Recognition of Objects Using Modified Coded Boundary Representation (MCBR) Method

Katsunori INOUE*, Shuichi FUKUDA**, Masashi OHKUBO***, Tong QIN****

Abstract

This study aims to build a object recognition system which uses Modified Coded Boundary Representation method[1] to store shape data in computers. The method of Coded Boundary Representation has several advantages such that it is very easy to get rotated, mirrored or conjugated shapes by simple list operations, and also very simple to extract or compare the features of shapes. The Coded Boundary Representation method utilizes Freeman's directional codes which is frequently used in the field of image processing. Because of these, it is considered that this method is suitable to develop a flexible object recognition system.

This paper describes the modification of the CBR method, the structure of shape models stored in computers and the algorithm for recognizing simple 3-D shapes got from camera, and demonstrates the effectiveness the MCBR method for object recognition with some illustrative examples.

KEY WORDS : (Object Recognition), (Modified Coded Boundary Representation),(Data Structure of Shape Model)

1. Introduction

It is very important to extract and recognize the shape of objects among line-clusters obtained after the image processing. However, it is very difficult owing to the problems such as the indefiniteness of the positional relations between the camera and objects, the indefiniteness of the object shapes, and the noise included in the image. So previously a method stored the models about object shapes in computers, and then compare them with the obtained object images for recognition.

The paper here will introduce a method which is suitable for the recognition of objects, called Coded Boundary Representation (CBR). Because CBR utilizes Freeman's directional codes used in image-processing field, it is possible to develop flexible object recognition systems.

2. Freeman's Directional Codes

Freeman proposed that a line drawing image can be represented[2], as shown in **Fig. 1 (a)(b)**, as the list of directional codes, to compress the massive binary data. The directional codes regards a line drawing image as the collection of pixels.

For a line such as (b), the directional codes are firstly defined in (a), and then the linkage relation between one pixel on the line with its next pixel on the line can be expressed by a directional code in 0-7. After repeating the second operation, the line can be represented as a sequence of digital numbers. In this way, the image data can be greatly compressed as shown.

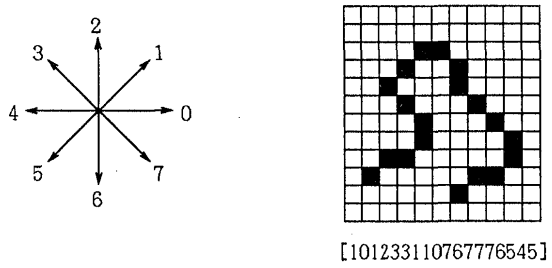


Fig. 1 Freeman's chain code
(a) Directional codes
(b) Code list

3. The method of Coded Boundary Representation (CBR method)

CBR method uses the idea of Freeman's directional codes. **Fig. 2** shows the CBR method which uses the 4 directional codes and represents a shape with the edge list (n_list), the start, and also end, vertex list (s_list, e_list), the length list (l_list) and the directional code list (d_list) of those edges.

† Received on Nov 9, 1991
* Professor
** Professor, Tokyo Metropolitan Institute of Technology
*** Research Associate
**** Assistant Professor, Harbin Institute of Technology , China

Transactions of JWRI is published by Welding Research Institute, Osaka University, Ibaraki, Osaka 567, Japan

The features of this CBR method include that it is easy to operate the lists for carrying out the functions such as rotating, mirroring, decomposing and composing; and it is also easy to get the topological features of a shape according to the d_list; and so on.

4. The Use of CBR method in Image Recognitions

4.1 The Modified CBR method (MCBR method)

To apply the CBR methood to the recognition of objects, it has to be improved. The CBR method's directional codes have been expressed in 4 values: s, e, n, w. Although the clear and simple shape representation is easy for computer to recognize the features of shapes and can carry out the shape operations in an unnumeric way, it is not convenient for recognizing objects if used without any modification.

Therefore, in MCBR method the directional codes are extended to real numbers ($0.0 \leqq d < 8.0$). Because of this change the processing such as decomposing, composing, etc., which used elements of d-list, have also to be modified.

With the MCBR method, it is possible to represent a line in any directions. It is also possible to express position relations, for example, between several lines with equalities or inequalities. Thus, a more flexible image processing can be defined.

4.2 The Shape Data Structure Used in MCBR method

Suppose the line-image obtained after various image processing satisfies the following conditions:

. the focus distance of camera is much larger than the distance Z from camera to object,and the depth of the object is shorter than Z,

. the outside shape loop of the object is completely closed.

