosen (end of Step 5).

In Step 6, E is minimized with respect to the parameter set PR. Note that E may have multiple local minima. As we are looking for a global minimum, we cannot use the standard minimization procedures for minimizing E. See Appendix I for a brief introduction to simulated annealing as applied to the present problem.

Let us assume that, at this stage, $N2$ objects from the model library (L) have been selected as possible matches with the object in the scene. Note the corresponding values of the parameter set PR (end of Step 6).

In Step 7, the original image of the object is used and subjected to the same operations as in Step 6. The initial values of the parameter set PR, as obtained in Step 6, serve as the starting points for the minimization strategy. Observe that the search space for the simulated annealing[17] procedure is reduced by a factor $N2/N$, in addition to the fact that the search space is constrained to the vicinity of parameter set PR, as obtained in Step 6. Acceptance of the match is based on a pre-specified threshold. The lowest value of the error measure is chosen as the one corresponding to the best match, for which the transformation parameter set is obtained from Step 6 (at the highest resolution).

Remark 3.1. For dealing with occluded images, the shape characteristics of Table I are no longer adequate. However, it is worth noting here that some of the most prominent features like the corners and the angles subtended at those points, along with the lengths of the sides giving rise to those corners are needed to resolve the problem of recognizing occluded objects. From the given scene, corners, the subtended angles at those points and the lengths of the sides constituting those corners are extracted. A suitable search strategy is employed to arrive at partial matches of the vertices and angles with the library models. The output of this procedure gives the (i) scale factor; (ii) shift in the vertices for alignment; and (iii) (affine) transformation matrix, for each of the object models in the library which make up the occluded object. The remaining steps of the present matching algorithm (suitably modified) are applied to minimize the error between the occluded object (in the scene) and the model objects as transformed, simultaneously, by the affine transformations (corresponding to each possible match) (end of Remark 3.1).

We now summarize the above steps as the following recognition algorithm for un-occluded objects.

PROGRAM OBJECT_RECOGNITION (Input = models of objects, segmented images of the scenes under consideration; Output = labelling of the objects in the scenes, the locations, and orientation of these objects);

Stage I: MODELLING

for each OBJECT to be modelled do
Begin
Find its area (A_m), perimeter (P_m), centre of gravity (x_{cm}, y_{cm}), shape factor (S_σ),

invariant factors 1 and 2 (INV_{1m}, INV_{2m}), and orientation (ORN_m).
Sort the model objects with respect, first, to the S_{om}, next, to INV_{1m} and, then, to INV_{2m}.
The LIBRARY listing now contains

($MODELNAME$, S_{oM}, INV_{1m}, INV_{2m}, (x_{cm}, y_{cm}), A_m, P_m, *and* ORN_m)

end (of MODELLING).

Stage II: RECOGNITION

1. After segmentation, for each object in the scene, calculate its area (A_o), perimeter (P_o), shape factor (S_{go}), centre of gravity (x_{co}, y_{co}), invariant factor 1 (INV_{1o}), invariant factor 2 (INV_{2o}) and orientation (ORN_o).
2. Level #0
   Set tolerance limits for matching.
   Search for the matching shape factors (with a pre-specified tolerance), and check the corresponding invariant factors 1 and 2. List the objects chosen for the next level analysis, and calculate the corresponding scale factors and the shifts in the centre of gravity with respect to the object under consideration.
3. Level #1
   Map the object to a lower resolution (and calculate the area, and centre of gravity).
   For each model chosen in Level #0, repeat the process (of area and centroid computation).
   Corresponding to each match, compute the initial values for scale factor, and translation.
   Set tolerance limits for termination of the optimization procedure.
   Invoke the PROCEDURE SIMULATED ANNEALING (Appendix I) to optimize the match with respect to the model number, scale factor, shift, and rotation.
   If the minimal error is less than a pre-specified tolerance, accept the match for the next level of matching.
4. Level #2
   Begin
   with the parameters obtained from Level #1 (and suitably scaled for the high resolution), invoke the PROCEDURE SIMULATED ANNEALING to optimize with respect to scale factor, shift and rotation, and pick the model(s) with the lowest error measures (in comparison with the pre-specified tolerance) as the one(s) matching the object in the scene. Note the values of the corresponding transformation parameters.
   end (of Level #2).

END OF RECOGNITION.

END OF PROGRAM OBJECT_RECOGNITION.

