A Proposal Method for Corner Detection with an Orthogonal Three-Direction Chain Code

Hermilo Sánchez-Cruz

Centro de Ciencias Básicas. Universidad Autónoma de Aguascalientes
Av. Universidad 940, Col. Universidad, CP. 20100
Aguascalientes, Aguascalientes. México. Fax: (52 449) 9 10 84 01
hsanchez@correo.uaa.mx

Abstract. Only three set of pattern chain elements to detect corners in irregular shapes are introduced. A code based on three orthogonal change directions, when visiting a contour shape, are used. Previous approaches for detecting corners employ eight different symbols and usually compute angles and maximum curvature. The three basic pattern contour chain elements, founded in this paper, represent changes of direction in the contour curves, requiring few computing power to obtain corners. Also, we have found that the method is independent of shape orientation.

Keywords: Shape corner; Contour; Chain element; Freeman chain code; Symbol chain code; Pattern substrings.

1 Introduction

Nowadays, corner detection of shape objects is an active field in object recognition and image retrival. In literature, usually the aim in obtaining corner points by computing angles of curvature on the contours of shapes is studied and representing discrete contours by Freeman chain codes. Freeman and Davis [1] proposed to find corners by computing incremental curvature to represent contour shapes by an eight-direction chain code. Since then, many authors have suggested to use this code when representing contour shapes. Part of the algorithm presented by Teh and Chin [2] consists on computing the curvature of contour points and detecting corners by a process of nonmaxima suppression. Liu and Srinath [3] have compared a number of corner detectors due to Medioni and Yasumoto [4], Beus and Tiu [5], Rosenfeld and Johnston [6], Rosenfeld and Weska [7] and Cheng and Hsu [8]. All those authors represented samples of shapes through a sequence of eight direction changes from 0-7, known as the Freeman Chain Code [9].

Wu [10] proposed an adaptive method to find local maximum curvatures of digital curves. Sobaina and Evans [11] described a corner detection from segmented areas using mathematical morphology employing paired triangular structuring elements.

Basak and Mahata [12] developed a connectionist model along with its state dynamics for detecting corners in bynary and gray level images. We have studied that it is suitable to represent any binary closed shape with binary resolution cells, and using only three symbols of a chain code, without loss of information

J. Blanc-Talon et al. (Eds.): ACIVS 2006, LNCS 4179, pp. 161–172, 2006.

[13]. This method of chain code is sufficient to represent binary shapes and represents low cost in storage memory. Techniques due to Freeman chain codes in finding corner detection are based in eight different directions (see Fig. 1a).

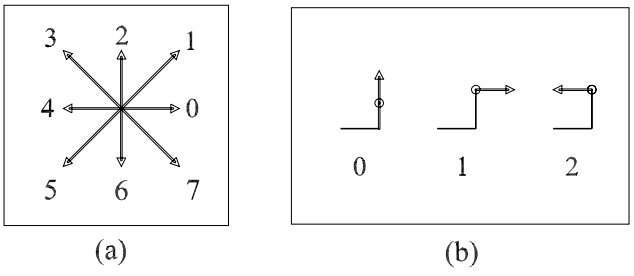


Fig. 1. Two different chain codes: (a) The eight different directions given by Freeman chain code. (b) The three orthogonal change directions.

We propose here to use a method of only three relative direction changes given by Fig. 1b. An advantage in using three symbols is its low storage power, as can be seen by the recent work duo to Sánchez-Cruz & Rodríguez-Dagnino [13]. They found that coding with three symbols is sufficient to represent binary shapes saving storage efficiently. Recently Yong Kui Liu & Boruk Zalik[15], found efficient storage properties by using Huffman coding applied on change directions of Freeman chain code.

For each orthogonal change direction code, chain segments are divided in three parts (given in Fig. 1b): a *reference segment* (in Fig. 1b appears as horizontal segment in each code), *a basis segment* (perpendicular to reference segment) and a segment indicating a direction change with regard to reference segment.

The meaning of the three symbols (see Ref[14] for 3D case), given by the set $\mathcal{C} = \{0,1,2\}$, is as follows: the element 0 represents the direction change which means to "go straight" through the contiguous straight line segments following the direction of the last segment; the '1' indicates a direction change upward with regard to the reference segment; and '2' means to "go back" with regard to the sense of the reference segment. In this work we have noticed that when the symbol '2' appears in a contour shape, can easily indicate an existing corner. In Section 2 definitions concerning to this article are presented, seeking the problem as a pattern substring search. In Section 3 some rules to detect shape corners are proposed; in Section 4 experimental proving of postulated rules are applied on some binary shapes; in Section 5 rotation independence is analized, and in Section 6 we give some conclusions.

