The Mapping Algorithm of Rectangular Vertex Chain Code from Thinned Binary Image

[1]Lili Ayu Wulandhari, [2]Habibolah Haron , [3]Ariffin Mohammad

Department of Modeling and Industrial Computing
Faculty of Computer Science and Information Systems
Universiti Teknologi Malaysia
81310 UTM Skudai, Johor Bahru, Malaysia
email: [1]lili.wulandhari@gmail.com,[2] habib@utm.my

Abstract

Image representation always becomes an important and interesting topic in image processing and pattern recognition. Since introduced by Freeman in 1961, that is known as Freeman Chain Code (FCC) the evolution and improvement of chain code representation scheme has been widely used as a topic of research. In 1999, Bribiesca introduced a new two-dimensional chain code scheme that is called Vertex Chain Code (VCC). An element of this chain indicates the number of cell vertices, which are in touch with the bounding contour of the shape in that element position. The VCC is composed of three regular cells, namely rectangular, triangular, and hexagonal. This paper covers one of the VCC cells, namely Rectangular ï VCC cell. An algorithm is developed to visualize a thinned binary image into rectangular cells and to transcribe the cells into Vertex Chain Code. The algorithm has been tested and validated in visualizing thinned binary images into rectangular-VCC cells and transcribing the cells into VCC by using three thinned binary images, which are cube, stair and rectangle. The results show that this algorithm is capable to visualize and transcribe them into VCC, and it also can be improved by testing on more thinned binary images.

Keyword: Vertex Chain Code, Rectangular Cells, Thinned Binary Image, Picture Description language

1. Introduction

The first approach for representing an image using chain code was introduced by Freeman in 1961 that is known as Freeman Chain Code (FCC) [1]. This code follows the contour in counter clockwise manner and keeps track of the directions as we go from one contour pixel to the next. The codes involve 4ï connected and 8ï connected paths. Figure 1(a) shows 4-connected and Figure 1(b) shows 8-connected of FCC

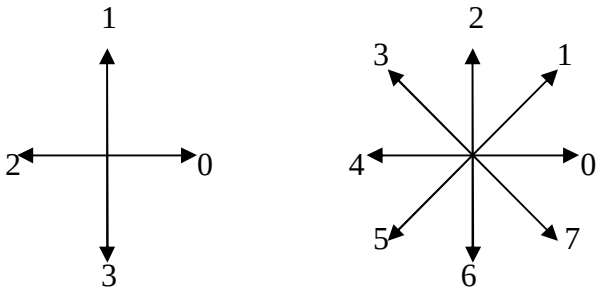


Figure 1. Neighbour Directions of FCC

In the 8-connected FCC, each code can be considered as the angular direction, in multiples of 45^0 that we must move to go from one contour pixel to the next. Figure 2 shows the example of Freeman Chain Code using 8-connected path.

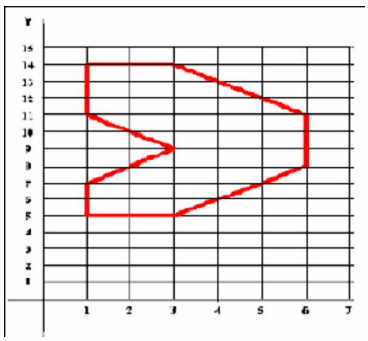


Figure 2. Example of Freeman Chain Code
Code Start from (1,5):00111222333446667755 66

Freeman [2] states that in general, a coding scheme for line structures must satisfy three objectives: (1) it must faithfully preserve the information of interest; (2) it must permit compact storage and convenient for display; and (3) it must facilitate any required processing. The three objectives are somewhat in conflict with each other, and any code necessarily involves a compromise among them.

Bribiesca [3] introduced Vertex Chain Code (VCC) in 1999. And this chain code complied with three objectives that Freeman proposed. Some important characteristic of the VCC are: (1) The VCC is invariant under translation and rotation and optionally may be invariant under starting point and mirroring transformation. (2) Using the VCC it is possible to represent shapes composed of triangular, rectangular, and hexagonal cells (Figure 3). (3) The chain elements represent real values not symbol such as other chain code, are part of the shape, indicate the number cell vertices of the contour nodes, may be operated for extracting interesting shape properties. (4) Using VCC it is possible to obtain relations between contour and interior of the shape.