4. Experimental results

A considerable number of experiments have been conducted using both synthetic (generated by a procedure similar to the creation of random dot images) and natural

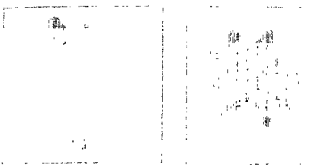

FIG. 3. Objects (synthetic) to be recognized.

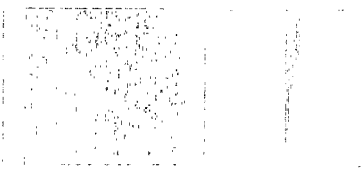

FIG. 4. Objects (real) to be recognized: Object *r*1 is shown as a gray level image (halftone), and Object *r*2 as a thresholded image.

objects. Based on these studies, we have arrived at suitable tolerance limits for accepting the possible matches. In the annealing strategy, many cooling schedules have been tried. Typical values are: $T_{start} = 500$; and cooling rate, $r = 0.995$.

Obviously, given a different set of models and objects in a scene, these experiments are to be repeated with a view to establishing the corresponding tolerance limits and cooling schedules. No general guidelines can be prescribed.

However, the results obtained indicate that the new method of matching works well. But, as far as real-time recognition is concerned, the proposed method *cannot be recommended*, and this remark is applicable to all the methods employing simulated annealing[17], unless, of course, a parallel version of the algorithm can be created to run on multi-processor machines.

Figures 1 and 2 give a typical set of synthetic and natural objects used in the study. The library contains about 35 objects of various types. For the first-level analysis, typical contents of the library are shown in the upper half of Table I. (Orientation information is not included here.) In figs 3 and 4 are given the scaled, rotated and shifted versions of some objects (as obtained by a suitable segmentation of the original scene). The corresponding parameters for a typical set of transformed objects are given in the lower half of Table I. Some intermediate and final results of the matching (in terms of the difference images created) are shown in figs 5 and 6 (which do not correspond to figs 3 and 4, the latter being given merely to illustrate the nature of model-to-object transformation). The effect of the

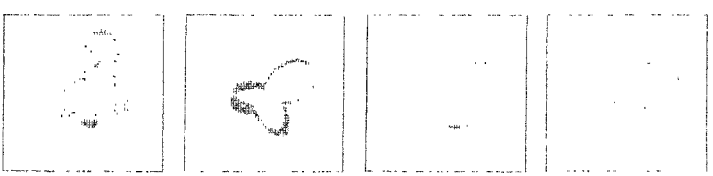

FIG. 5. Results of the recognition algorithm (synthetic objects). Matching errors shown: begin, intermediate (1 and 2) and final.

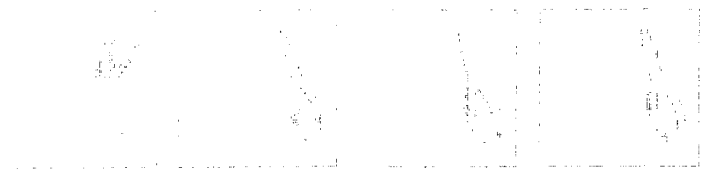

FIG. 6. Results of the recognition algorithm (real objects). Matching errors shown: begin, intermediate (1 and 2) and final.

incorrect match is reported as a large error (quantified by the larger whiteness of the difference image) in the optimization procedure, which disqualifies the attempted match. The correct match is the one with the smallest error.

Remark 4.1. Regarding the computational requirements, the following points are to be noted:

(i) The creation of the library is done off-line. In view of the problems related to the vagaries of digitization, it is advisable to store some rotated and scaled versions of each model object. This would minimize the possibility of wrong matches.

(ii) Level 1 of the recognition operation can be accomplished quite fast in view of the sorted parameter table. On a *PC–AT*, this takes a couple of seconds for a library containing about 35 model parameter sets. Reduction in the number of possible matches is possible by incorporating angles at suitably chosen vertices ('interest points', like corners). This reduces the dimensionality of search in Levels #4 and 5 (optimization).

(iii) Evidently, the computationally intensive part of the recognition procedure is optimization using simulated annealing (in the 4-/5-dimensional parameter space). For an image (of the object) of size 128 by 128, on a *PC–AT* (80286), the recognition part of the algorithm takes about 4 hours. In view of the non-convex nature of the error functional, it is unlikely that any better strategy than what has been attempted can be recommended in a general context.

5. Conclusions

Simulated annealing has been employed to optimize the matching of an isolated object in a scene with its model in the library which is so created as to contain some appropriate invariant features of all the model objects under consideration, arranged in a suitable hierarchy.

