Pergamon

Pattern Recognition, Vol. 30, No. 2, pp. 245–252, 1997

0031–3203/97 $17.00+.00

PII:S-0031-3203(96)00080-5

A MACHINE LEARNING APPROACH FOR ACQUIRING DESCRIPTIVE CLASSIFICATION RULES OF SHAPE CONTOURS[1]

JUI-CHI HSU and SHU-YUEN HWANG*

Department of Computer Science and Information Engineering, National Chiao-Tung University, Hsinchu, Taiwan, R.O.C.

(*Received* 28 *September* 1995)

Abstract—We devise a method to generate descriptive classification rules of shape contours by using inductive learning. The classification rules are represented in the form of logic programs. We first transform input objects from pixel representation into predicate representation. The transformation consists of preprocessing, feature extraction and symbolic transformation. We then use FOIL which is an indictive logic programming system to produce classification rules. Experiments on two sets of data were performed to justify our proposed method. Copyright © 1997 Pattern Recognition Society. Published by Elsevier Science Ltd.

Shape representation Classification Machine learning FOIL
Inductive logic programming

1. INTRODUCTION

Shapes are important information for acquiring the notion of objects. Much research that utilizes shape information to perform tasks in computer vision has been done. For example, Hogg expressed that shapes can evoke a wide range of visual concepts relevant to describing object geometric properties.[1] Jovanović[2] used shape information to classify model tanks, airplanes, helicopters, trucks and armored cars. Katzir *et al.*[3] utilized shape information to recognize spoons. Ueda and Suzuki[4] made use of shape information to classify sedan, hatchback and wagon cars. Ansari[5] adopted a landmark-based approach using shape information to recognize wrenches, needle-nose pliers, wire cutters, and wire strippers. Medioni *et al.*[6,7] designed a system to find features hidden in shapes for pattern recognition.

One common requirement in these works, as in most other object recognition systems, is to construct classification rules or models that can be used to classify or compare with an input object. Without the help of domain experts, machine learning techniques have to be used to achieve this goal. In addition to works mentioned above, Connell and Brady[8] designed a system that combines recognition and learning. The system utilizes shape information and generates a semantic network representation for an object. Hättich and Wandres[9] devised a system to construct 2D models based on "learning by showing". Jovanović[2] proposed a learning algorithm based on *l*-classifier to generate and modify the structure of a pattern class. Fichera *et al.*[10] used fuzzy logic and the INDUCE system to learn the organs of human brains. Ueda and Suzuki[4] retained perceptually relevant features in shapes to learn shape models. These results show that machine learning techniques really help the task of recognition patterns well.

However, one drawback in existing research is that the characteristics extracted and learned by these systems are not easily understandable to humans. This is also the gap that exists between machine vision and human perception. Consider how people recognize simple objects such as screws or spoons. Features in these objects perceived by human are something like that screws always have a hat and a rectangle, and spoons have a long ladle. These characteristics can help us easily to determine what an object is, but will be missed if method in most previous research were to be used. For this reason, we devise a method that discovers knowledge from shape contours by using inductive learning techniques. The discovered knowledge is symbolic classification rules of two classes and represented in the form of logic programs. Our goal is to have the classification rules as descriptive as possible.

The structure of this paper is as follows. Section 2 describes the transformation of shape contours from pixel level to symbolic representation. Section 3 describes the learning procedure. Experimental results are shown in Section 4. Section 5 is the conclusion.

2. FROM PIXELS TO PREDICATES

This section describes how to transform the representation of input objects from pixel level into predicate level. We skip the image acquisition and boundary extraction steps, and assume that the input are in the

* Author to whom correspondence should be addressed. Tel.: 886-35-781366; fax: 886-35-724176; e-mail: syhwang@cise.nctu.edu.tw.

[1]Research supported by National Science Council, R.O.C., under grant number NSC 84-0408-E-009-014.

form of boundary pixels, with possible noises. The transformation consists of three steps: processing, feature extraction, and symbolic transformation.

2.1. Preprocessing

The goal of preprocessing is to eliminate the effect of noises and geometric transformations, such as translation and rotation, on the results of later processings. In our system, the step of preprocessing includes smoothing and the Hotelling transform.