2 Some Definitions

Our proposal method considers to find a specific set of pattern substrings of length l, trying to find all those substrings in a shape contour coded by a chain

that match with those patterns. Let us consider, for example $l = 11$ as the length of the substrings. Which substrings are all composed of 11 symbols and which of them are considered corner chains? In fact, there are substrings composed of 11 symbols, of course, not all are considered corner chains due to its low curvature or because the region they are associated in the contour shape is not "well behaved", as we explain at once.

Let $\mathcal{P}$ denote the *complete chain code* (or simply *chain code*) associated to the shape contour, given by the string of symbols p_i of eq(1).

$$\mathcal{P} = p_1 p_2 \cdots p_n, \tag{1}$$

and P the contour discrete perimeter, given by the number of symbols of the chain code.

Consider a substring template of l symbols: $C \in \mathcal{P}$, given by eq(2).

$$C = a_1 a_2 \cdots a_l, \quad l \ll P, \tag{2}$$

as a *contour chain element*, or simply: *chain element*, this is, a small piece of contour from the whole shape contour.

Let us consider $m = l/2$ the middle point of a substring of size l, so that a_m, the pivot, be the center of the substring. It is possible to associate a pair of line segments to any chain element. They can be drown up from to the opposite end points, producing an angle φ. We define a *well behaved* chain element when an angle has been subtended by a pair of associated line segments such that the chain does not form loops. Observe a sample of chain elements and their corresponding visual meaning in Fig. 2.

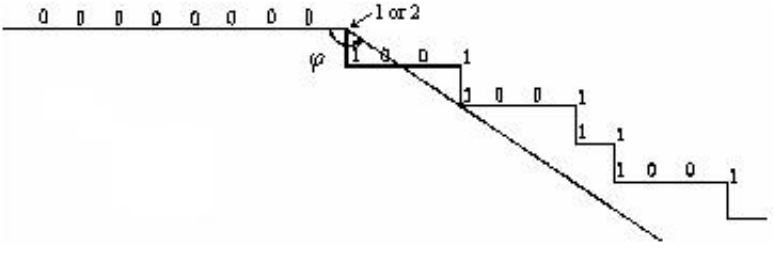


Fig. 2. Angle φ of curvature. Consider a 5-neighborhood of pivot (1+2): 00000(**1+2**)10011.

We define a *neighborhood of radii r*, when considering a piece of the complete string; this region is composed of a small number of symbols in comparing with the whole contour chain code, r symbols on one side of a particular pivot symbol, and r symbols on the other side of the pivot symbol. Fig. 2 presents an example of neighborhood of radii 5: chain elements having 00000 in its left first part, 10011 in the right part, and 1 or 2 (1+2 to abreviate) as a pivot symbol could appear on the contour shape.

We propose to save calculation of corner-angles or curvature changes directly. Instead we give a family of substrings that represents high curvature. Well behaved substrings should not be considered a corner chain when their corner chains associated angles are so much obtuse.

Another definition we need is a *well behaved contour shape*, this is, a contour shape having been smoothed in such a manner that there is no noise or local defects.

3 Rules for Detecting Chain Corners

In this work we did our experiments with a vicinity of eleven symbols in chain elements giving good results in finding chain corners.

A way to obtain a complete set of templates considered chains corners, is to search all the substrings arrays composed of l symbols from the set $\mathcal{C} = \{0,1,2\}$, calculate the angle associated to each substring and apply the threshold to see if it is a chain corner. But we propose a small enough set of template substrings to find the evident chain corners from an arbitrary set of 2D shapes. To find a group of pattern substrings or pattern chain elements considered as chain corners is to focus in a vecinity of each change code contour, by for example eleven chain segments, nine of them labeled with symbols, representing orthogonal direction changes. The first two are called reference segment and basis segment, respectively. There are a huge number of combinations given by nine symbols (11 segments) and we are interested on finding chain elements that have no loops. Even more, fixing the reference segment of the chain, there are 3^9 combinations duo to the other nine chain directions. Fig. 3 presents part of these combinations.

