A Fast Algorithm of Neighborhood Coding and Operations in Neighborhood Coding Image

by
Anke CEN*, Chengxun WANG** and Hiromitsu HAMA***
(Received Sept. 30, 1995)

Synopsis
In this paper, a fast algorithm for neighborhood coding of binary images is described. The fast algorithm for border tracing is improved by labeling the traced border points in the coding space. The neighborhood coding is combined with mathematical morphology, then give the possibility to extends the applications of the neighborhood coding to more than $3 \times 3$ windows.

Keywords: Neighborhood coding, border tracing, Mathematical Morphology

1. Introduction
In binary image processing, we often encounter the $3 \times 3$ window processing. This is essentially a parallel operation. If running on sequential computers, it is very time consume. Sobel suggested a neighborhood coding (NC) for binary images [1]. Then the $3 \times 3$ window operation can be performed by use lookup tables [2], that is much faster than the normal operations. But, if the encoding of binary images to neighborhood coded images (NCI) is done with an ordinary sequential computer, most of this advantage is lost since the encoding process entails accessing all eight neighbors of a pixel. To solve this problem, Sobel proposed the use of a simple hardware in one pass through the image array. With the use of a hardware encoder, although the speed is high, the disadvantage is of less flexible because the encoding hardware is not available on the market and the image size can not change easily. If look into carefully the encoding process, one can find when one pixel is to be encoded, the most information is already included in its neighbors which are encoded previously. It is not necessary to access all the eight neighbors for encoding this pixel. then the algorithm thus developed can be much more efficient.

In section 2, we propose an algorithm for fast neighborhood encoding. In section 3, we improve the border tracing algorithm in the $NC$ space. The algorithm for boundary following in neighborhood space given by Sobel can obtain Freeman code [3] directly from

---

* Student, Doctor Course of Dept. of Information and Computer Engineering, Osaka City Univ.

** Professor, Dept. of Electronics Engineering, Dalian Univ. of Technology.

*** Professor, Dept. of Information and Computer Engineering, Osaka City Univ.
using lookup tables, and speed up the operation. However, the mark problem has not been solved in that algorithm, so that repeated tracing may be unavoidable in some complicated cases. By using the crack boundary conception \cite{4}, we developed a method for marking the traced boundary in the coded space. In section 4, we use NC in mathematical morphology \cite{5} \cite{6} operations so that it may be extended to more than $3 \times 3$ windows operation, by structure decomposition.

2. Principle of the fast algorithm for NC

Sobel's neighborhood coding defines as Fig. 1. The code of pixel $P$ is expressed by its eight neighbors in a $3 \times 3$ window in the order as shown in Fig. 1 forming one byte ($NB$) of number. The ordinary coding should access memories eight times corresponding to the eight neighbors for a pixel. Besides, the result must put in another memory space as big as the original image space. But, looking carefully, one can find when one pixel is being encoded, much information can be obtained from its neighbors already encoded. This is shown in Fig. 2 where $NB_1, NB_2, NB_3$ and $NB_4$ are the neighbors of $P$, which are already encoded. The code of $P$ can be obtained only from $NB_2, NB_4$ and $b_7$. $NB_4$ is the code obtained immediately before and can be retained in the register. Therefore, the coding needs accessing the memories only twice for getting $NB_2$ and $b_7$, and some logical shift operations. The resulting value can be put in the position of the $P$ directly and this will not affects the encoding of the following pixels. Using this method, we developed a fast neighborhood coding algorithm which is $\tau$ times faster than the ordinary one,

$$\tau = \frac{8TR + TW + TS}{2TR + TW + TS}$$

where $\tau$ is operating speed ratio of the fast algorithm and the ordinary one, $TR$ and $TW$ are the access times of reading and writing memory, $TS$ and $TP$ the the time taken by the logic operations for CPU to form a $NB$ in the fast algorithm and the ordinary one, respectively. Because the values of $TS$ and $TP$ are much smaller than $TR$ and $TW$ which can be considered as the same values, the ratio $\tau$ take a value about 2 to 3. Another feature of the algorithm is that it needs no additional image memory space for forming a NC. The reconstruction of the image from the coded images is very simple. See Fig. 3, assuming that the reconstruction scanning already reaches to a pixel $P$, the value of the pixel $P$ in the original image is expressed by the fourth bit of $NB_0$.

\[\begin{array}{ccc}
  b_3 & b_2 & b_1 \\
  b_4 & P & b_0 \\
  b_5 & b_6 & b_7 \\
\end{array}\]

Fig. 1. Sobel's neighborhood coding.
A Fast Algorithm of Neighborhood Coding and Operations in Neighborhood Coding Image

Fig. 2. Pattern of reaching to pixel \( P \). \( NB_1, NB_2, NB_3 \) and \( NB_4 \) have been encoded.

Fig. 3. Reconstruction pattern of reaching the pixel \( P \).

The encoding procedure is shown in Fig. 4. \( NB_2 \) and \( b_7 \) are in the memory which must be accessed. \( NB_4 \) is retained in a register which is used to store the result of the coding. It is much faster to access the register than the memory. Recombining the bits of these memories and register shown in Fig. 4, one obtains the code value of pixel \( P \). The algorithm is easily implemented with assembler language.