The goal of smoothing is to reduce the influence of noises on contours. We employed a modified Gaussian kernel to smooth a contour.[11] The goal of the Hotelling transform[12] is to align all objects along their principal axes, so that the rotation of an object will not affect later processings. The X and Y axes in the experimental data (see Figs 4–10) are the principal axes found by the Hotelling transform.

2.2. Feature extraction

Feature extraction plays a key role in our system because it decides the information that an object can provide to the learning algorithm, and thus decides the classification rules. The first step of feature extraction is to divide the contour of a shape into segments. We employ a modified *k-curvature algorithm*[13] in this process. The algorithm can divide a contour by finding some segmentation points based on local curvature of segments. An "*" mark represents a segmentation point in the experimental data (see Figs 4–10).

Once the segments of a shape contour have been obtained, properties of segments are computed. The properties need to include all important characteristics of the shape, based on the segmented components.[14] The output of this step includes three sets S^i for each segment, and several A^js and P^ks for entire contour, as described below. We will omit the superscripts of these sets.

S is the set that contains the information of individual segments. $S = \langle id, loc, ht, len, con, sym \rangle$. The components of S are defined below:

- *id* is the identification of the segment.
- *loc* indicates the location of the segment. We divide an image into four quadrants based on the axes found in the Hotelling transform. The assign numbers 0–7 indicate the different start points and end points of the segment.
- *ht* is the height of the segment, which the largest distance from the segment to the straight line formed by connecting the start point and the end point of the segment.
- *len* is the length of the segment.
- *con* is the number used to indicate the segment is convex or concave or straight.
- *sym* represents whether the segment is symmetric.

A is a set which represents the interrelations between two adjacent segments. $A = \langle id, idc, \theta \rangle$, as described below:

- *id* is the identification number of a segment.
- *idc* is the identification number of the segment that is clockwise adjacent to *id*.
- θ is the angle between segments *id* and *idc*.

P is the set which represents the possible parallelity of some segments, $P = \langle id_1, id_2, \phi \rangle$, as described below

- id_1 is the identification number of a certain segment.
- id_2 is the identification number of another segment.
- ϕ is the angle between segments id_1 and id_2, and is close to 0° or 180°.

The set P is *ad hoc* compared to general information provided by S and A, which are usually referred to in literature of shape representation.

2.3. Symbolic transformation

Eight types of predicates are used to represent information extracted. Type 1 links a segment with a contour. Type 2–6 correspond to S, type 6 is related to A, and type 8 is for P.

1. `Have (i,n)`: This means that segment `n` belongs to contour `i`.
2. `Token1 (id, loc)`: The first two elements of A are extracted to become the parameters of the predicate `Token1`. This predicate represents the location of segment `id`.

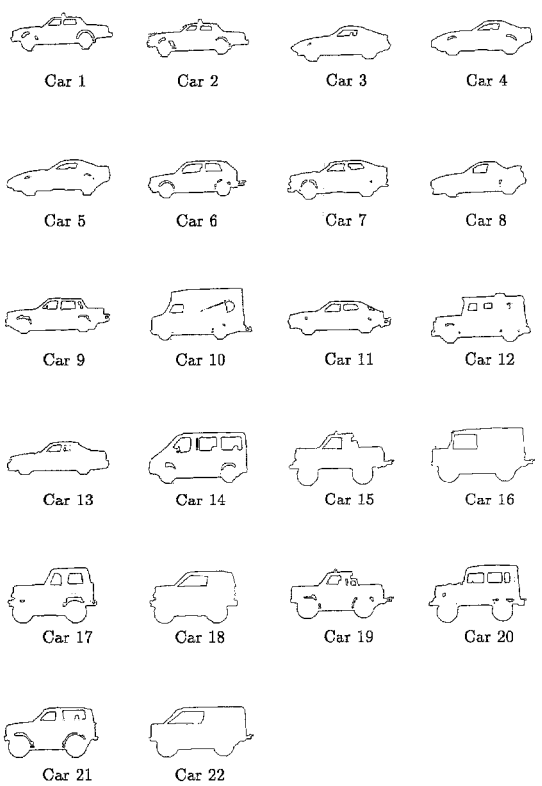


Fig. 1. Contour data of experiment 1.