In the Vertex Chain Code, the boundaries or contours of any discrete shape that are composed of regular cells can be represented by chains. Therefore, these chains represent closed boundaries. The minimum perimeter of closed boundary corresponds to the shape composed only of one cell. An element of a chain indicates the number of cell vertices, which are in touch with the bounding contour of the shape in that element position [3]. Figure 4 shows Vertex Chain Code of Rectangular ï VCC cells which indicate the number of cell vertices, which are in touch with the bounding contour of the rectangle in that element position.

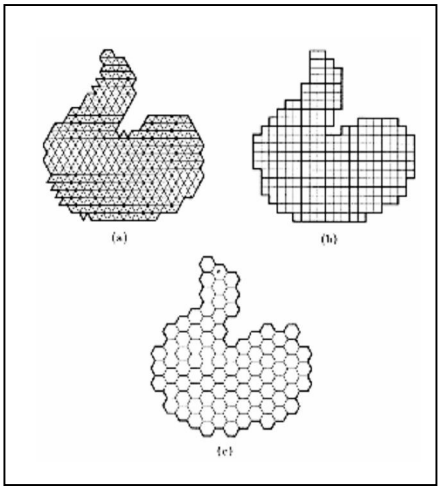


Figure 3. Example of VCC Cells:
(a) Triangular cell (b) Rectangular cell, and (c) Hexagonal cell

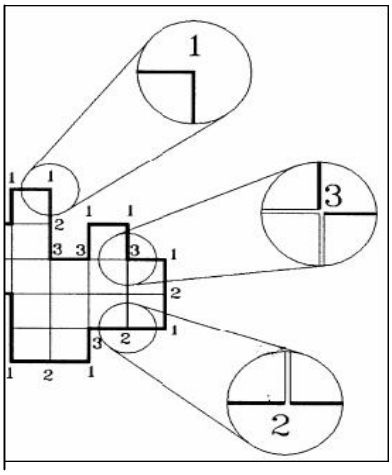


Figure 4. The Example of Rectangular Cells-VCC

This paper presents an algorithm that is used to derive the rectangular cells of VCC from a thinned binary image, and then transcribe the cells into Vertex Chain Code. The algorithm is tested and validated by using three thinned binary images, namely cube, stair, and rectangle.

2. The Mapping Algorithm of Rectangular-VCC

The mapping algorithm of rectangular-VCC consisted of three processes that are pre-processing, visualizing and transcribing process and is shown in Figure 5. Pre-processing process is the step of thinning a binary image into thinned binary image. Then the thinned binary image is visualized into rectangular-VCC cells that is called visualizing process. The last process is to transcribe the rectangular-VCC cells into Vertex Chain Code, that is called transcribing process.

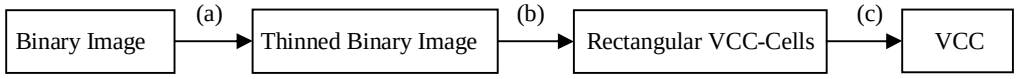


Figure 5. Flow of the mapping algorithm

2.1 The Pre-Processing

This algorithm use thinned binary image as input. Binary images are images whose pixels have only two possible intensity values. They are normally displayed as black and white. Numerically, the two values are often 0 for black and, either 1 or 255 for white. In the simplest case, an image may consist of a single object or several separated objects or relatively high intensity. This allows figure separation by thresholding. In order to create the twoï valued binary image, a simple threshold may be applied so that all the pixels in the image plane are classified into object and background pixels. A binary image function can then be constructed such that pixels above the threshold are foreground (ñ1ò) and below the threshold are background (ñ0ò) (Figure 6).

For several purposes a binary image need to be thinned. Thinned binary image is a binary image which the width is reduced to just a single pixel (Figure 7). Thinning process [4] is an important pre-processing step in pattern analysis because it reduces memory of requirement for storing the essential structural information presented in pattern. For this purpose, the thinning algorithm that is created by Haron [5] [6] is applied. This thinning algorithm uses two-valued connectivity rules. The pixel of 1 will be replaced by pixel 0 when the number of pixel 1 of the neighbourhood eight direction of connectivity pixel is greater than 3.