Fig. 4. The procedure of forming \( NB_p \). Where, \( NB_2 \) and \( b_7 \) are in the memory which must be accessed, \( NB_4 \) is retained in a register which is used to store the result of the coding.

3. The labeling of the traced border points in \( NC \) space

Assume the object of the image \( F \) is denoted by \( s, s \subseteq F \), and its border by \( s', s' \subseteq s \); the background is \( \neg s \subseteq F \); the value in \( s \) and \( \neg s \) are 1 and 0, respectively; \( s' \) is an eight-connected curve. During the border point labeling, the neighborhood bytes (\( NB \)) of the border points are modified. This is as if when border is traced along the crack between
s and !'s keeping s to the left of the crack border as shown in Fig. 5. The labeling fills up the passed cracks to prevent repeated tracing. To do this, the points (€!s) on the right side of the passed crack are set to thin 1 shown in Fig. 5. The labeling is done in the neighborhood code space and it does not affect the neighborhood points. Let $C_0$ and $C_1$ be the direction chain code pointing to $P$ and one from $P$ to next border point, respectively. The values of the four-connected neighborhood points (€!s) of $P$ on the right side of the border are set to thin 1. Fig. 6 shows some examples of the modifications of border points. Where $NB_P$ is the neighborhood code of the point $P$, expressed in hexadecimal. The symbol $\phi$ in Fig. 6 denotes 0 or 1. Note that the modifying process is only an OR operation between the original $NB_P$ and a modifier obtained from a lookup table $LABTAB(C_0, C_1)$, where $C_0$, $C_1 = 0, 1, ..., 7$. Because all labeling process is accomplished by looking up tables and logical operations, the program runs fast. The algorithm is described as following:

$$
C_0C_1 = 31h \quad \Rightarrow \quad Nb_P = Nb_P \cdot OR \cdot 01h
$$

$$
C_0C_1 = 35h \quad \Rightarrow \quad Nb_P = Nb_P \cdot OR \cdot 15h
$$

$$
C_0C_1 = 37h \quad \Rightarrow \quad Nb_P = Nb_P \cdot OR \cdot 55h
$$

Fig. 5. Directions of border following. Points (€!s) on the right side of the passed crack are set to thin 1.

Fig. 6. Examples of the modifications of border points. Where $NB_P$ is the neighborhood code of the point $P$, expressed in hexadecimal.
3.1 Determination of starting points

The feature patterns of hole border and outer border are used to decide the starting points of a hole border and outer border. The feature pattern of the outer border starting point is $01 \varphi$, and $110$ for the hole border. They all take the left border point for the starting point. These points are the first met points during scanning. One can decided starting border points as follows:

$$p = 1,$$  \hspace{1cm} (2)

$$NB_p \cdot AND \cdot 11h = 10 \varphi_h p \text{ is a starting outer \;} B P(Border \; Point);$$

$$11 \varphi_h p \text{ is not a starting \;} B P$$  \hspace{1cm} (3)

Because all border starting points are the left border points, the algorithm cannot take the right points of a hole for the points of an outer border, and vise versa.

3.2 Check the starting chain code direction

The starting direction of the chain code means that the direction pointing from the last point to the starting border point. The importance is that only in a correct starting direction can a correct border be traced from the starting point. For the direction of hole border starting points, the only difference is that all value $0$ in the pattern must be on the right side of the border. The direction of the chain code is determined by looking up the table:

$$i_{dir} = ID I R T B(NB p \varphi)$$  \hspace{1cm} (4)

3.3 Border tracing

From the direction of the chain code $C_{i+1}$ in which the preceding border point $P_{i-1}$ pointing to the current border point $P_i$ and the $NB_p$ of the current border point, the value of the chain code $C_i$ of the next border point can be obtained by looking up the two dimensional table:

$$C_i = D I R T A B(NB p_i, C_{i-1})$$  \hspace{1cm} (5)

Then the current border point $P_i$ is updated to $P_{i+1}$ by changing a pointer pointing to $P_i$ to point to the next border point $P_{i+1}$,

$$CP_{i+1} = CP_i + \delta_i$$  \hspace{1cm} (6)
Where $CP_i$ and $CP_{i+1}$ are the pointers pointing to $P_i$ and $P_{i+1}$ respectively, and is $\delta_i$ a displacement between $P_i$ and $P_{i+1}$, which can be obtained from the table:

$$\delta_i = DPIX(P_i)$$

(7)

So, the search for each point and the chain code can be obtained only by looking up tables twice.

3.4 Labeling of the border points

We have described the labeling already in Fig. 5 and Fig. 6, that using the table $LABTAB(C_{i-1}, C_i)$, we can obtain the value which should be $ORB$ with $NB_P$.