3. `Token2 (id, HT)`: `Token2` is used to show whether segment `id` is sharp or smooth, the parameter HT is a digitized result of *ht*. We assign 13 to HT if the corresponding `ht` is greater than the medium of the set of all *hts* in *S*, otherwise 12 is assigned.
4. `Token3 (id, LEN)`: `Token3` can determine whether segment `id` is narrow or wide. Similar to HT, the parameter LEN is set to 9 if the corresponding *len* is greater than the medium of the set of all *lens* in *S*, otherwise it is set to 8.
5. `Token4 (id, CON)`: `Token4` can show what shape of the segment `id` is. CON is 15 if segment `id` is concave, 16 if `id` is nearly a straight line, and 17 if segment `id` is convex.
6. `Token5 (id, SYM)`: `Token5` can identify whether segment `id` is symmetric or not. If segment `id` is not symmetric, SYM is equal to 32 else SYM is equal to 33.
7. `Adj(n1,n2,agl)`: This predicate is used to represent the relationships of adjacent segments. The parameters `n1` and `n2` are formed by concatenating the digit *i* with *id* and *idc*, respectively. Whereas the third parameter `agl` is a symbolic term that is formed by first clustering all θs into 64 groups and then for each group assigning a symbolic term.
8. `Par(p1,p2,pal)`: This predicate is a symbolic counterpart of *P*. The parameters `p1` and `p2` are formed by concatenating the digit *i* with id_1 and id_2, respectively. The third parameter is set to "`a`" if $\phi \leq \epsilon$, and "`-a`" if $\phi \geq 180 - \epsilon$.

Because we use a complete description on a shape contour, it is not surprising that a shape contour may result in a long list of predicates. For example, Fig. 2 is a symbolic representation of a car 2 in Fig. 1.

3. LEARNING SYMBOLIC CLASSIFICATION RULES

As reviewed in Section 1, machine techniques have been used in learning various aspects of image shapes. A theoretical analysis on learning visual concepts shows the high computational complexity in learning visual concepts in pixel level.[15] This is another reason that we intend to perform learning on symbolic level (e.g. see Fig. 2).

```
Have(2a,2a0), Have(2a,2a1), Have(2a,2a2), Have(2a,2a3), Have(2a,2a4),
Have(2a,2a5), Have(2a,2a6), Have(2a,2a7), Have(2a,2a8), Have(2a,2a9),
Have(2a,2a10), Have(2a,2a11), Have(2a,2a12), Have(2a,2a13), Have(2a,2a14),
Have(2a,2a15), Have(2a,2a16), Have(2a,2a17),
Token1(2a0,1), Token1(2a1,1), Token1(2a2,1), Token1(2a3,0), Token1(2a4,7),
Token1(2a5,7), Token1(2a6,7), Token1(2a7,7), Token1(2a8,6), Token1(2a9,5),
Token1(2a10,5), Token1(2a11,5), Token1(2a12,5), Token1(2a13,4),
Token1(2a14,3),Token1(2a15,3), Token1(2a16,3), Token1(2a17,2),
Token2(2a0,12), Token2(2a1,12), Token2(2a2,13), Token2(2a3,13),
Token2(2a4,13), Token2(2a5,13), Token2(2a6,13), Token2(2a7,12),
Token2(2a8,12), Token2(2a9,12), Token2(2a10,12), Token2(2a11,13),
Token2(2a12,12), Token2(2a13,13), Token2(2a14,13), Token2(2a15,12),
Token2(2a16,12), Token2(2a17,12),
Token3(2a0,8), Token3(2a1,8), Token3(2a2,9), Token3(2a3,9), Token3(2a4,9),
Token3(2a5,9), Token3(2a6,9), Token3(2a7,8), Token3(2a8,8), Token3(2a9,8),
Token3(2a10,9), Token3(2a11,9), Token3(2a12,8), Token3(2a13,9), Token3(2a14,9)
Token3(2a15,8), Token3(2a16,8), Token3(2a17,8),
Token4(2a0,17), Token4(2a1,17), Token4(2a2,15), Token4(2a3,17),
Token4(2a4,15), Token4(2a5,17), Token4(2a6,15), Token4(2a7,17),
Token4(2a8,15), Token4(2a9,17), Token4(2a10,15), Token4(2a11,17),
Token4(2a12,15), Token4(2a13,17), Token4(2a14,15), Token4(2a15,17),
Token4(2a16,17), Token4(2a17,17),
Token5(2a0,32), Token5(2a1,32), Token5(2a2,32), Token5(2a3,32),
Token5(2a4,32), Token5(2a5,32), Token5(2a6,32), Token5(2a7,33),
Token5(2a8,33), Token5(2a9,33), Token5(2a10,32), Token5(2a11,32),
Token5(2a12,32), Token5(2a13,32), Token5(2a14,32), Token5(2a15,32),
Token5(2a16,32), Token5(2a17,32),
Adj(2a11,2a12,1ang1), Adj(2a7,2a8,1ang2), Adj(2a6,2a7,1ang3),
Adj(2a5,2a7,1ang3),
Par(2a0,2a6,-a), Par(2a1,2a6,-a), Par(2a3,2a12,-a), Par(2a4,2a14,-a),
Par(2a7,2a9,a), Par(2a7,2a10,a), Par(2a7,2a11,a), Par(2a14,2a4,-a).
```