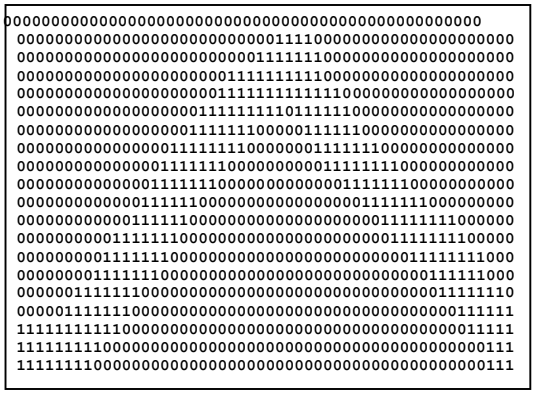


Figure 6. Binary Image

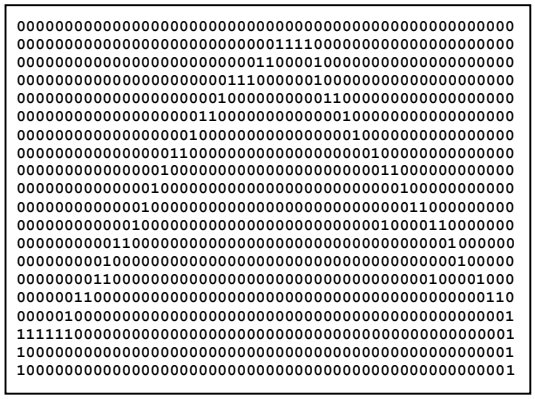


Figure 7. Thinned Binary Image

In this algorithm, every element of thinned binary image is declared as array variables. Then all of the operations of the images are according to its rows and columns

2.2 The Visualizing Algorithm

The visualizing algorithm of rectangular-VCC is an algorithm that visualizes a thinned binary image into its rectangular cells. The algorithm has two-valued connectivity thinned binary images as input. Each code 1 in the thinned binary image represents each form of the rectangular cell. The direction of code 1 that adjacent to the other code 1 leads to the formation of next rectangle. Figure 8 shows the representation of Rectangular-VCC that is formatted by the direction of code 1 that adjacent to the other code 1. When each code in binary image is visualized, a line drawing that consisted of rectangle cell will be created [7][8][9].

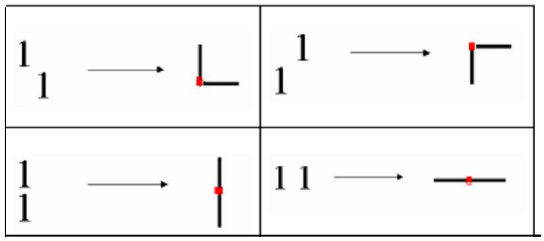


Figure 8. Representation of Rectangular ï VCC

The algorithm considers eight directions of code 1 that adjacent to the others. Each code 1 is compared with other eight directions. Figure 8 shows eight direction connectivity that is used in this paper.

(row+1, column-1)	(row+1, column)	(row+1, colum+1)
(row, column-1)	(row, column)	(row, column+1)
(row-1, column-1)	(row-1, column)	(row-1, column+1)

Figure 9. Eight direction connectivity

Every code 1 fills one rectangle that is the length 1. In this algorithm, every horizontal line is drawn from the left to the right, and every vertical line is drawn from the bottom to the top. Based on these rules, the visualizing algorithm of rectangular VCC is created.