3.5 Ending of border tracing

The tracing ends when the next border point is the starting border point and the direction of the chain code pointing to the next point equals to the starting direction of the chain code. As shown in Fig. 7, border tracing does not end until the returning direction of the chain code coincides with the starting direction of the chain code. The experiment result of border tracing is shown in Fig. 8. The image size is 256×256 and the running time is about 0.43 seconds on IBM PC-AT.

Fig. 7. Border tracing processing, which does not end until the returning direction of the chain code coincides with the starting direction of the chain code.

Fig. 8. The experimental result of border tracing.
4. Mathematical morphological operations in NCI

The mathematical morphology is a powerful tool for digital image analysis \(^7\) \(^8\), because the algorithm is essentially parallel. If running on sequential computers, it is time consumed. If the structure pattern is limited in an 3×3 window and the reference point which is in the center of the window is 1, it can be expressed with one byte \(PB\), then one can do the simple logical operation for \(NB\) and \(PB\) to obtain the result in the window area, so speed up the operation. We define \(PB\) is the neighborhood code for the center point, \(i.e.\ PB = \text{NBCx}\). For example:

\[
\begin{align*}
B = \cdot x \cdot \quad \text{and} \quad B = x \cdot \\
\end{align*}
\]

then \(PB = 55h\) and \(01h\), respectively. The algorithm for neighborhood coding erosion (NCE) processed as following: (1) Check if the original state of the current point equals 1 (this can be done by checking the neighborhood code of one of its neighborhood point); (2) if the \(NB\) of current point contains \(PB\), \(i.e.\ (NB \cdot AND \cdot PB) = PB\)?

If (1) and (2) hold simultaneously the result for that point is 1, and 0, vice versa. In practice, neighborhood coding and pattern operation can be combined together to accomplish. There for the operation can being accomplished by only one scanning of the image. This further speeds up the operation. The method is after using the fast algorithm for neighborhood coding which is mentioned in section 2, perform operation with \(PB\) immediately. It should be noted that if the result is put in point \(P\), the result of the next coding will be error. The problem can be solved if we put the result in another memory unit for the use of the coding for next row, then perform the erosion operation. This memory unit can be selected at lower right corner \(b_7 \cdot b_7\) is already used during the producing of \(NB_p\), and not used afterwards. Therefore, during the coding in the next row \(NB_{x}\) to be used is at \(N\)-neighbor instead of \(E\)-neighbor. The algorithm for neighborhood coding dilate (NCD) is basically same as NCE. The only difference is the logical operation is \((NB \cdot AND \cdot PB) = 0\)? if it holds, the result is 1, and 0, vice versa. The opening and closing operations \(^9\):

\[
\begin{align*}
X \cdot B &= (X \ominus B) \oplus B \\
X \oplus B &= (X \ominus B) \ominus B
\end{align*}
\]

The \(PB\) which corresponds to \(B\) can be obtained by cyclic shifting left or right form \(PB\). For comparison, we give out the experiment results for using both direct and coding methods shown in Table 1. In Table 1, \(D\) is for direct method and \(C\) is for coding method.

From the table, we can see with the increasing size of the pattern, the speed for the direct method slows down, but for the coding method, the speed remains unchanged. When the structure pattern \(PB\) size is greater than 3×3, one can decompose the pattern into some smaller ones which are within the 3×3 windows, and the NCE method still can be used.
Table 1. Comparison of the algorithms for using both neighborhood coding and direct method

<table>
<thead>
<tr>
<th>pattern</th>
<th>PB</th>
<th>erosion</th>
<th>dilate</th>
<th>open</th>
<th>close</th>
</tr>
</thead>
<tbody>
<tr>
<td>• • •</td>
<td>$D$</td>
<td>$C$</td>
<td>$D$</td>
<td>$C$</td>
<td>$D$</td>
</tr>
<tr>
<td>• $x$ •</td>
<td>$FF$</td>
<td>1.92, 1.26</td>
<td>1.81, 1.26</td>
<td>3.67, 2.58</td>
<td>3.67, 2.58</td>
</tr>
<tr>
<td>• • •</td>
<td>$D$</td>
<td>$C$</td>
<td>$D$</td>
<td>$C$</td>
<td>$D$</td>
</tr>
<tr>
<td>• $x$ •</td>
<td>55</td>
<td>1.64, 1.26</td>
<td>1.59, 1.26</td>
<td>3.23, 2.58</td>
<td>3.23, 2.58</td>
</tr>
<tr>
<td>•</td>
<td>$D$</td>
<td>$C$</td>
<td>$D$</td>
<td>$C$</td>
<td>$D$</td>
</tr>
<tr>
<td>• $x$ •</td>
<td>01</td>
<td>1.53, 1.26</td>
<td>1.48, 1.26</td>
<td>3.01, 2.58</td>
<td>3.01, 2.58</td>
</tr>
</tbody>
</table>

5. Summary

We have given a fast algorithm for neighborhood coding, and a method for labeling the traced border points in the coding space, and improved the fast algorithm for border tracing using neighborhood coding. Finally, we combine the neighborhood coding and mathematical morphological in operations into one algorithm, it shows a possibility to extending the applications of the neighborhood coding to more than $3 \times 3$ windows.

Reference