Fig. 2. Symbolic representation of car 2.

Recent development in machine learning has been focused on inductive logic programming (ILP), which has gained success in many domains, and is adopted in our system. We give a brief definition of ILP here. For more details please refer to reference (16). The learning agent of ILP is provided with background knowledge B, positive examples E, negative examples N, and constructs a hypothesis H. B, E, N, and H are logic programs. A logic program is a set of definite clauses each having the form

$$h \leftarrow b_1, b_2, \ldots$$

where h is an atom, referred to as the *head* of the clause, and b_1, b_2,. . .is a set of atoms, referred to as the *body* of the clause. Usually E and N contain only ground unit-clauses with either head or body empty. The conditions for construction of H are $B \wedge H \vdash E$ and $B \wedge H \nvdash N$, that is to say, B and H together must logically imply E, but must not imply N.

Many ILP-based systems have been developed, they differ in the way to construct H. Our system employ FOIL[17] as the learning agent since it has a good performance in many aspects. The system FOIL uses a top-down approach for constructing H. It constructs a rule by appending predicates to the body of the rule gradually. The selection of predicates is based on a measure of information gain.

The action of FOIL is described below.

H is empty. Select a new head literal x to learn, and let a new Horn clause c be x.

1. Choose a literal y according to the conditions described below.
   (a) y and c must contain at least one common variable.
   (b) y has the maximum information gain corresponding to c, i.e. it can contain the most positive examples and the least negative examples.
2. Add y into the body of c.
3. If there are still negative tuples in this new clause, go to 1, if not, add c into H.

Until H can explain all positive examples.

Once the input shapes have been transformed into the form of logic rules, they are stored in a file. FOIL can read the file and generate classification rules. Many alternative techniques in FOIL can improve the performance of learning. For example, we can change the type of each attribute in a predicate, the maximum variable depth, etc. We can also permit whether to find negated literals in a rule or not.

Moreover, the modification of the symbolic representation can be made to meet the need of generating more rules. For example, assume that uth contour and vth contour belong to the same class, they can be represented as `S(u)` and `S(v)`, or `S(u,v)` alone. `S(u)` means that the uth contour belongs to the `S` class. `S(u,v)` means that uth and vth contours are the same class `S`. The former representation can make FOIL run faster, but have few rules. The effect of the latter representation is adverse. We will show this effect in our experiment.

One limitation of the FOIL system is that when more than one feature can tell positive examples from negative ones, FOIL will only extract one of them. This is because the design of FOIL is to classify two classes, but not to describe their differences. To achieve our goal for obtaining descriptive rules, we exclude those tuples which coincide with the predicates that have been learned again and again until all features are found. This will also be shown in the next section.

4. EXPERIMENTAL RESULTS

Our experiment includes two sets. The first set comprises 22 matchbox cars (see Fig. 1). These cars are divided into two classes. The second set includes 14 spoons (see Fig. 3). The first eight spoons are western style, and the rest are Chinese style spoons. Both data are further divided into a training set and a test set. In both experiments, we first used training data to learn, then used testing data to verify whether the results are correct and complete.

