PERGAMON

Computers & Graphics 26 (2002) 401–408

COMPUTERS & GRAPHICS

www.elsevier.com/locate/cag

Best Papers of CAD & CG 2001

BRDC: binary representation of displacement code for line

Lanfang Miao[a,b], Xinguo Liu[a], Qunsheng Peng[a,*], Hujun Bao[a]

[a] *State Key Lab. of CAD&CG, Zhejiang Univerisity, Hangzhou 310027, China*
[b] *Department of Computer Science, Zhejiang Normal University, Jinhua 321004, China*

Abstract

In raster graphics, a line is displayed as a sequence of connected pixels that best approximate the line with minimum deviation. The *displacement code* of a line is a sequence of binary codes, each of which represents the displacement of a pixel on the line to its immediate predecessor pixel on the line. In fact, the displacement code records the entire process of drawing a line with successive pixels and it is deterministic for each specific line. In this paper, we study the important properties of the binary representation of *displacement code*, called BRDC, including calculation formula, periodicity, complement, decomposition etc. At last, we put forward an efficient adaptive multi-pixel line drawing algorithm based on exploited properties of BRDC, which demonstrates that BRDC is significant for designing efficient line drawing algorithms.

Keywords: Raster display devices; Line and curve generation; Display algorithms; Displacement code

1. Introduction

Line is one of the most fundamental elements in computer graphics. In raster graphics, a line is displayed as a sequence of connected pixels, which best approximate the line with minimum deviation. Many efforts have been paid to developing efficient line drawing algorithms [1]. These algorithms can be classified into two types. One type relates to single-step algorithms, which generate one pixel on the line during each iteration. The other relates to multi-step algorithms, which generate multiple successive pixels on the line during each iteration. One of the best-known single-step algorithms is the Bresenham algorithm [2] proposed by Bresenham in 1965, which involves only arithmetic operations of integers, i.e., one integer addition and one sign judgment for each pixel's generation. Because of its simplicity, it is well-suited for hardware implementation. Many attempts [3–8] later have been tried to improve the efficiency of the Bresenham algorithm. One way to accelerate the line drawing process is generating multiple pixels on the line during each iteration [9–13]. Taking advantage of the symmetry of a line from either endpoint, the bi-directional method is proposed in [9] to draw a line from both directions. The double-step algorithm in [10] generates two pixels each time according to the line's slope. Also in [11] a triple-step approach is proposed. Bao et al. analyzed all the possible configurations of four pixels, called the quadruple-step running codes of line, and proposed a Quad-step algorithm in [12]. A general N-step method proposed in [13] considers even larger steps to accelerate the process of line drawing. In these multi-step algorithms, a pre-determined fixed number, for example 2, 3, or 4 etc, of pixels are generated at each iteration. In addition, there are some other adaptive multi-step algorithms [14–17], which adaptively determines the step according to the slopes of the lines. In [17], three kinds of pixel approximations of a line are discussed, and a new algorithm is given, which is 20 times faster than the original Bresenham algorithm.

After reviewing the published line drawing algorithms, we found that the entire line drawing process can be represented by the displacement codes, which record the displacement of each pixel on the line to its immediate predecessor pixel on the line. The major

*Corresponding author. Tel.: +86-571-87951045; fax: +86-571-87951780.

E-mail address: peng@cad.zju.edu.cn (Q. Peng).

0097-8493/02/$ - see front matter
PII: S0097-8493(02)00083-3

difference between various line drawing algorithms is how they determine the displacement code. The binary representation of displacement code (BRDC) is interesting for addressing line drawing problem, because: (1) For a given line, its BRDC is unique, and (2) the BRDC of a line can be determined prior according to its slope. (3) BRDC has lots of nice properties, which can be employed to improve the line drawing process. In the next section of this paper, we will first give a formal definition of BRDC. Then we examine the important properties of the BRDC, including calculation formula, periodicity, decomposition, etc. in Sections 3 and 4. Then in Section 5, we put forward an adaptive multi-pixel line drawing algorithm based on the BRDC, and we show that the BRDC is significant for designing efficient line drawing algorithms. In the last section some future work are addressed.