Now here is the example of an image of rectangular solid obtained from camera. **Fig. 3 (a)(b)** are the two patterns from the image which have a couple of face loop. Now we show the structure of the shape model stored in computers in Fig. 3. At first each face loop on the rectangular solid is assigned with a quadrilateral type, then it is represented in MCBR method as:

$$\begin{aligned} &\text{n\_list}\ (\text{loop}_i, [N_{i1}, N_{i2}, N_{i3}, N_{i4}]) \\ &\text{s\_list}\ (\text{loop}_i, [V_{i1}, V_{i2}, V_{i3}, V_{i4}]) \\ &\text{e\_list}\ (\text{loop}_i, [V_{i2}, V_{i3}, V_{i4}, V_{i1}]) \\ &\text{d\_list}\ (\text{loop}_i, [D_{i1}, D_{i2}, D_{i3}, D_{i4}]) \\ &\text{l\_list}\ (\text{loop}_i, [L_{i1}, L_{i2}, L_{i3}, L_{i4}]) \end{aligned} \quad (1)$$

where,

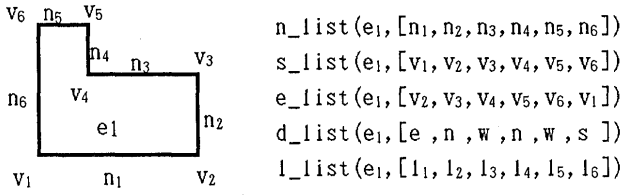
loop_i : identity loop name

N_{ij} : identity line number,
'- N_{ij}' means opposite to 'N_{ij}'

V_{ij} : identity vertex name

D_{ij} : directional code
$(0.0 \leqq D_{ij} < 8.0)$

L_{ij} : line length



e_i : loop name
n_i : edge number
v_i : vertex name
s,e,n,w : directional code
l_i : edge length

Fig. 2 Coded Boundary Representation

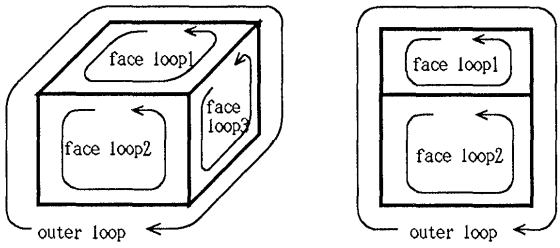


Fig. 3 Image of rectangular solids
(a) 3 loops pattern
(b) 2 loops pattern

All the loops are the tetragons and they should be nearly in the shape of parallelograms. The conditions of one tetragonal loop being a parallelogram can be described as:

$$\left.\begin{aligned} D_{i1} &= D_{i3} \pm 4.0\ (0.0 \leqq D_{i1} < 8.0) \\ D_{i2} &= D_{i4} \pm 4.0\ (0.0 \leqq D_{i2} < 8.0) \end{aligned}\right\} \quad (2)$$

or

$$\left.\begin{aligned} L_{i1} &= L_{i3} \\ L_{i2} &= L_{i4} \end{aligned}\right\} \quad (3)$$

and so on.

In Fig. 3(a)(b) the numbers of face parallelogram loops are 3 and 2 correspondingly, and the condition of a rectangular solid is that these parallelograms must contact each other. These conditions can be described as:

$$\begin{aligned} &{}^{\forall}i \in [1, 2, 3, , , I], j = (i + 1) \bmod I + 1, \\ &{}^{\exists}m, n \text{ that } N_{im} \in [N_{ik}]_k, N_{jn} \in [N_{jk}]_k \\ &\text{make} \qquad N_{im} = -N_{jn} \end{aligned} \quad (4)$$

4.3 The Algorithm of Object Recognition

The data structure given from the image processing unit to the object recognition unit are expressed in the form of:

$$\text{line } (N_i, D_i, L_i, [S_{xi}, S_{yi}, E_{xi}, E_{yi}]) \quad (5)$$

where,

N_i : indentity line number
D_i : directional code $(0.0 \leqq D_i < 8.0)$
L_i : edge length
(S_{xi}, S_{yi}) : start point coordinate
(E_{xi}, E_{yi}) : end point coordinate

In the object recognition unit, the first step is to recognize the misunderstood lines that each of them must have represented more than one real shape edge but was processed as one visible line by the image processing unit owing to the visual angle, and to divide each of them into 2 or more line parts. The **Fig. 4 (a)(b)** is an example that a T type crossed line is divided into two line parts.