Unlike most of the results of the literature, the proposed method does not depend upon any approximation of the object's boundaries by straight lines and arcs of curves. Even though the complexity of matching has been reduced by dealing with the invariant features

of the object, and by adopting a multiresolution strategy for optimization of the match parameters, real-time operation cannot be achieved. This is due to the very nature of the optimization scheme used, and is in marked contrast with the results reported[21] on robotic arm control using vision (dealing with a class of objects which can be represented by a limited number of primitives) where real-time recognition has been demonstrated to be possible.

Acknowledgements

The authors wish to thank Dr K. R. Ramakrishnan and Ms Shashikala Rao for some helpful discussions. Special credit is due to Mr K. Ramani for the excellent software environment created for displaying images. Thanks are also due to the anonymous referees for their comments on an earlier version of the paper. One of the referees brought references 18 and 19 to the authors' notice, as applications of simulated annealing to image analysis.

References

1. PAVLIDIS, T. — A review of algorithms for shape analysis, *Computer Graphics Image Processing*, 1978, **7**, 243–258.

2. MERO, L. — An algorithm for scale- and rotation-invariant recognition of two-dimensional objects, *Computer Graphics Image Processing*, 1981, **15**, 279–287.

3. WALLACE, T. P. AND WINTZ, P. A. — An efficient three-dimensional aircraft recognition algorithm using normalized Fourier descriptors, *Computer Graphics Image Processing*, 1980, **13**, 99–126.

4. PERSOON, E. AND FU, K. S. — Shape discrimination using Fourier descriptors, *IEEE Trans.*, 1977, **SMC-7**, 170–179.

5. PAVLIDIS, T. — Algorithms for shape analysis of contours and waveforms, *IEEE Trans.*, 1980, **PAMI-2**, 301–312.

6. HU, M.-K. — Visual pattern recognition by moment invariants, *IRE Trans.*, 1962, **IT-8**, 179–187.

7. BALLARD, D. H. — Generalizing the Hough transform to detect arbitrary shapes, *Pattern Recognition*, 1981, **13**, 111–122.

8. GUPTA, L. AND SRINATH, M. D. — Contour sequence moments for the classification of closed planar shapes, *Pattern Recognition*, 1987, **20**, 267–272.

9. BHANU, R. S. AND FAUGERAS, O. D. — Shape matching of two-dimensional objects, *IEEE Trans.*, 1984, **PAMI-6**, 137–156.

10. TSUI, H. T. AND CHAN, M. H. — Recognition of partially occluded two-dimensional objects, *IEE Proc., Pt E*, 1987, **134**, 9–16.

11. AYACHE, N. AND FAUGERAS, O. D. — HYPER: A new approach for the recognition and positioning of two-dimensional objects, *IEEE Trans.*, 1986, **PAMI-8**, 44–54.

12. REITBOECK, H. J. P. AND ALTMANN, J.	A model for size- and rotation invariant pattern processing in the visual system, *Biol. Cybernetics*, 1984, **51**, 113–121.
13. ALTMANN, J. AND REITBOECK, H. J. P.	A fast correlation method for scale- and translation-invariant pattern recognition, *IEEE Trans.*, 1984, **PAMI-6**, 46–57.
14. HUMMEL, R. AND WOLFSON, H.	Affine invariant matching, *Proc. Image Understanding Workshop*, DARPA, pp. 351–364, April 1988.
15. CASS, T. A.	Robust parallel computation of 2D model-based recognition, *Proc. Image Understanding Workshop*, DARPA, pp. 640–650, April 1988.
16. BANERJEE, S., SASTRY, P. S. AND RAMAKRISHNAN, K. R.	*2-D object recognition through a stochastic search in the space of possible interpretations*, Technical Report, Electrical Engineering Department, Indian Institute of Science, Bangalore, India, May 1988.
17. KIRKPATRICK, S., GELATT, C. D. AND VECHHI, M. P.	Optimization by simulated annealing, *Science*, 1983, **220**, 671–680.
18. CARNEVALI, P. I., COLETTI, L. AND PATERNELLO, S.	Image processing by simulated annealing, *IBM J. Res. Dev.*, 1985, **29**, 569–579.
19. SMITH, W. E., BARRET, H. H. AND PAXMAN, R. G.	Reconstruction of objects from coded images by simulated annealing, *Opt. Letters*, 1983, **8**, 199–201.
20. WEISS, I.	Projective invariants of shapes, *Proc. Image Understanding Workshop*, DARPA, pp. 1125–1134, April 1988.
21. USHA RAJESWARI, RAMAKRISHNAN, K. R. AND VENKATESH, Y. V.	*On robotic manipulation using vision*, Technical Report, Electrical Engineering Department, Indian Institute of Science, Bangalore, India, June 1989 (to be published).