2. Definition of BRDC

A raster graphics display is logically a 2D grid of pixels as shown in Fig. 6. The task of drawing a line is to determine which pixels belong to the line. Due to the limited resolution of the display, it is an approximation process. For example, L is a line from $A(x_a, y_a)$ to $B(x_b, y_b)$, as shown in Fig. 1. Without loss of generality, we suppose that the slope m of line L is in the interval [0, 1]. The intersection of L and a grid line $x = x_i$ is $(x_i, Y(x_i))$, where $x_i = x_a + i$, $i = 0, 1, 2, \ldots, x_b - x_a$, and $Y(x_i) = x_a + m(x_i - x_a)$.

Obviously the nearest pixel to the intersection is (x_i, y_i), where $y_i = [Y(x_i) + 0.5]$. $[\cdot]$ is the truncation operator. Therefore, the pixel approximation of line L is $\{(x_i, y_i) | i = 0, 1, 2, \ldots, x_b - x_a\}$.

Although it is straightforward to represent a line by the coordinates of all pixels on the line, the representation is somewhat complex, since it is location-dependent. In the following, we will develop a more concise representation scheme based on the relative positions between each pair of adjacent pixels on the line. Recall that the slope m of line L is between 0 and 1, we have

$$\begin{aligned} y_{i+1} - y_i &= [Y(x_{i+1}) + 0.5] - [Y(x_i) + 0.5] \\ &= [Y(x_i) + m + 0.5] - [Y(x_i) + 0.5] \in [0, 1]. \end{aligned}$$

Therefore, $y_{i+1} - y_i$ is either 0 or 1. It is clear that the pixel approximation $\{P_i = (x_i, y_i) | i = 0, 1, 2, \ldots, x_b - x_a\}$ and the binary code sequence $\{y_{i+1} - y_i | i = 0, 1, 2, \ldots, x_b - x_a\}$ can be determined from each other. Note that no matter which value $y_{i+1} - y_i$ takes, the line moves horizontally by a unit step. And if $y_{i+1} - y_i$ is 1, the line also moves vertically by a unit step. We adopt code 1 to represent a step of diagonal movement and code 0 to represent a step of horizontal movement between two successive pixels on the line. Code 1 is referred as the *jumping code*. Therefore, a line can be represented equivalently by its *displacement code*.

It is clear that BRDC is translation invariant, i.e. if the line is translated to a new position in the pixel grid, its BRDC remains unchanged. In the rest of this paper we assume that the lines pass through the origin without loss of generality. Under this assumption, the lines lie in the first octant, and have the following form:

$$Y_{H,k}(X) = \frac{k}{H} X,$$

where H and k are two integers and $0 < k < H$. And we adopt the notations and abbreviations in Table 1 for convenience.

When $k = 1$, we can easily calculate the BRDC $A_{H,1}$ of line $Y_{H,1}$ according to the definition:

$$\underbrace{0\,0\cdots 0}_{T0s}\,1\,\underbrace{0\,0\cdots 0\,0}_{(H-1)0s}\,1\,\underbrace{0\,0\cdots 0\,0}_{(H-1)0s}\,1\,0\,0\cdots. \tag{1}$$

It is obvious that the following three claims on the above BRDC hold:

Claim 1. *$A_{H,1}$ is a* periodic *sequence, and its period is H.*

Claim 2. *There is one and only one* jumping code *in one period of $A_{H,1}$.*

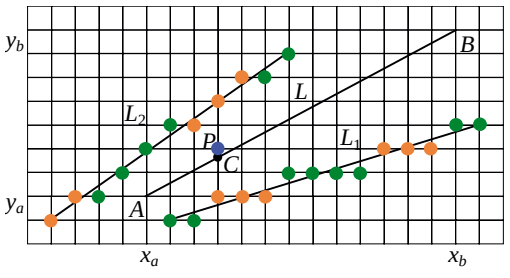

Fig. 1. Pixel (grid points) Approximation of lines. Some nearby pixels are used to approximate the lines.