4.1. Experiment 1

We divided 22 cars into two classes according to the size of their wheels intuitively: cars 1–14 are classified as class 1, and the remaining cars belong to class 2. Also

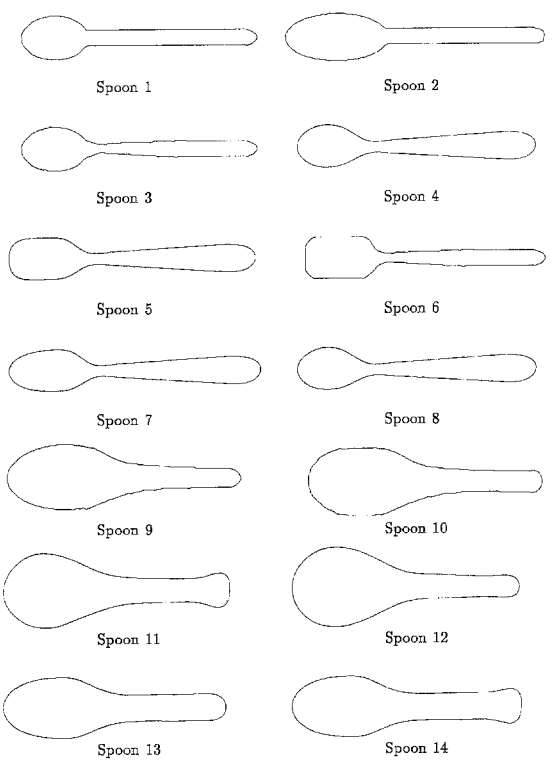


Fig. 3. Contour data of experiment 2.

cars 2, 4, 6, 7, 8, 17, 19, 20, and 21 were selected as the training set, and the rest are in the testing set.

We are interested in the effect of different representations on the learning result, therefore two different forms are used in our experiment. Form I represents an object using head `Car1(A)`, it means that A belongs to class 1; also `Car2(A)` means that A belongs to class 2. Form II uses `Car1(A,B)` to represent that A and B belong to the same class 1, and `Car2(A,B)` means that A and B belong to the same class 2.

The results using form I is shown below. Our iteration is a pass of the FOIL system. The time taken by each iteration is also shown.

```
iteration 1 (0.6 s):
    Car2(A):- not(Car1(A)).
    Car1(A):- not(Car2(A)).

iteration 2 (0.6 s):
   Car1(A):- Have (A,B),
             Token1(B,7),
             adj(B,D,E).

iteration 3 (0.5 s):
   Car1(A):- Have(A,B),
             adj(B,C,lang1).

iteration 4 (34.5 s):
  Car1(A):- Have(A,B),
            Adj(B,C,lang2).
```

After the first iteration, all predicates with form `Car2(X)` were removed from all rules in training data, in order to find more classification characteristics. The rule obtained in iteration 2 indicates that one characteristic to tell class 1 from class 2 is a pair of specific segments, as shown in Fig. 4. We delete `Token(X,7)` from training data. At the third iteration, the characteristics found are shown in Fig. 5. Similarly `Adj(X,Y,lang1)` was deleted before iteration 4. The characteristic found in the fourth iteration is shown in Fig. 6. The learning algorithm is not able to generate any rule after the fourth iteration. It is obvious that the rules generated by FOIL are consistent with our original classification based on the sizes of wheels.

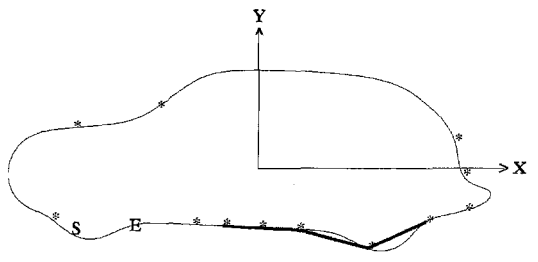


Fig. 4. One characteristic (in bold line) to classify class 1 and class 2, discovered in the second iteration of learning in form I.

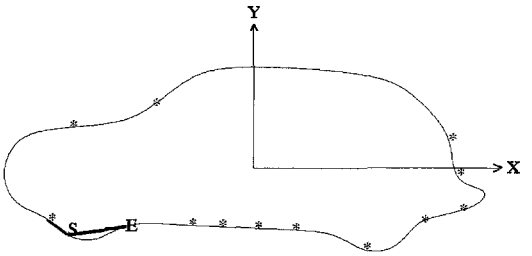


Fig. 5. One characteristic (in bold line) to classify class 1 and class 2, discovered in the third iteration of learning in form I.