By analizing the different chain sets mentioned, we have observed that pattern substrings representing corners. Parameters we have to take into account are the next:

l: states for the size of substring.

q: represents "many" times a symbol is repeated in a substring. This quantity depends on resolution of binary object. By *many* we define that the number of symbols is greater than $l/4$, so $q \in (l/4, l]$.

Part of the study made to find a simple pattern of substrings that represent corners, was to find that we have to consider only the cases when there is an appreciable direction change when visiting the discrete contour. At first glance, this happens with high probability when a symbol '2' from the orthogonal changes appears. As was introduced in Section 1, this symbol represents "go back" when covering the shape contour, indicating a corner shape. Of course, in a given situations, where there is some noise or local shape defects, '2' symbol should not constitute a perceptible corner. So, our first pattern chain class to be proven is composed of a string of 1s or 0s, then a '2' symbol and then a sequence of 1s or 0s again.

Another pattern chain elements, that represents contour change directions, ocurres when there are many 0s (with possibly some pairs of 1s) following a '1'

Fig. 3. Part of the complete set of 3^9 substrings

as a middle symbol of the chaing element, following again many 0s (with possibly some pairs of 1s).

The other pattern chain elements represents only changes of directions of the contour shape. Changes of directions ocurre simply when there are a substring of many 1s, with possibly some 0s, following a substring of many 0s (with possibly some 1s); or viceversa, a substring of many 0s (with possibly some 1s) following a substring of many 1s (with possibly some 0s).

So, to simplify the pattern of chain elements that correspond to chain corners, we are talking about pattern strings of discrete chain corners, postulated by next regular expressions:

$$S_1 = (0+1)^{l/2}(\mathbf{2})(0+1)^{l/2},$$
$$S_2 = (0^q + 1_p)^{l/2}(\mathbf{1})(0^q + 1_p)^{l/2},$$
$$S_3 = (0^q + 1_p)^{l/2}(\mathbf{1})(0 + 1_p{}^q)^{l/2} + (1^q + 0)^{l/2}(\mathbf{1})(1 + 0^q)^{l/2}, \tag{3}$$

where q represents *many* symbols, p states for a pair of symbols: $p = \{0, 2\}$. Rule S_1 states that when a symbol 2 appers, the chain element can be considered a corner chain. Rule S_2 means there are many zeros in both sides of the middle point of the the chain element. Rule S_3 means that substring has many zeros or ones in the first part of the chain element and many ones or zeros in the second part of the chain. See Fig. 4 for a sample of these rules. Our proposed method relies on looking for these pattern substrings on any contour shape.

We consider shapes represented by resolution cells, each having a value 0 or 1. For the implementation of an algorithm to encode this shape we have to visit the ones that represent the contour shape, i.e., the ones of the boundary. We follow the contour of the shape clockwise sense, updating in every step, the reference segment with the contiguous segment, and giving one of the three symbols according to each orthogonal change direction (see Ref[13] for a detailed explanation).

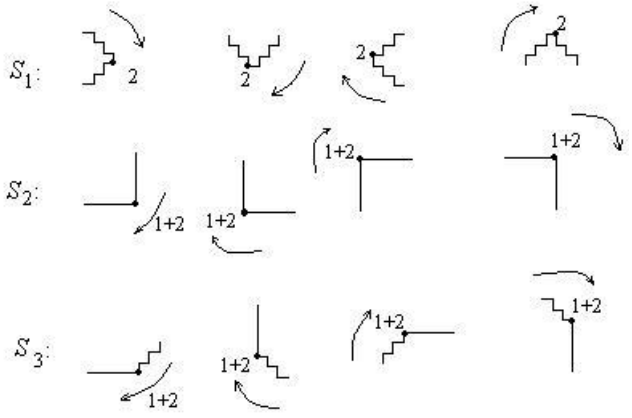


Fig. 4. Samples of the three types of corners, each invariant under rotations tranforms