Table 1
Notations and abbreviations

Notation	Meaning	Abbr.
H, k	H, k: Integers and $0 < k < H$	
g, h	$g = H\%k$, $h = k - g$	
$Y_{H,k}$	Line: $Y_{H,k}(X) = \frac{k}{H}X$	Y_k
$A_{H,k}$	BRDC of line $Y_{H,k}$	A_k
$\overline{A_{H,k}}$	Conjugation of $A_{H,k}$: $\overline{A_{H,k}}(i) = A_{H-k,k}(H-1-i)$	$\overline{A_k}$
T_H	The *jumping position* of $A_{H,1}$	T
$T_{H,k}$	$T_{H,k}(i) = (T_H - ik)\%H$	T_k
$S_{H,k}$	*Segment code* of $A_{H,k}$	S_k

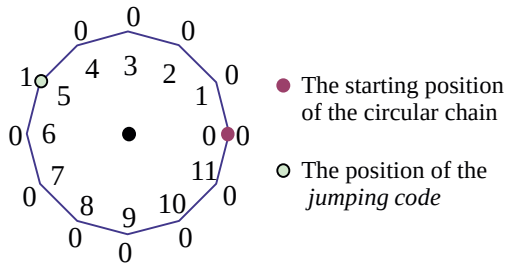

Fig. 2. Circular chain for the BRDC $A_{12,1}$ of line $Y_{12,1}(X) = \frac{1}{12}X$.

Claim 3. *In $A_{H,1}$, the number of the initial 0's is $T = \left[\frac{H-1}{2}\right]$, i.e. its first* jump code *occurs at the position $T = \left[\frac{H-1}{2}\right]$.*

The periodic BRDC in (1) can be illustrated by a circular chain defined on an H-sided equilateral polygon, as shown in Fig. 2.

However, when $k > 1$, the BRDC $A_{H,k}$ of line $Y_{H,k}$ may be somewhat difficult to calculate directly using its definition. After carefully examining some specified examples of the BRDC, we find a synthesis method to compute $A_{H,k}$, which is concluded in the following lemma.

Lemma 1. *Let H and k be two integers satisfying $0 < k < H$. Then the BRDC A_k of line Y_k can be synthesized using the BRDC A_1 of line Y_1 by the following formula*:

$$A_k(i) = \sum_{s=0}^{k-1} A_1(i \cdot k + s). \tag{2}$$

Proof. According to the definition of BRDC, we have

$$A_k(i) = y_{k,i+1} - y_{k,i} \quad \text{and} \quad A_1(i) = y_{1,i+1} - y_{1,i}.$$

Since

$$Y_k(x_i) = \frac{k}{H}x_i = \frac{1}{H}(k \cdot x_i) = Y_1(k \cdot x_i) = Y_1(x_{i \cdot k}).$$

Then

$$y_{k,i} = [Y_k(x_i) + 0.5] = [Y_1(x_{i \cdot k}) + 0.5] = y_{1,i \cdot k}.$$

$$\begin{aligned} A_k(i) &= y_{k,i+1} - y_{k,i} = y_{1,i \cdot k+k} - y_{1,i \cdot k} \\ &= \sum_{s=0}^{k-1}(y_{1,i \cdot k+s+1} - y_{1,i \cdot k+s}) = \sum_{s=0}^{k-1} A_1(i \cdot k + s). \end{aligned}$$

□

Since A_1 is a periodic sequence, A_k is also a periodic sequence according to formula (2) in Lemma 1. And the period of A_k is $H/(H,k)$ according to group theory, where (H,k) is the greatest common divisor of H and k.

For every k codes in A_1 grouped together, formula (2) implies that $A_k(i)$ is the summation of the codes in the ith group. Taking advantage of the circular chain shown in Fig. 2, we can obtain in order $A_k(0), A_k(1), A_k(2), \ldots$ more conveniently through the following method: Sum the first k codes in the circular chain to obtain one code in A_k, then rotate clockwise the circular chain by k codes. Note that there is one and only one *jumping code* in the circular chain of A_k according to Claim 2. Since k, the length of each segment, is $< H$, the whole length of the chain, the summation can be obtained by testing whether the *jumping position* of the circular chain is $< k$ or not, instead of summing up all of the elements. After i rotations, the *jumping position* of the circular chain becomes

$$T_k(i) = (T - ik)\%H. \tag{3}$$

Therefore,

$$A_k(i) = \begin{cases} 1, & T_k(i) = (T - ik)\%H < k, \\ 0 & \text{otherwise.} \end{cases} \tag{4}$$

Since $T_k(0) = T$according to Claim 3, we can obtain $A_k(i)$ and $T_k(i)$ one by one using the following test:

(a) If $T_k(i) < k$, then $A_k(i) = 1$ and $T_k(i+1) = T_k(i) - k + H$;
(b) If $T_k(i) \geqslant k$, then $A_k(i) = 0$ and $T_k(i+1) = T_k(i) - k$.