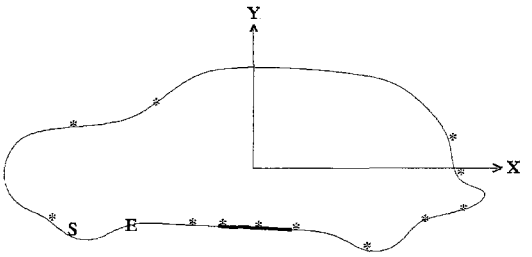


Fig. 6. One characteristic (in bold line) to classify class 1 and class 2, discovered in the fourth iteration of learning in form I.

The result based on form II is

```
iteration 1 (0.8 s):
 Car1(A,B):- not(Car2(A,C)),
             not(Car2(b,C)).
 Car2(A,B):- not(Car1(A,C)),
             not (Car1(B,C)).

iteration 2 (56.7 s):
Car1(A,B):- Have(A,C),
            Adj(C,D,E),
            Have(B,F),
            Adj(G,F,H).

iteration 3 (30.5 s):
Car2(A,A):- Have(A,C),
            Token2(C,13),
            Token1(C,6).

iteration 4 (193.4 s):
Car2(A,B):- Have(A,C),
            Token3(C,9),
            Adj(C,E,0ang3),
            Have(B,G),
            Token2(G,9),
            Adj(G,I,0ang3).
```

Similarly, some predicates were taken away after each iteration in order to obtain more rules. Compared to form I, the rules obtained in the second iteration have already included all characteristics obtained using form I. The third iteration did not generate any interesting result and can be ignored. One more characteristic was found in the fourth iteration, as shown in Fig. 7. As indicated, learning using form II used much more time than using form I.

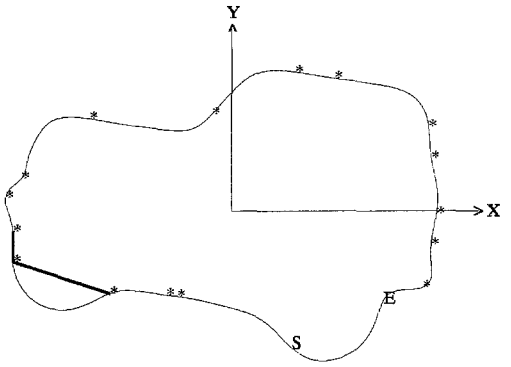


Fig. 7. One characteristic (in bold line) to classify class 1 and class 2, discovered in the fourth iteration of learning in form II.

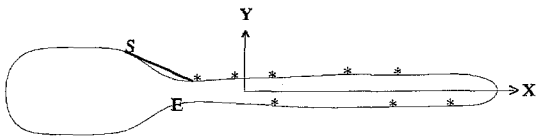
We then used the rules learned for classifying testing data, and the result was consistent. Another test is to perform the same learning but using the testing data. We found the consequent rules are the same as those generated by using the training data.

4.2. *Experiment 2*

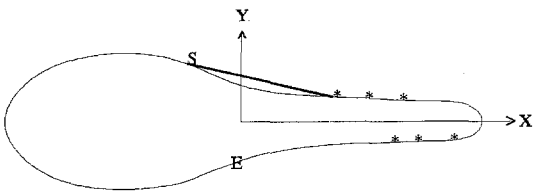
We first split 14 spoons into a training set and a testing set. Spoons 1, 2, 3, 6, 9, 10, 14 are in the training set, and the rest are in the testing set. As in experiment 1, two forms are used in this experiment. Form I is `S1(A)` and `S2(A)`, while form II is `S1(A,B)` and `S2(A,B)`. They have similar meanings as in Section 4.1.

The result using form I is

```
iteration 1 (0.2 s):
     S1(A):- not(S2(A)).
     S2(A):- not(S1(A)).

iteration 2 (0.4 s):
     S1(A):- Have(A,B),
             Token1(B,3).
```



the western style spoon



the chinese style spoon

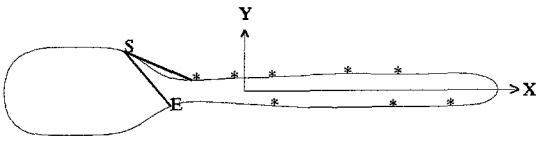
Fig. 8. The corresponding features in counter of the learning result obtained in the 2nd iteration of experiment 2, form I.