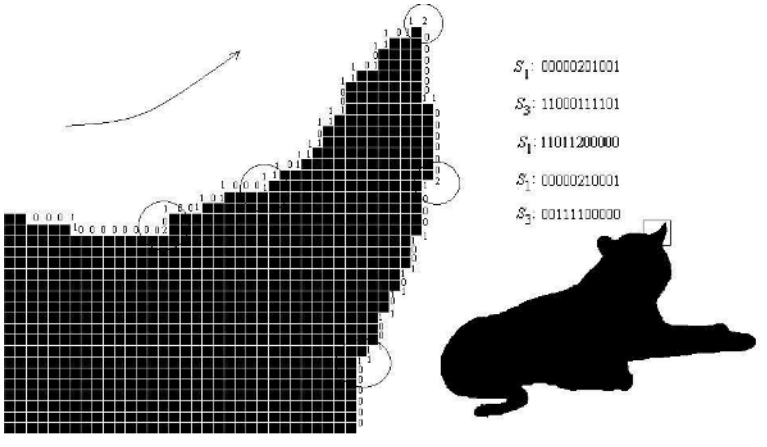


Fig. 5. Examples of chain elements, covering the contour on clockwise sense. The grid is part of the inner shape.

The object shape is confined to a minimum rectangle that is visited line by line, from left to right and from top to bottom. The first cell resolution, of the object to be visited, is that which appears at the leftmost and highest part of the occupied region. Fig. 5 shows part of a contour shape and examples of chain elements coded by the three symbols of the orthogonal directions given in Fig. 1b. Given this representation, we can reconstruct the original image by interpreting the code of every symbol in terms of the direction changes that can follow.

Finally, the pattern substrings, S_1, S_2 and S_3 are parsing the resulting chain string of the complete contour.

4 Experiments

Consider the set of four shapes $\mathcal{S} = \{$*Irregular shape, Circles, Hammer, Tigger* $\}$ showed in Fig. 6 as binary objects.

Consider *Irregular shape* object and its corresponding chain code (Fig. 7), 51 chain corners were found in its conotur shape. Some of them are so closed, in

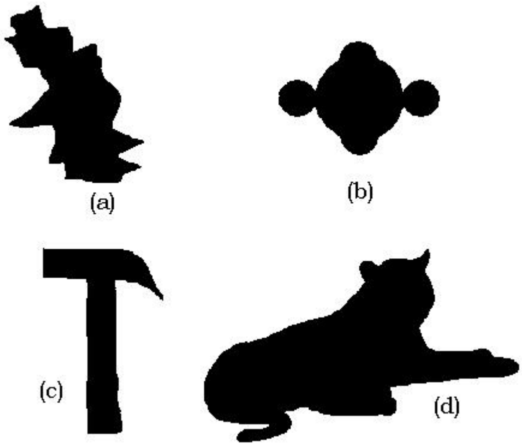
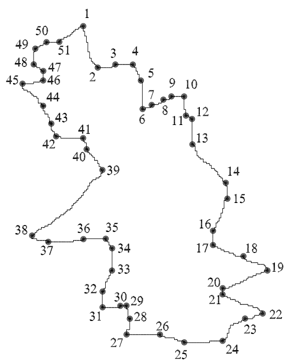


Fig. 6. Four shapes: (a) *Irregular shape*; (b) shape of intersecting *Circles*; (c) *Hammer*; (d) *Tigger*



(a)

0111120011001100011000001100011011111111110000000000210110000000000002111101101100011110000000000000000
00010210011110001101111000111100000000200000001100010111110000000000000000111100110011111010101
10111010111110111111011111011000000021110000110011001101111001101100000000211011011011001100111100
1100011111100110001100011111020110110011011001111011011001101101100010002110110110011011111000011
001100111101020110011110000110110110011010110011001110110000000110000000000000002100011011000001
1001100000000000000000000100201000000210000011010002100000000002000000000210110001101101100110
0000000000000210110111110000000000000021000001100000000000000021000000110010211111111011011011011
1110111101111110111011111111111111101111110110111110110111011021011011010101111111101100011110100000
00000000110011101111000110001101010110011010100111011111110110111002000000110001101000002111110111
110000000210111011011011000000011110110110111111111

(b)

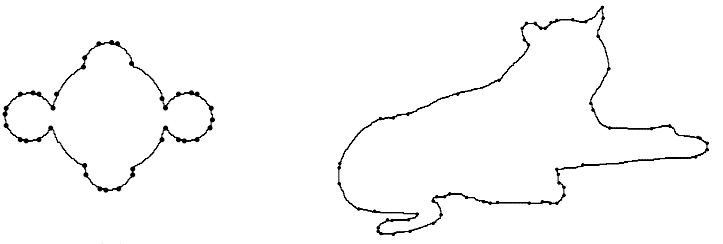
Fig. 7. *Irregular shape* and its corresponding 3-symbol chain code: (a) the shape; (b) its chain code

Table 1. Chain elements encountered from the Fig. 7 that belong to one of the two classes of chain patterns postulated. Corners 5,9,27,41 and 46 are split by two closed chain corners.