Taking (a) and (b) as the main loop control, we can easily develop a line drawing algorithm, which generates one pixel by one integer addition and one comparison, as shown in the Algorithm 1 in Fig. 6. This algorithm can also be considered as an efficient implementation of Bresenham algorithm.

3. Properties of BRDC

In this section, we study some important properties of BRDC. The first property is about the symmetry of BRDC.

Property 1. *Let H and k be two integers satisfying $0 < k < H$. If H is an odd integer, then A_k is symmetric, i.e.*

$$A_k(i) = A_k(H - 1 - i).$$

Proof. Since H is odd, and $T = H - \frac{1}{2}$, then

$$T = H - 1 - T \tag{5}$$

According to (3), there exists an integer m, such that $T_k(i) = T - ik + mH$.

If $A_k(i) = 1$, then $0 \leqslant T_k(i) < k$, i.e.

$$0 \leqslant T - ik + mH < k.$$

Substituting T with $H-1-T$, we have

$$0 \leqslant T + (i+1)k - (m+1)H < k. \quad (6)$$

So,

$$\begin{aligned} T_k(H-1-i) &= (T-(H-1-i)k)\%H \\ &= (T+(1+i)k)\%H \\ &= T+(1+i)k-(m+1)H<k. \end{aligned}$$

Then $A_k(H-1-i)=1$ according to (4). Similarly, we can prove that if $A_k(H-1-i)=1$, then $A_k(i)=1$. Therefore, $A_k(i)=A_k(H-1-i)$, i.e. A_k is symmetric. □

In fact, the symmetry property has been used in some previous work to generate pixels from bi-directions to accelerate the line drawing procedure. However, if H is an even integer, Property 1 may be not true, since $2T \neq H-1$, which is critical in the proof of Property 1. For example, $A_{6,1}(2)$ and $A_{6,1}(3)$ are not equal, since $A_{6,1}(2)=1$ and $A_{6,1}(3)=0$, as shown in Fig. 3. It shows that the previous works on bi-directional line drawing algorithms may in fact generate fault results. From the point of pixel approximation, this is due to the ambiguity of the nearest pixel when the intersection of the line and the grid line is the midpoint of its lower pixel and upper pixel, which may occur as H is an even integer.

Property 2. *Let H and k be two integers satisfying $0<k<H$. If H is an odd integer, then A_k and A_{H-k} are complementary, i.e.*

$$A_k(i) = 1 - A_{H-k}(i).$$

Proof. If $A_k(i)=1$, according to the proof of Property 1 there exists an integer m, such that (6) holds, i.e.

$$0 \leqslant T + (i+1)k - (m+1)H < k.$$

Then

$$H-k \leqslant T - i(H-k) - (m-i)H < H,$$

$$\begin{aligned} T_{H-k}(i) &= (T-i(H-k))\%H \\ &= T-i(H-k)-(m-i)H \geqslant H-k. \end{aligned}$$

So, $A_{H-k}(i)=0$ according to (4).

Similarly we can prove that, if $A_{H-k}(i)=1$, then $A_k(i)=0$. Therefore,

$$A_k(i) = 1 - A_{H-k}(i).$$

□

$A_{6,1}$	0	0	1	0	0	0	…
$A_{6,5}$	1	1	1	0	1	1	…

Fig. 3. Properties 1 and 2 are not true for $H=6$, $k=1$.

When H is an even integer, Property 2 may not be true. For example, $A_{6,1}(2)$ and $1-A_{6,5}(2)$ are not equal, since $A_{6,1}(2)=1$ and $A_{6,5}(1)=0$, as shown in Fig. 3. Therefore, we must be careful when generalizing algorithms designed for lines in the first half of the first octant to lines in the second half of the first octant.

We define the reverse of the A_{H-k} as the conjugation of A_k, denoted by $\overline{A_k}$, i.e.

$$\overline{A_k} = A_{H-k}(H-1-i).$$

According to Properties 1 and 2 , when H is odd, we have

$$A_k(i) = 1 - \overline{A_k}(i).$$

We call this *conjugate complementary*. One of the most interesting things is that the conjugate complementary property holds also when H is an even integer, as shown in Property 3.