After the first iteration, predicates in the form `S2(X)` were taken away from all object's description. The second iteration generated a classification characteristic, as shown in Fig. 8.

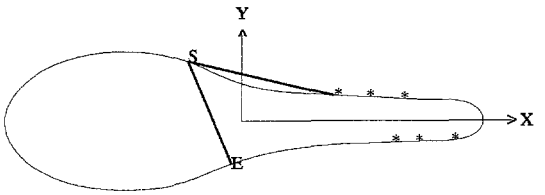
The results using form II is

```
iteration 1 (0.2 s):
   S1(A,B):- not(S2(A,C)),
             not(S2(B,C)).
   S2(A,B):- not(S1(A,C)),
             not(S1(B,C)).

iteration 2 (9.0 s):
  S1(A,B):- Have(A,C),
            Token1(C,3),
            Have(B,E),
            Token1(E,3).

iteration 3 (16.9 s):
  S1(A,B):- Have(A,C),
            Token2(C,17),
            Adj(C,I,J),
            Have(B,E),
            Token2(E,17),
            Adj(E,G,H):

iteration 4 (18.1 s):
   S1(A,B):- Have (A,C),
             Have(B,D),
             Token1(C,7),
             Adj(C,F,G),
             Adj(D,H,G).

iteration 5 (172 s):

iteration 6 (19.7 s):
   S2(A,B):- Have(A,C),
             Token1(C,2),
             Adj(I,C,J),
             Have(B,E),
             Token1(E,2),
             Adj(G,E,H).

iteration 7 (20.1 s):
   S2(A,B):- Have (A,C),
             Have(B,D),
             Token3(C,9),
             Adj(C,F,G),
             Adj(D,H,G).
```

`S2(X,Y)` was excluded from all descriptions after iteration 1. The second iteration obtained the same result as the second iteration using form I. After `Token1(X,3)` was deleted, the third iteration generated rules that correspond to Fig. 8, then `Token2(X,17)` was removed. The fourth iteration generated a rule as shown in upper spoon of Fig. 10, then `Token1(X,7)` was deleted. Note that no rule was generated in iteration 5, so we removed one more predicate `S1(X,Y)`, in order to obtain the description of class 2. Iteration 6 generated characteristics shown in Fig. 9, then `Token1(X,2)` was taken away. Finally, iteration 7 generated characteristics shown in the lower spoon of Fig. 10. Similarly, much

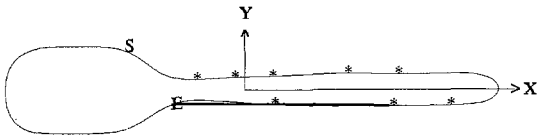


the western style spoon

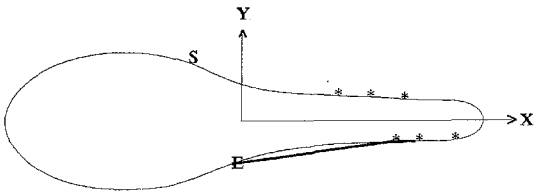


the chinese style spoon

Fig. 9. The corresponding features in counter of the learning result obtained in the 3rd iteration of experiment 2, form II.



the western style spoon



the chinese style spoon

Fig. 10. The corresponding features in counter of the learning result obtained in the 3nd and 4th iterations of experiment 2, form II.

more time was consumed compared to the learning in form I.

We followed the same procedure used in the first experiment to check the consistency of training data and testing data. The experimental results show that the consistency exists. That is, the obtained rules using the training data are the same as those using the testing data.

5. CONCLUSIONS

Experimental results demonstrated the capability of our method on knowledge extraction from shape contours. The generated rules reflect the properties of contours that tell one class from the other, and are easily understandable to humans. Our approach is rotation invariant and noisy resistant. Using the principal axes transform makes contours located in a consistent co-ordinates, no matter how contours rotate. The smoothing method eliminates the effect of noisy data.