Num corner	*Chain element*	*Class pattern*	*Num corner*	*Chain element*	*Class pattern*
1	01111200110	S_1	12	01111100000	S_3
2	11111100000	S_3	13	00000111100	S_3
3	00000210110	S_1	14	11101100000	S_3
4	00000211110	S_1	15	00000211100	S_1
5a	11000111110	S_3	16	01101100000	S_3
5b	00111100000	S_3	17	00000211011	S_1
6	00010210011	S_1	18	10000111111	S_3
7	01111100011	S_3	19	11110201101	S_1
8	10111100011	S_1	20	11000100021	S_2
9a	11000111100	S_3	21	01000211011	S_1
9b	00111100000	S_3	22	11010201100	S_1
10	00000200000	S_1	23	10000110110	S_3
11	11000101111	S_3	24	11101100000	S_3
12	01111100000	S_3	25	00000210001	S_1
26	01101100000	S_3	39	10110210110	S_1
27a	00000100201	S_2	40	11011000111	S_3
27b	00100201000	S_1	41a	11000111110	S_3
28	00000210000	S_1	41b	11110100000	S_3
29	00000110100	S_2	42	00000110011	S_3
30	01000210000	S_1	43	10111100011	S_3
31	00000200000	S_1	44	11000110101	S_3
32	00000210110	S_1	45	11100200000	S_1
33	11001100000	S_3	46a	11000110100	S_2
34	00000210110	S_1	46b	00110100000	S_2
35	11111100000	S_3	47	00000211110	S_1
36	00000210000	S_1	48	11111100000	S_3
37	00000210000	S_1	49	00000210111	S_1
38	10010211111	S_1	50	01101100000	S_3
			51	00000111101	S_3

such a manner that their corresponding pivots are in the neighborhood of each other, in this case we could define only one corner. In Table 1 is listed each of the chain elements and the corresponding class pattern given by eq(3) of the *Irregular shape* contour.

From Figures 7 to 10, contour shapes of the set $\mathcal{S}$ and their corresponding chain codes are presented. They also show the results of applying the method proposed to search chain corners given a substring length of $l = 11$.



10000021000011110111101111101111101111101100001101110011011011111110111111111111011111110110111101100001100
01100012011111101111101111011110000110000021000011110111011111011111101100001100002100001101111110111
11101111011110000110000021000011110111101111101111211100110000110111101101111101111111101111111111
10110110111011000011011111101111101111011110000110000021000011110111011111011111101100001100101101101
11111110111111111111011111101101110100110001100111112111101111011110000110000021000011110111101111012001
10011011000111101111011011111101111111111110111111110110111001100001101111101111011110000110000
021000011110111

(c)

12000000110000002100001100110110110011110000000021011011111011011110011011000110011000110001100001100
00000002101100011011010101100111011001101101011101011000111111111011011011000021110000110011011110
11001111011011011010111111011001101100100000000000000000000000110000000000000000000000000000000210
00011000011000110000000002100111101011001101110111100000011000000110000011000011001101111011110111
11000000002010010111111001100011011110000011000000000000000000000000000000011000000110000001100
00000000000000000001100000001100000110000110000011000000000000000011000000001100000000000000000000
01101100001101100000000000000001100110000001002100011000000000110100101010111011000000000210111100101
11111110110000000021110000000002101100000000000000021000000000000000000000000000000000000000000021000
00000021001100000011001102121011001111100000000001101101101010000000000000000210011011011000000000011
00011100021101100101101101111000110000021000010110111110100111000110011110011001100110011011000110001
11011000000011000000000011001012100000000000000002120111110000210011111110110110111112100000210000
00000000000000000011011000110001010021100000000000000001100000000000000000000000000000000000011000000011
02121011000011000001100000110011101111111101111111111010110110110001100111110011011001100000000000
00000000021000001100110011011010011011011011010110111101111111011111111011111110111111111111110111
10110110110101111101101101101100110000011000011210000021011001010000011000000001010000011011001101010
11011001100110001101101111011110011001111111001100110001100011011000011100000110110110000011000011
00110000110001100011001101100111101111001111001100110111111101101111010111011111101011110011001011011
11111110101111111111111100101011011110002111011111111000110110000110002111011111010100000000000020100
11111101111000002111001011101011100110001111000100000000000000002100001100011000000002010011011000 1
11101111101111001101011101