The Visualizing Algorithm: to visualize thinned binary image into rectangular cells

```
Input = thinned binary image
image ≠ 0
for row = 1 to row = maxrow
      for column = 1 to column = maxcolum
            if image (row,column)= 1 then
                column _A   = column+1
                 row_A       =  row +1
                 column_B   = column -1
                 row_B       =  row – 1
                 if image (row,  column_A)  = 1 then
                       for x = column to x<=column_A
                            y = row
                           draw a horizontal line whose length = 1 from coordinate (x,y)
                        end for
                     end if
                          if  image (row_A, column)=1 then
```

```
                    for y= row to y<=row_A
                         x= column
                  draw a vertical line whose length = 1 from coordinate (x,y)
                 end for
              end if
              if image (row_A, column_A)=1 then
                  x = column_A
                  y = row_A
                 draw a vertical line whose length = 1 from coordinate (x,y)
                 x = column
                 y = row_A
                draw a horizontal line whose length = 1 from coordinate (x,y)
             end if
             if image(row, column_B) = 1 then
                 for x = column_B to x<= column
                     y = row
                    draw a horizontal line  whose length = 1 from coordinate (x,y)
                 end for
              end if
              if  image(row_A, column_B) = 1 then
                  x = column
                  y = row
                 draw a vertical line whose length = 1 from coordinate (x,y)
                 x = column_B
                 y = row_A
                draw a horizontal line whose length = 1 from coordinate (x,y)
              end if
              if image(row_B, column_B)=1 then
                 x= column_B
                 y= row
                 draw a horizontal line whose length = 1 from coordinate (x,y)
                  x = column
                  y = row
                 draw a vertical line whose length = 1 from coordinate (x,y)
            end if
            if image(row_B,column) =1 then
               for y = row_B to y<=row
                    x = column
                   draw a vertical line whose  length = 1 from coordinate (x,y)
                end for
            end if
            if image(baris_B, column_A) then
               x = column_A
               y = row_B
               draw a vertical line whose length = 1 from coordinate (x,y)
                 x = column
                 y = row
                 draw a horizontal line whose length = 1 from coordinate (x,y)
            end if
          end if
       end for
end for
```

2.3 The Transcribing Algorithm

The transcribing algorithm is an algorithm that transcribes the rectangular cells into vertex chain code. The algorithm uses eight direction connectivity. Rectangular Vertex Chain Code has three different codes, namely 1, 2, and 3. The code indicates the number of cell vertices, which are in touch with the bounding contour of the shape in that element position [3]. The algorithm focuses to the corner of each rectangle cells. Each corner of rectangle in the cells is named by A, B, C, and D (Figure 10). The algorithm covers every corner of rectangle by its own rules according to the eight direction connectivity.

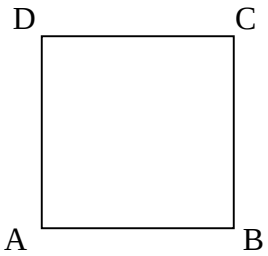


Figure 10. Rectangle in Rectangular Cells

The transcribing algorithm that is to transcribe a thinned binary image into vertex chain code is shown below.

The Transcribing Algorithm: to transcribe a thinned binary image into vertex chain code

```
Input = Rectangular-VCC and thinned binary image
image ≠ 0
for row = 1 to row = maxrow
      for column = 1 to column = maxcolum

         if corner in position A then
           if image (row,column)=1 and image (row,column_B)= 0 and image(row_B,column_B)=0 and image (row_B,column)=0
                then VCC=1
           end if
           if image (row,column)=1 and image (row,column_B)= 1 and image(row_B,column_B)=0 and image (row_B,column)=0
                then VCC = 2
           end if
           if image (row,column)=1 and    image (row_B,column)= 1 and image(row,column_B)=0 and image (row_B,column_B)=0
                then VCC = 2
           end if

           if image (row,column)=0 and image (row,column_B)= 1 and image(row_B,column)=1 and image (row_B,column_B)=0
                then VCC = 3
           end if
        end if
        if corner in position B then
          if image (row,column)=1 and image (row,column_A)= 0 and image(row_B,column)=0 and image (row_B,column_A)=0
                then VCC=1
           end if
           if image (row,column)=1 and image (row,column_A)= 1 and image(row_B,column)=0 and image (row_B,column_A)=0
                then VCC=2
           end if
          if image (row,column)=1 and           image (row_B,column)= 1 and image(row,column_A)=0 and image
                 (row_B,column_A)=0  then VCC=2
            end if
            if image (row,column)=0 and image (row,column_A)= 1 and image(row_B,column)=1 and image (row_B,column_A)=0
                then VCC = 3
           end if
       end if
        if corner in position C then
          if image (row,column)=1 and    image (row_A,column)= 0 and image(row_A,column_A)=0 and image (row,column_A)=0
                then VCC = 1
           end if
          if image (row,column)=1 and    image (row_A,column)= 1 and image(row,column_A)=0 and image (row_A,column_A)=0
                then VCC = 2
           end if
          if image (row,column)=1 and    image (row,column_A)= 1 and image(row_A,column)=0 and image (row_A,column_A)=0
                then VCC = 2
           end if
           if image (row,column)=0 and    image (row_A,column)= 1 and image(row,column_A)=1 and image (row_A,column_A)=0
                then VCC = 3
           end if
```