Property 3. *Let H and k be two integers satisfying $0<k<H$. Then*

$$A_k(i) = 1 - \overline{A_k}(i).$$

Proof. According to (3), there exists an integer m, such that $T_k(i) = T - ik + mH$.

If $A_k(i)=1$, then $0 \leqslant T_k(i) < k$ according to (4). Then

$$0 \leqslant T - ik + mH < k,$$

$$H-k \leqslant T-(i+1)k+(m+1)H < H,$$

$$\begin{aligned} T_{H-k}(H-1-i) &= (T-(H-1-i)(H-K))\%H \\ &= (T-(i+1)K)\%H \\ &= T-(i+1)K+(m+1)H \\ &\geqslant H-K. \end{aligned}$$

So, $A_{H-k}(H-1-i)=0$ according to (4), i.e.

$$\overline{A_k}(i) = 0.$$

Similarly we can prove that, if $A_k(i)=0$, then

$$\overline{A_H}(i) = A_{H-k}(H-1-i) = 1.$$

Therefore,

$$A_k(i) = 1 - A_{H-k}(H-1-i) = \overline{A_k}(i).$$

□

By observation, we find that BRDC is composed of a few repeated patterns as shown in Fig. 4. In Fig 4(a), the repeated patterns are 10,000 and 100,000, while in Fig. 4(b) the repeated patterns are 01111 and 011111. And the repeated pattern either starts with 1 and followed by a number of 0's, or starts with 0 and

0 0 1 0 0 0 0 1 0 0 0 1 0 0 0 0 1 0 0 0 0 1 0 0

(a) the first period of $A_{24,5}$

1 1 0 1 1 1 1 0 1 1 1 1 0 1 1 1 0 1 1 1 1 0 1 1

(b) the first period of $A_{24,19}$

Fig. 4. Repeated patterns of *displacement code*. (a) In $A_{24,5}$, the patterns are 1000 and 10000. (b) In $A_{24,19}$, the patterns are 0111 and 01111.

followed by a number of 1's. We conclude these, respectively, in Properties 4 and 5.

Property 4. *Let H and k be two integers satisfying $0<k<H$, and w be the number of 0's between two adjacent 1's in BRDC A_k. Then*

$$w=\left[\frac{H-k}{k}\right] \quad or \quad w=\left[\frac{H}{k}\right]=\left[\frac{H-k}{k}\right]+1.$$

Proof. Let $A_k(i)$ and $A_k(i+n)$ be a pair of adjacent 1's, then $w=n-1$. According to (3), we have

$$0\leqslant T_k(i)<k,\ T_k(i+1)\geqslant k,\ \ldots,T_k(i+n-1)\geqslant k,$$
$$0\leqslant T_k(i+n)<k.$$

Again according to (3) and (4), we can obtain the following results one by one:

$$0\leqslant T_k(i)<k,$$

$$T_k(i+1)=T_k(i)-k+H\geqslant k,\ \ldots,$$

$$T_k(i+n-1)=T_k(i+n-2)-k\geqslant k$$

$$T_k(i+n)=T_k(i+n-1)-k<k.$$

Combining these inequalities, we have

$$T_k(i+n-1)=T_k(i)+H-(n-1)k\geqslant k \quad \text{and}$$
$$T_k(i+n)=T_k(i)+H-nk<k.$$

Since $0\leqslant T_k(i)<k$, then

$$H-(n-1)k>0 \quad \text{and} \quad H-nk<k,$$

$$\frac{H-k}{k}-1<n-1<\frac{H}{k},$$
$$\frac{H-k}{k}-1<w<\frac{H}{k}.$$

$$\left[\frac{H-k}{k}\right]\leqslant w\leqslant\left[\frac{H}{k}\right]=\left[\frac{H-k}{k}\right]+1.$$

Therefore,

$$w=\left[\frac{H-k}{k}\right] \quad \text{or} \quad w=\left[\frac{H}{k}\right]=\left[\frac{H-k}{k}\right]+1.$$

□

If $k<H-k$, then $[(H-k)/k]>1$. According to the above property, the BRDC A_k is composed of some simple pattern, which starts with a *jumping code* followed by some 0's.

Property 5. *Let H and k be two integers satisfying $0<k<H$, and w be the number of 1's between two adjacent 0's in BRDC A_k. Then*

$$w=\left[\frac{k}{H-k}\right] \quad \text{or} \quad w=\left[\frac{H}{H-k}\right]=\left[\frac{k}{H-k}\right]+1.$$

Proof. Although this lemma can be proved using a similar method as that used for Property 4, we provide a simpler method taking advantage of Properties 3 and 4.