The next step in the object recognition unit is to generate an opposite line for each line in the following way:

$$\text{line } (N_i', D_i', L_i', [S_{xi}', S_{yi}', E_{xi}', E_{yi}']) \quad (6)$$

where,

$N_i' = -N_i$
$D_i' = D_i \pm 4.0 \quad (0.0 \leqq D_i' < 8.0)$
$L_i' = L_i$
$(S_{xi}', S_{yi}') = (E_{xi}, E_{yi})$
$(E_{xi}', E_{yi}') = (S_{xi}, S_{yi})$

Then, with total data about the lines, the object recognition unit reconstructs the face triangular loops, tetragonal loops and polygonal loops. All the loops are in the counterclockwise. In this step the rule of "any loop will not use the data once used" is applied, it means that all the loops are constructed only by the line data which have not been used by any other loops yet. This is possible because each line has been added an opposite line, and is necessary for the reconstruction of object shape in next step.

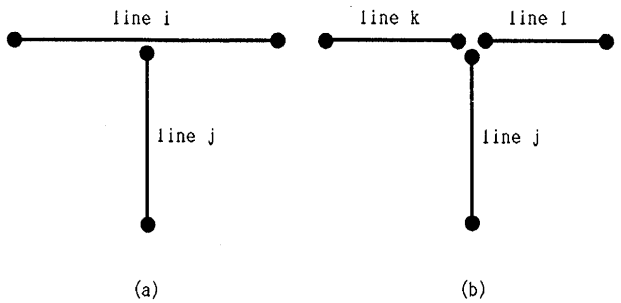


Fig. 4 Dividing process of line data
(a) Line data of T type
(b) Line data after dividing

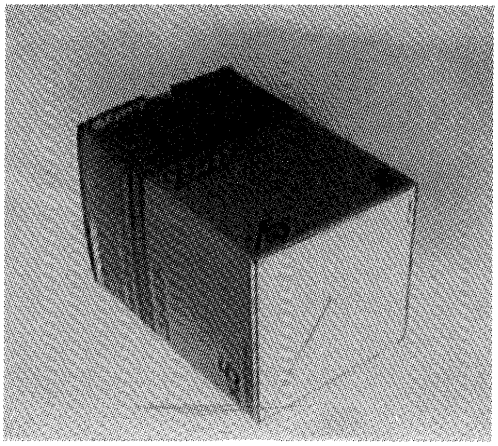
This step is to construct the outside shape loop of the object. The shape loop consists of all the line data that have never been used by inner loops. Certainly this loop is in the clockwise. The data of remained lines are given up since they are the lines which have less than 2 across points to other lines in the shape.

Comparing the received loop set including face loops and shape loop with the data in Database stored before, the shape can be recognized.

5. Examples in Applications

Fig. 5 and **6** show the recognized result of the rectangular solid with the MCBR method. (a) is the original image; (b) is the data of lines derived from (a) after image processing; (c) is the shape lists reconstructed from (b) with the MCBR method; and (d) is the recognized result using shape lists.

6. Conclusion



(a)

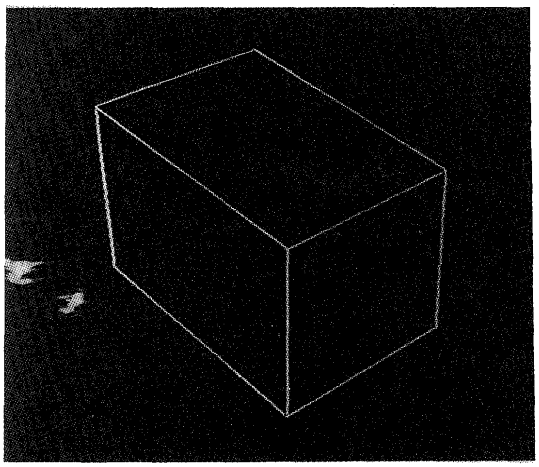
```
line(1,160,6.007957,[268,254,269,414]).
line(2,153,6.150079,[86,116,104,268]).
line(3,218,7.286915,[238,62,423,178]).
line(4,228,7.173979,[86,116,268,254]).
line(5,220,7.077690,[104,268,269,414]).
line(6,161,0.434628,[86,116,238,62]).
line(7,172,0.580439,[268,254,423,178]).
line(8,151,1.907411,[412,329,423,178]).
line(9,166,0.682833,[269,414,412,329]).
```

(b)

```
4
n_list(shape,[6,3,-8,-9,-5,-2])
s_list(shape,[v6,v5,v3,v2,v1,v7])
e_list(shape,[v5,v3,v2,v1,v7,v6])
l_list(shape,[161,218,151,166,220,153])
d_list(shape,[0.434628,7.28691,5.90741,4.68283,3.07769,2.15008])
```