(d)

Fig. 8. Objects and their chain codes: (a) *Circles*; (b) *Tigger*; (c) and (d) their respective chain codes

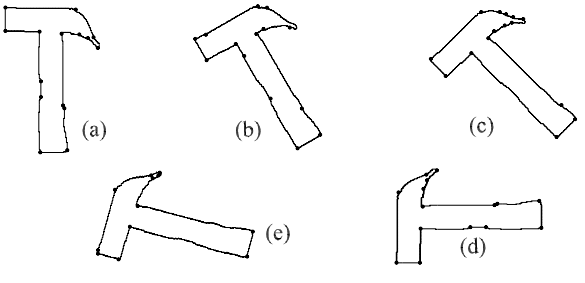
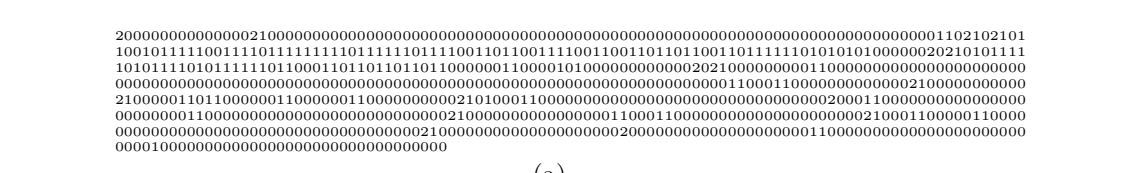
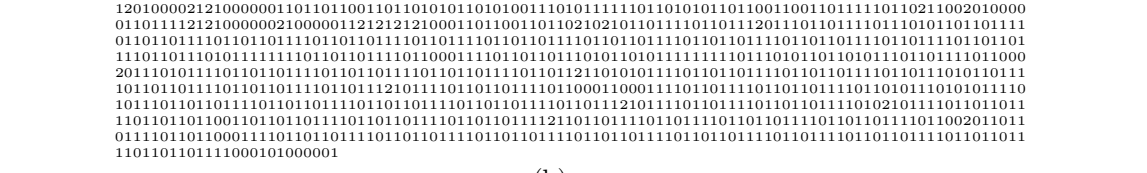


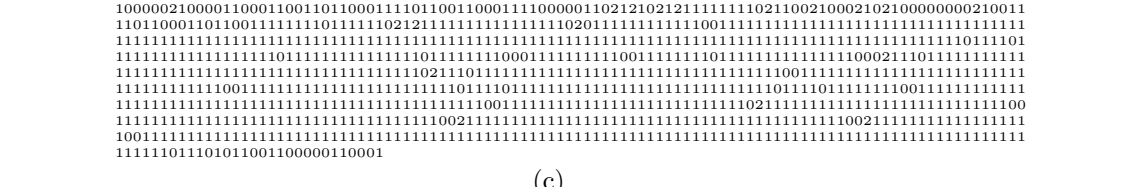
Fig. 9. *Hammer* shape and its corners despite rotations: (a) original shape; rotated (b) 30°, (c) 45°, (d) 75° and (e) 90° in counterclockwise sense



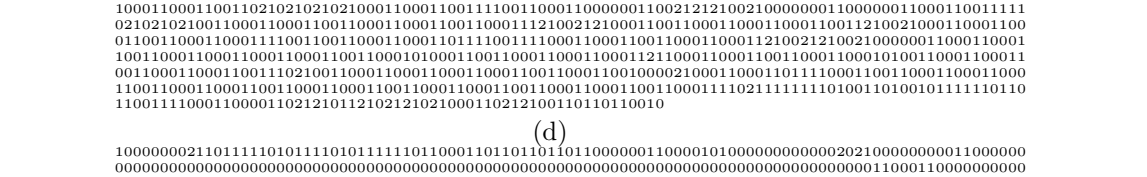
(a)