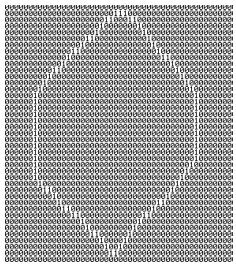
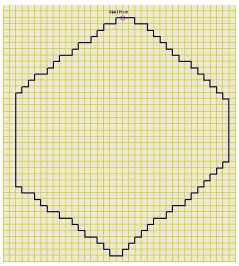
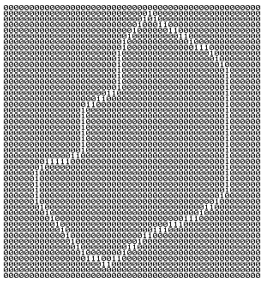
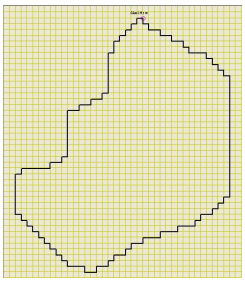
```
      end if
      if corner in position D then
        if image (row,column)=1 and    image (row_A,column_B)= 0 and image(row,column_B)=0 and image (row_A,column)=0
             then VCC = 1
        end if
        if image (row,column)=1 and    image (row_A,column)= 1 and image(row_A,column_B)=0 and image (row,column_B)=0
             then VCC = 2
        end if
        if image (row,column)=1 and    image (row,column_B)= 1 and image(row_A,column)=0 and image (row_A,column_B)=0
             then VCC = 2
        end if
       if image (row,column)=0 and    image (row_A,column)= 1 and image(row,column_B)=1 and image (row_A,column_B)=0
              then VCC = 2
        end if
      end if
    end for
  end for
```

3. Results

3.1 Experimental Results

All of these algorithms are tested and validated by using three thinned binary images, namely cube, stair, and rectangle.

Figure 11 shows the result of the experiment that use the visualizing and transcribing algorithm. Thinned binary image is visualized into rectangular-VCC by using Visualizing Algorithm, and rectangular-VCC is transcribed into Vertex Chain Code by using Transcribing Algorithm. Both of this algorithm is called Mapping Algorithm

No.	Thinned Binary Image	Rectangular-VCC Cells	Vertex Chain Code (VCC)
a.			2213213131313131321313 1313131312222222221313 1313131313123131231313 1313131213131321313132 1321313132131222222222 2222213131312313131231 312313131231
b.			1313213213213132213131 3131222222222222222222 2213131312313122312231 2231231312313123121322 1131313131313131313222 2221312222123132222222 1231231231322222213213 1313131

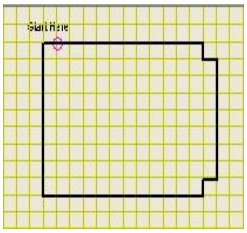
c.			22222222222131222222213 12222222222212222222221

Figure 11. Rectangular-VCC Cells and Vertex Chain Code of Three Thinned Binary Image
(a) Cube, (b) Stair, (c) Rectangle

3.2 The Interface

Testing and validation of the algorithm are performed by developing a prototype system using Visual Studio 6 programming language. Figure 12 shows the interface of the system.