According to Property 4, the number of 0's between two adjacent 1's in A_{H-k} is $[k/H-k]$ or $[H/(H-k)]=[k/(H-k)]+1$. According to the definition of $\overline{A_k}$, the number of 0's between two adjacent 1's in $\overline{A_k}$ is also $[k/(H-k)]$ or $[H/(H-k)]=[k/(H-k)]+1$. Therefore, according to Property 3 the number of 1's between two adjacent 0's in A_{H-k} is $[k/(H-k)]$ or $[H/(H-k)]=[k/(H-k)]+1$.

If $H-k<k$, then $[k/(H-k)]>1$. According to the above property, BRDC A_k is composed of some simple pattern, which starts with a 0 followed by some *jumping codes*.

4. Decomposition of BRDC

Properties 4 and 5 imply that the length of each repeated simple segment is $N+1$ or $N+2$. In order to study the distribution of the different lengths of the simple segments, *segment code* S_k is defined for the BRDC A_k as follows: If $0<k<H-k$, then

$$S_k(i)=\begin{cases}\text{the } i\text{th segment starts with} & 1\\ 0 \quad \text{followed by } N+1\ 0'\text{s;} & \\ 1 \quad \text{the } i\text{th segment starts with} & 1\\ \text{followed by } N\ 0'\text{s.} & \end{cases}$$

Otherwise, $0<H-k<k$, then

$$S_k(i)=\begin{cases}\text{the } i\text{th segment starts with} & 0\\ 0 \quad \text{followed by } N\ 1'\text{s;} & \\ 1 \quad \text{the } i\text{th segment starts with} & 0\\ \text{followed by } N+1\ 1'\text{s.} & \end{cases}$$

According to Property 3:

$$A_k(i)=1-A_{H-k}(H-1-i),$$

the following claim holds.

Claim 4. *For* $0<k<H-k$, *we have*

$$S_{H-k}(i) = 1 - S_k(k-1-i).$$

Lemma 2 (Decomposition). *Let H and k be two integers satisfying* $0<k<H-k$. *There exists a certain integer m, such that*

$$S_{H,k}(i) = A_{k,g}(m+i).$$

Proof. We first suppose that H and k are relatively prime. Let $N = [(H-k)/k]$.

Let P_i be the index in $A_{H,k}$ of the jumping code of the ith segment. Then $T_{H,k}(P_0) = T_H\%k < k$. According to Property 4, there is at least N number of 0's following a jumping code. Then

$$\begin{aligned} T_{H,k}(P_i+N+1) &= T_{H,k}(P_i) - k + H - Nk \\ &= T_{H,k}(P_i) + H\%k = T_{H,k}(P_i) + h. \end{aligned}$$

Therefore:

(a) If $T_{H,k}(P_i)+h<k$, i.e. $T_{H,k}(P_i)<g$, then the number of all 0's following the jumping code at P_i is N. Then

$$\begin{aligned} S_{H,k}(i) &= 1 \quad \text{and} \quad T_{H,k}(P_{i+1}) = T_{H,k}(P_i) + h \\ &= T_{H,k}(P_i) - g + k. \end{aligned}$$

(b) Otherwise, $T_{H,k}(P_i) \geqslant g$, the number of all 0's following the jumping code at P_i is $N+1$. Then $S_{H,k}(i) = 0$, and $T_{H,k}(P_{i+1}) = T_{H,k}(P_i) + h - k = T_{H,k}(P_i) - g$.

Since H and k are relatively prime, then g and h are also relatively prime. Then there exists a certain integer m, such that $T_{k,g}(m) = (T_k - mg)\%k = T_{H,k}(P_0)$. According to (4) and (5) in Section 2, we have

(c) If $T_{k,g}(i)<g$, then $A_{k,g}(i) = 1$ and $T_{k,g}(i+1) = T_{k,g}(i) - g + k$;

(d) If $T_{k,g}(i) \geqslant g$, then $A_{k,g}(i) = 0$ and $T_{k,g}(i+1) = T_{k,g}(i) - g$;

Comparing the above (a) and (b) with (c) and (d), we have

$$T_{k,g}(m+i) = T_{H,k}(P_i) \quad \text{and} \quad S_{H,k}(i) = A_{k,g}(m+i).$$