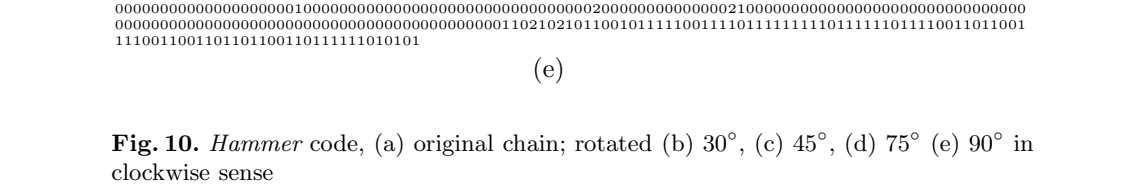
(b)



(c)



(d)



(e)

Fig. 10. *Hammer* code, (a) original chain; rotated (b) 30°, (c) 45°, (d) 75° (e) 90° in clockwise sense

5 Rotation Independence

When a rotation transformation is applied to any shape, a grid basis is maintained fixed, so the chain code of the same figure changes substantially. Nevertheless, most important corners can be found with the set of pattern substrings given by the rules (3).

Searching the pattern substrings given by eq(3) the most important shape corners are conserved in *Hammer* shape (Fig. 9), despite its chain code changes when rotated (as can be seen in Fig. 10).

6 Conclusions

To save time and memory storage to manage this kind of objects, we used three symbols that represent orthogonal directions when covering contours of binary shapes. With this method we have found shape corners including there where is apparently circular, like happenning with figure constructed by intersecting circles. We found three classes of patter substrings to obtain the most important shape corners in contour shapes, preventing to compute angles and curvatures directly; so we have presented a new research topic. Also, we have analized that the proposed metod is invariant under rotations of shape contours. As future work most be studied if this method is invariant under scale transforms. A universal and simplified set of pattern substrings, comparing with other chain codes in literature is suggested to be investigated.

Acknowledgments

We would like to thank PROMEP program and CONACyT council for their support in finishing this work.

References

1. H. Freeman and L. S. Davis, A Corner-Finding Algorithm for Chain-Coded Curves. IEEE Trans. Comput. 26: (1977) 297-303.
2. C-H. Teh, and R.T. Chin, On the Detection of Dominant Points on Digital Curves. IEEE Trans of Pattern Anal and Mach Int. 11 (8) (1989) 859-872.
3. Hong-Chih Liu; M.D. Srinath. Corner Detection From Chain-code. Pattern Recognition. 23 (1/2) (1990) 51-68.
4. G. Medioni; Y. Yasumoto. Corner detection and curve representation using cubic B-Splines. Comput. Vision Graphics Image Process. 39: (1987) 267-278.
5. H.L. Beus; S.S. H. Tiu. An improved corner detection algorithm based on chain-coded plane curves. Pattern Recognition. 20 (1987) 291-296.
6. A. Rosenfeld; E. Johnston. Angle detection on digital curves. IEEE Trans Comput. 22: (1973) 875-878.
7. A. Rosenfeld; J.S. Weszka. An improved method of angle detection on digital curves. IEEE Trans. Comput. 24: (1975) 940-941.
8. F. Cheng; W. Hsu. Parallel algorithm for corner finding on digital curves. Pattern Recognition Lett. 8: (1988) 47-53.
9. H. Freeman. On the Encoding of Arbitrary Geometric Configurations, IRE Trans. on Electr. Comp. 10 (2) (1961) 260-268.
10. W. Wen-Yen. An adaptive method for detecting dominant points. Pattern Recognition. 36 (2003) 2231-2237.
11. A. Sobaina & J.P.O. Evans. Morphological corner detector using paired triangular structuring elements. Pattern Recognition. 38 (2005) 1087-1098.
12. J. Basak and D. Mahata. Connectionist Model for Corner Detection in Binary and Gray Images. IEEE Trans. on Neural Net. 11 (5) (2000) 1124-32.

13. H. Sánchez-Cruz; R. M. Rodríguez-Dagnino. Compressing bi-level images by means of a 3-bit chain code. Optical Engineering. SPIE. 44 (9) (2005) pp 1-8. 097004.
14. E. Bribiesca. A chain code for representing 3D curves. Pattern Recognition. 33(5)(2000),755-765.
15. Yong Kui Liu; Boruk Zalik. An efficient chain code with Huffman coding. Pattern Recognition 38 (4) (2005) 553-557.