## 3.5 Differential Chain Coding

Chain code proposed by Freeman [18] in 1974, is an effective representation for arbitrary curves. Typically, this representation is based on the four or eight-directional encoding where the direction of each segment is coded using a numbering scheme given in Figure 3.9.

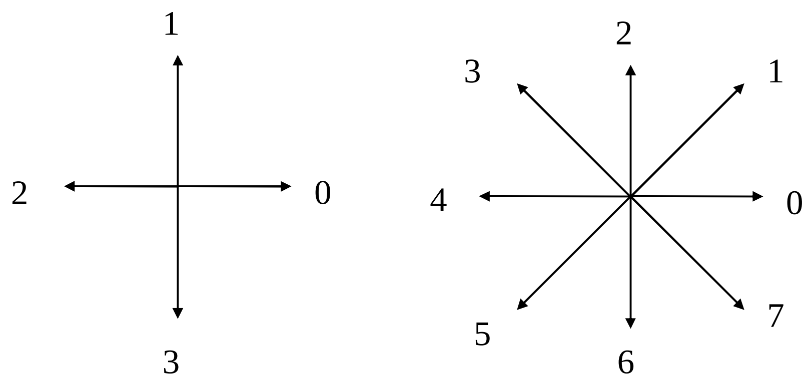


**Figure 3.9** 4 and 8-connected directions

The chain code is obtained by following a boundary in a direction explained in Section 3.3 -Edge Following, and assigning a direction to each segment.
A chain code is defined by

$$C = \{c_i \mid i = 1, \ldots, N\} \tag{3.4}$$

where $c_i$ represents a direction according to the eight-directional chain code given in Figure 3.9 and N is the number of edge segments. Let $e_i = c_i - c_{i-1}$, for i=2,...,N be the difference in direction. Then the chain difference is defined by

$$d_i = \begin{cases} e_i + 8 & e_i < -3 \\ e_i - 8 & e_i > 4 \\ e_i & \textit{else} \end{cases} \tag{3.5}$$

A differential chain code is defined by $D = \{d_i \mid i = 2,3,\ldots,N\}$. Smoothness property of curves suggests that the difference between two consecutive direction is small. The Huffman coding technique utilizes the statistics of the messages to make the most frequent symbols correspond to the shorter encoding and the rare symbols correspond to the longer encoding. Chain code does not only include the contours but also the starting point of them. Since the starting points in an edge map are distributed almost equally, they can be coded effectively in the form of distance between lexicographically ordered starting points. Another practical issue with the chain code is the need of  a codeword for termination of contours. This problem is simply solved by defining the termination as a new direction. General view of edge map with contour code is given in Figure 3.10.

| Contour $_1$ | | | | Contour $_n$ |
|---|---|---|---|---|
| Starting Point | Huffman coded differential chain code | Termination Code | | Termination Code |

**Figure 3.10** Huffman coded differential chain code format

Huffman coding makes two passes on the data stream. In the fist pass, frequency distribution of  the data stream is obtained. Huffman coding scheme constructs a binary code book tree which is known to be uniquely decodable and optimal. In the second pass, each code is assigned to an appropriate codeword  using the code book such that less probable messages are assigned to longer codewords and more probable messages are assigned to shorter codewords. Since the difference in direction accommodate in the interval (-1..1), there is no need to send the codebook.

It is implicitly assumed to be as in Figure 3.11. Experimental results given in Table (3.4,5,6) also prove the assumption.