If H and k are not relatively prime, let F be the greatest common divisor. Then, according to the above proof, there exists an integer m such that $S_{H/F,k/F}(i) = A_{k/F,(H/F)\%(k/F)}(m+i)$. Since $Y_{H,k} = Y_{H/F,k/F}$, then $A_{H,k} = A_{H/F,k/F}$. Then $S_{H,k} = S_{H/F,k/F}$ by the definition of the segment code. It is clear that F is also the common divisor of k and g. Then $A_{k,g} = A_{k/F,g/F} = A_{k/F,(H/F)\%(k/F)}$. Therefore $S_{H,k}(i) = A_{k,g}(m+i)$. □

Taking advantage of Property 3, Claim 4 and Lemma 2, we can easily prove the following lemma.

Lemma 3 (Decomposition). *Let H and k be two integers satisfying* $0<k<H-k$. *There exists a certain integer m, such that*

$$S_{H,H-k}(i) = A_{k,k-g}(m+i)$$

Since BRDC is a periodic sequence, Lemma 2 and Lemma 3 show that the *segment code* is nothing, but a simpler BRDC derived from the original BRDC.

5. Adaptive multi-pixel algorithm

In this section, we propose an adaptive multi-pixel line drawing algorithm as the application of Properties 3–5, and Lemma 2.

As shown in Fig. 1, the pixel approximation of line L_1 and L_2 can be divided into many simple segments: horizontal segments and catercorner segments. This can be analyzed by their BRDC. For a line in the first hexadecant, according to Property 4, the repeated simple segment in its BRDC starts with a jumping code followed by a number of 0's. By the definition of BRDC, the repeated segments correspond to horizontal line segments, as shown in Fig. 5. And the length of each segment is at least $\geqslant 2$.

For a line in the second hexadecant, according to Property 5, the repeated simple segment in its BRDC starts with a code 0 followed by a number of *jumping codes*. Again by the definition of BRDC, the repeated segments correspond to the catercorner line segments. And the length of each line segment is at least $\geqslant 2$.

Additionally, the length of each segment can be exactly calculated according to Lemma 2. The above conclusions suggest that, instead of pixel by pixel, we can draw a line segment by segment so as to improve the efficiency of line drawing. Based upon this, we propose a more efficient line drawing algorithm accordingly.

Shown as well in Properties 4 and 5, A_k and A_{H-k} have the same internal characteristics according to

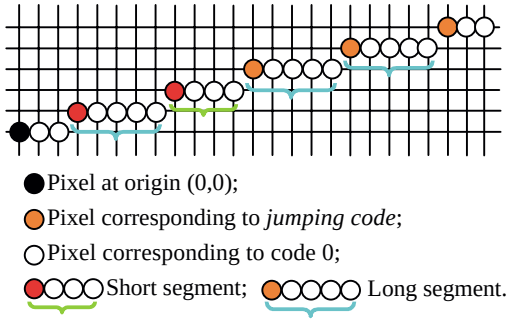

Fig. 5. Repeated segments in a BRDC correspond to simple line segments. In this figure, the line is $Y_{24,5}$.

Property 3. In other words, they are equivalent under the relationship of conjugate complements. Therefore, we introduce the algorithm only for lines in the first hexadecant.

Let L be a line in the first hexadecant, (x_a, y_a) and(x_b, y_b) be its end points and $x_a < x_b$. Then $H = x_b - x_a$, $k = y_b - y_a$ and $k < H - k$. Before introducing the algorithm, several things need to be clarified. Denote $N = [(H-k)/k]$ as in the proof of Lemma 2. Firstly, we need to calculate the value of P_0 and $T_k(P_0)$ in the proof of Lemma 2. Since $k < H - k$, A_k starts with a number $n = [T/k] = [N/2]$ of 0's. Then $P_0 = n$, and $T_k(P_0) = T - nk = T\%k$. Secondly, the BRDC of the limited line also ends with some 0's. Since the BRDC is periodic, its ending 0's and starting 0's together correspond to a repeated segment. If the corresponding segment is short, then the ending 0s' number is n; otherwise it is $n+1$. At last the number of repeated segments is $k-1$.

Algorithm 1: Single-pixel line drawing algorithm.