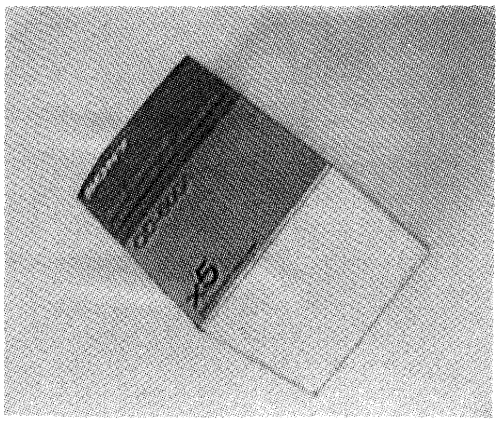
```
n_list(13,[5,-1,-4,2])
s_list(13,[v7,v1,v4,v6])
e_list(13,[v1,v4,v6,v7])
l_list(13,[220,160,228,153])
d_list(13,[7.07769,2.00796,3.17398,6.15008])
n_list(12,[7,-3,-6,4])
s_list(12,[v4,v3,v5,v6])
e_list(12,[v3,v5,v6,v4])
l_list(12,[172,218,161,228])
d_list(12,[0.580439,3.28691,4.43463,7.17398])
n_list(11,[9,8,-7,1])
s_list(11,[v1,v2,v3,v4])
e_list(11,[v2,v3,v4,v1])
l_list(11,[166,151,172,160])
d_list(11,[0.682833,1.90741,4.58044,6.00796])
```

(c)



(d)

Fig. 5 Result of a loops pattern
(a) Original image
(b) Data of lines derived from (a)
(c) Reconstructed shape lists with (b)
(d) Recognition result



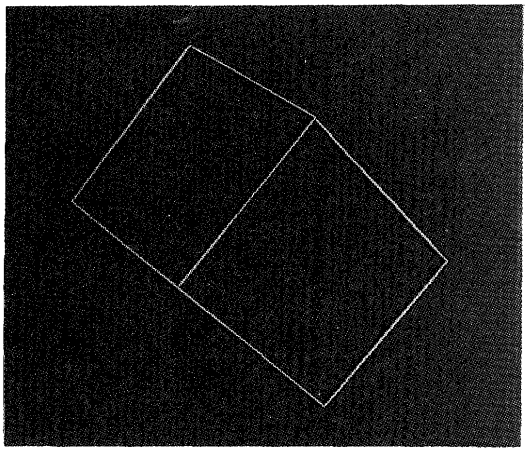
(a)

```
line(1,319,7.132795,[190,53,438,254]).
line(2,192,6.962563,[69,196,201,336]).
line(3,140,7.336757,[201,336,323,406]).
line(4,187,1.105859,[69,196,190,53]).
line(5,213,1.135301,[201,336,335,170]).
line(6,190,1.175325,[323,406,438,254]).
```

(b)

```
3
n_list(shape,[102,-6,-3,-2,4,101])
s_list(shape,[v3,v2,v1,v4,v6,v5])
e_list(shape,[v2,v1,v4,v6,v5,v3])
l_list(shape,[133,190,140,192,187,186])
d_list(shape,[7.13279,5.17532,3.33676,2.96256,1.10586,7.13279])
n_list(12,[5,-101,-4,2])
s_list(12,[v4,v3,v5,v6])
e_list(12,[v3,v5,v6,v4])
l_list(12,[213,186,187,192])
d_list(12,[1.1353,3.13279,5.10586,6.96256])
n_list(11,[6,-102,-5,3])
s_list(11,[v1,v2,v3,v4])
e_list(11,[v2,v3,v4,v1])
l_list(11,[190,133,213,140])
d_list(11,[1.17532,3.13279,5.1353,7.33676])
```

(c)



(d)

Fig. 6 Result of a loops pattern
(a) Original image
(b) Data of lines derived from (a)
(c) Reconstructed shape lists with (b)
(d) Recognition result

As discussed above, the features of MCBR method are that it is easy to carry out the image processing by the simple list operations and easy to extract the shape features of an image. Our next work will aim at using the CBR method in recognizing the moving objects and the stereo-photographic applications.

References

1) Shuichi Fukuda: A New Shape Model: Coded Shape Representation, The 3rd Computational Mechanics Conference, JSME, No. 900-69, 1990, 183-184.
2) Freeman.H: On the Encoding of Arbitrary Geometric Configurations, IRE Trans. Electronic Computers, Vol. EC-10. No2, 1961, 260-268.