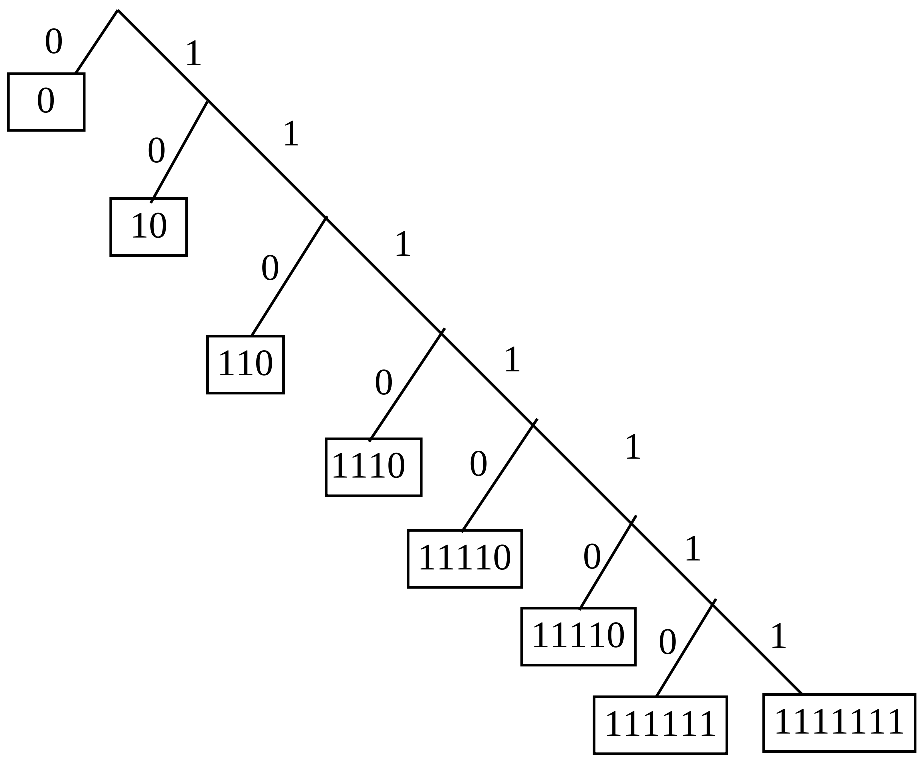


**Figure 3.11** Code Book for Differential Chain

CR stands for Compression Ratio, ACL stands for Average Code Length, and ECL stands for End point Code Length in Table 3.5, Table 3.6, Table 3.7.

**Table 3.5** Contour coding results for BRAIN.HIPS
$(w_{\text{length}} = -1.0,\ \ w_{\text{contrast}} = -1.0)$

| Contour | Edge | %Threshold | CR | ACL | ECL |
|---|---|---|---|---|---|
| 339 | 4472 | 100 | 18.22:1 | 2.06 | 12 |
| 236 | 3998 | 75 | 21.7:1 | 2.00 | 13 |
| 164 | 3506 | 50 | 26.4:1 | 1.95 | 14 |
| 84 | 2855 | 25 | 36.5:1 | 1.88 | 15 |
| 36 | 1982 | 10 | 58.25:1 | 1.83 | 15 |

**Table 3.6** Contour coding results for LENNA.HIPS
$(w_{\text{length}} = -1.0,\ \ w_{\text{contrast}} = -1.0, w_{\text{curvature}} = 0.5)$

| Contour | Edge | %Threshold | CR | ACL | ECL |
|---|---|---|---|---|---|
| 232 | 2754 | 100 | 17.1:1 | 1.973 | 9 |
| 165 | 2494 | 75 | 20.0:1 | 1.926 | 10 |
| 110 | 2199 | 50 | 24.6:1 | 1.88 | 10 |
| 56 | 1872 | 20 | 35.5:1 | 1.81 | 11 |

**Table 3.7** Contour coding results for HOUSE.HIPS
$(w_{\text{length}} = -1.0,\ \ w_{\text{contrast}} = -1.0)$

| Contour | Edge | %Threshold | CR | ACL | ECL |
|---|---|---|---|---|---|
| 635 | 6626 | 100 | 23.6:1 | 2.01 | 13 |
| 475 | 6094 | 75 | 28.1:1 | 1.96 | 13 |
| 317 | 5336 | 50 | 35.1:1 | 1.89 | 14 |
| 160 | 4150 | 25 | 52.5:1 | 1.77 | 15 |
| 97 | 3466 | 20 | 71.5:1 | 1.49 | 13 |
| 64 | 3102 | 10 | 86.1:1 | 1.65 | 13 |
| 32 | 2663 | 5 | 109:1 | 1.61 | 14 |