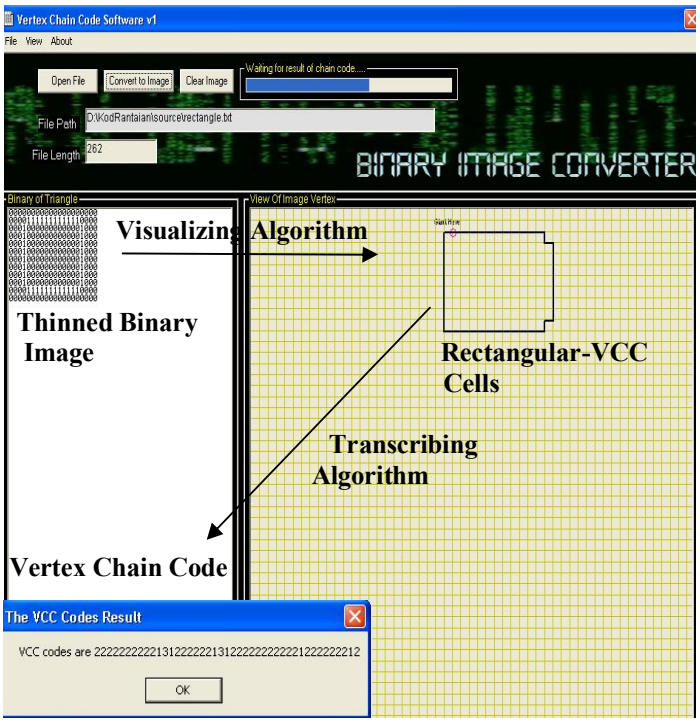


Figure 12. The Interface of Prototype System

The interface shown in Figure 12 consists of three processes in the mapping algorithm. The input is a thinned binary image. A thinned binary image is then visualized into rectangular-VCC cells by pressing the *convert to image* button. Then the process is continued to transcribe the rectangular-VCC cells into Vertex Chain Code. The code will be displayed automatically when the process finished.

4. Conclusion

The mapping algorithm has been tested and validated in visualizing and transcribing thinned binary images into VCC by using three thinned binary image objects that are cube, stair, and rectangle. The results show that the visualizing algorithm capable to visualize thinned binary image into rectangular-VCC cells. Reciprocally the transcribing algorithm also capable to transcribe the rectangular-VCC cells into Vertex Chain Code. Both of this algorithm is called The Mapping Algorithm of Rectangular Vertex Chain Code

Acknowledgement

The authors honourably appreciate Ministry of Science, Technology and Innovation (MOSTI) for the FRGS grant and Research Management Center (RMC), University of Technology Malaysia (UTM) for the support in making this projects success

5. References

1. Freeman H, On The Encoding of Arbitrary Geometric Configurations. IRE Trans EC-10 (2), 1961, 260-268
2. Freeman H, Computer Processing of Line-Drawing Images, ACM Computing Surveys 6, 1974, 57-97
3. Bribiesca E, A New Chain Code. Pattern Recognition, Vol. 32, Issue 2, 1999, 235-251
4. Wing-nin Leung, C M Ng, and P C Yu, Contour Following parallel Thinning for Simple Binary Images, 2000 IEEE International Conference, Systems, Man, and Cybernatics, Vol. 3, 2000, 1650-1655
5. Habibollah Haron, Dzulkifli Mohamed, and Siti Mariyam Shamsuddin, Extraction of Junction, Lines, and Regions of Irregular Line Drawing: The Chain Code Processing Algorithm, Jurnal Teknologi No. 38(D), Penerbit UTM, 2003, 1-28
6. Habibollah Haron, Dzulkifli Mohamad, and Siti Mariyam Shamsuddin, Enhancement of Thinning Algorithm For 2D Line Drawing of 3D objects, Journal of Mathematics & Computer Sciences (Computer Science Series), India, December 2003, 169-174
7. Habibollah Haron, Siti Mariyam Shamsuddin, and Dzulkifli Mohamed, Corner Detection of Chain-Coded Representation of Thinned Binary Image, International Journal of Computer Mathematics, U.K., Vol. 82 No 8, August 2005, 941-950
8. Syarul Haniz Subri, Analisis Rangkaian Neural Dalam Pengesanan Simpang Bagi Penterjemah Lakaran Pintar Application of Neural Network in Corner Detection of Intelligence Sketch Interpreter, MSc Thesis, Universiti Teknologi Malaysia, 2006
9. Syarul Haniz Subri, Habibollah Haron, and Roselina Sallehuddin , Neural Network Corner Detection of Vertex Chain Code, ICGST International Journal on Artificial Intelligence and Machine Learning (AIML), Vol. (6) Issue(1), January 2006, 37-43.