Limitation of our approach are discussed below. First, our method is not scale invariant. Second, if the features used to discriminate different classes of objects are only small sharp protrusions or indentations on contours, these features might be filtered by the smoothing method. Third, the k-curvature algorithm may not segment the curvature faithfully. Lastly, while using FOIL to learn, we have to assign some parameters to FOIL. These parameters can change the type of each attribute in a predicate, the maximum variable depth, the appearance of a negated literal, the minimum accuracy of any rule, etc. In FOIL learning, these parameters affect performance greatly. The values of these parameters are assigned basing on our present experience.

REFERENCES

1. D.S. Hogg, Shape in machine vision, *Image and Vision Computing* **6**(11), 309–316 (1993).
2. L. Jovanović, Learning algorithm based on modified structure of pattern classes, *Proc. 11th IAPR Int. Conf. Pattern Recognition*, 487–490, (1992).
3. N. Katzir, M. Lindenbaum and M. Porat, Curve segmentation under partial occlusion, *IEEE Trans. Pattern Anal. Mach. Intell. PAMI* **16**(5), 513–519 (1994).
4. N. Ueda and S. Suzuki, Learning visual models from shape contours using multiscale convex/concave structure matching, *IEEE Trans. Pattern Anal. Mach. Intell. PAMI* **14**(4), 337–352 (1993).
5. N. Ansari, Shape recognition: A landmark-based approach, UMI Dissertation Services (1988).
6. P. Saint-Marc, H. Rom and G. Medioni, B-spline contour representation and symmetry detection, *IEEE Trans. Pattern Anal. Mach. Intell. PAMI* **15**(11), 1191–1197 (1993).
7. H. Rom and G. Medioni, Hierarchical decomposition and axial shape decomposition, *IEEE Trans. Pattern Anal. Mach. Intell. PAMI* **15**(10), 973–981 (1993).
8. J.H. Connel and M. Brady, Generating and generalizing models of visual objects, *Artificial Intell.* **3**, 159–183 (1987).
9. Hättich, W. H. Wandres, Automatic learning of structural models for workpiece recognition systems, *IEEE Trans. Pattern Anal. Mach. Intell. PAMI* **12**(3), 279–281 (1990).
10. O. Fichera, P. Pellegretti, F. Roli and S. B. Serpico, Automatic acquisition of visual models for image recognition, *Proc. 11th IAPR Int. Conf. Pattern Recognition*, 95–99 (September 1992).
11. F. Mokhtarian and A. Mackworth, Scale-based description and recognition of planar curves and two-dimensional shapes, *IEEE Trans. Pattern Anal. Mach. Intell. PAMI* **8**(1), 34–43 (1986).
12. R. C. Gonzalez and R. E. Woods, *Digital Image Processing*, Addison-Wesley, Reading, MA (1992).
13. A. Rosenfeld and E. Johnston, Angle detection on digital curves, *IEEE Trans. Computers* **22**, 875–878 (1973).
14. E. Saund, Identifying salient circular arcs on curves, *Computer Vision Graphics Image Process.* **58**(3), 327–337 (1993).
15. H. Shvaytser, Learnable and nonlearnable visual concepts, *IEEE Trans. Pattern Anal. Mach. Intell. PAMI* **12**(5), 459–466 (1990).
16. S. Muggleton, Inductive logic programming: derivations, successes and shortcomings, *SIGART Bulletin* **5**(1), 5–11 (1994).
17. R. M. Cameron-Jones and J. R. Quinlan, Efficient top-down induction of logic programs, *SIGART Bulletin* **5**(1), 33–42 (1994).

About the Author — JUI-CHI HSU received a B.S. from Department of Computer Science and Information Engineering, National Taiwan University in 1990, an M.S. from the Department of Computer Science and Information Engineering, National Chiao Tung University in 1995. His research interests include Pattern Recognition and Machine Learning.

About the Author — SHU-YUEN HWANG is a Professor in the Department of Computer Science and Information Engineering, National Chiao Tung University. He received B.S. and M.S. degrees in Electrical Engineering from National Taiwan University in 1981 and 1983; and a Ph.D. degree in Computer Science from the University of Washington in 1989. His current research interests include Artificial Intelligence, Computer Simulation and Mobile Computing.