Appendix I

Simulated annealing

Simulated annealing, introduced by Kirkpatrick and others, is a strategy for combinatorial optimization problems, such as minimizing a function of many variables. On the other hand, annealing, as is well known, is a manufacturing process in which a molten metal is cooled extremely slowly to produce a stress-free solid. In mathematical terms, this means achieving a configuration of atoms with energy near or at the global minimum.

As applied to non-physical optimization problems, we employ an analogous set of 'controlled cooling operations', in order to achieve the desired solution. For instance, in the present problem of object recognition, we seek to find a 'configuration' of parameters,

$$PR = \{\alpha, x_{sh}, y_{sh}, \Theta, \text{model}\},$$

that minimizes the error function,

$$E(PR) = \Sigma_{i,j} |f_{or}(i,j) - f^{(\text{model})}(i,j)|,$$

where the various symbols carry the same meaning as in Sec. 3(a), Step 5. Note that the error function may have many local minima, but its global minimum is zero, corresponding to a perfect match.

We are interested in iterative improvement strategies, which attempt to perturb some existing, suboptimal solution (for the parameters in the set PR) in the direction of a better, lower-cost solution. However, this strategy gets easily trapped in local minima—solutions that look good in some small neighbourhood of the error function, are not necessarily the global minimum. One scheme to overcome this limitation is simply to try many initial configurations, improve each, and use the best answer found. However, there is no guarantee of finding a good solution.

Simulated annealing offers a strategy similar to the iterative improvement, with a major difference: annealing allows perturbations to move uphill, on the 'surface' of the error function, in a controlled fashion. Consequently, it is possible to jump out of local minima and potentially fall into a more promising downhill path. The control is exercised through the parameter T (pseudo-temperature) in the annealing algorithm.

The main idea, then, is to propose some perturbation in the parameter set PR, and evaluate the change, δE, in the error function. If $\delta E < 0$, the new configuration has low energy, and is accepted as the start point for the next 'move' or perturbation. However, if $\delta E > 0$, the move may still be permitted under certain conditions. The new, higher energy configuration is acceptable, as moderated by the *current temperature*. The probability of accepting a move which results in $\delta E > 0$, at temperature T, is given by

$$\text{Prob}[\text{accept}] = \exp(-\delta E/T).$$

In practice, however, this probabilistic acceptance is achieved by generating a uniform random number, U, in $[0, 1]$, and comparing it against Prob[accept]. *Only if* Prob[accept] $> U$ is the move accepted. Thus, it is likely that very probable moves may be rejected, and very improbable moves may be accepted—at least occasionally. By successively lowering the temperature, and running the algorithm, we can simulate the material coming to equilibrium at each newly reduced temperature.

The cooling schedule, prescribed, for instance, by,

$$T_{n+1} = r^* T_n, r < 1,$$

(where n is the iteration number of a chosen set of moves, say, 100), is a sequence of decreasing temperatures to moderate the acceptance of uphill moves over the course of the solution. Initially, this effective temperature parameter is high enough to permit an aggressive random search of the parameter space PR. A typical stopping criterion is to terminate annealing when the error reduction over three/four successive temperatures is sufficiently small, *e.g.*, less than 1%.

This part of the algorithm as actually implemented is given below:

```
PROCEDURE SIMULATED_ANNEALING;
    var α, x_sh, y_sh, Θ, model; f_ot, f^(model);
    {Recall that E(PR) = Σ_{i,j} |f_ot(i,j) − f^(model)(i,j)|}
```

```
begin
        set I = number of iterations to attempt;
        T = current temperature; r = annealing rate;
{E at temperature T is denoted by E^(T)}
  Repeat
    for m = 1 to I do
      begin
        perturb PR = {α, x_sh, y_sh, Θ, model}
        evaluate δE^(T) = δ(Σ_{i,j} |f_ot(i,j) − f^(model)(i,j)|)
        if δE^(T) < 0 then
          accept perturbed PR and update the configuration
        else
        begin
          accept perturbed PR with probability = exp(−δE^(T)/T) and update the
        configuration if accepted
        end
  end
        T := r* T
  until
        E^(r*r*r*T), E^(r*r*T), E^(r*T), E^(T) do not differ by more than 1%
end {SIMULATED_ANNEALING PROCEDURE}.
```