Input: Two end points (x_a,y_a) and (x_b,y_b) of a line in the first octant, and $x_a < y_a$.

```
H ← x_b − x_a
k ← y_b − y_a
T ← [(H−1)/2]
DrawPixel(x_a, y_b)
WHILE x_a < x_b DO
  IF t < k THEN
    y_a ← y_a + 1
    t ← t − k + H
  ELSE
    t ← t − k
  END IF
  x_a ← x_a + 1
  DrawPixel(x_a, y_b)
END WHILE
```

Algorithm 2: Adaptive multi-pixel line drawing algorithm.

Input: Two end points (x_a,y_a) and (x_b,y_b) of a line in the first half of the first octant, and $x_a < y_a$.

```
H ← x_b − x_a
k ← y_b − y_a
Divide(H, k, g, t, N , n)
DrawHS(x_a, y_b, n)
y ← y_a + 1
WHILE y < y_b DO
  IF t < g THEN
    DrawHS(x, y, N+1)
    t ← t − g + k
  ELSE
    DrawHS(x, y, N+2)
    t ← t − g
  END IF
  y ← y + 1
END WHILE
IF t < g THEN
  DrawHS(x, y, n)
ELSE
  DrawHS(x, y, n+1)
END IF
```

PROC Divide(H, k, g, t, N , n)

```
N ← [H/k] − 1
g ← H % k
n ← [N/2]
T ← [(H−1)/2]
IF N is odd THEN
  t ← T − [(H − k − g)/2]
ELSE
  t ← T − [(H − g)/2]
END IF
```

Fig. 6. Adaptive multi-pixel line drawing algorithm based on BRDC

In summary, the pseudo-code of the adaptive multi-pixel line drawing algorithm is described in Algorithm 2 in Fig. 6. In this algorithm, routine Divide(H,k,g,t,N,n) takes H and k as input and computes the values of g, t (i.e. $T_k(P_0)$ in the proof of Lemma 2). And starting from pixel (x,y), the routine of DrawHS(x,y,c) draws a horizontal line segment of c pixels, and the updated value of x is returned.

This algorithm is implemented in a Window NT workstation. Noted also in the previous works, the simple software implementation is not particular interesting, since in practice it would be realized at the chip level. Moreover, the results depend on the quality of the code generated by the complier operating system. Therefore, we theoretically analyze the computation cost. Note that only integers are involved in this algorithm, and the values of N and g can be obtained together by one integer division operation in the routine of Divide(…).

The computations involved in this algorithm is summarized in Table 2, where some operations such as "division by 2", increment of x and y are ignored. In the routine of Divide(…), there are totally three additions/subtractions and one division: $H-k$, $H-1$, $H-k+g$ or $H-g$, $H\%k$. In the initialization step of the main routine, there are totally four additions/subtractions: $N+1$, $N+2$, $-g+k$, $n+1$. In the loop of the main routine, there are k additions/subtractions.

Therefore, our algorithm is more efficient than the original Bresenham algorithm. The longer the repeated segment is, the more efficient the algorithm is. And our algorithm can adaptively determine the pixel number of the repeated segment for different lines.

Table 2
Comparison for line of $k/H < 0.5$

	Bresenham algorithm	Adaptive algorithm
Addition	$H+2$	$k+7$
Division	0	1
Testing	H	$k+2$

6. Conclusions and future works

We have presented an abstract representation called the BRDC, and studied some important properties of the BRDC, such as periodicity, symmetric, conjugate complementary, etc. These properties are very useful for designing and analyzing line drawing algorithms. Then we proposed a method to decompose a BRDC into a simpler one. At last, based on the above results, we have developed an efficient adaptive multi-pixel line drawing algorithm, which demonstrates the feasibility of BRDC.

Future work includes exploiting more useful properties of the BRDC of a line, and investigating the BRDC for curves, such as circles, and ellipses, etc.

Acknowledgements

The project is partly supported by the National Natural Science Funds of China for Excellent Young Scholars (Grant No: 69925204) and Innovative Research Groups (Grant No: 60021201) and for Research (Grant No: 60033010). The authors wish to thank the anonymous reviewers for their valuable comments.

References

[1] Tang Rongxi, Wang Jiaye, Peng Qunsheng. Computer graphics course. Beijing: Academic Press, 1990 [in Chinese].
[2] Bresenham JE. Algorithm for computer control of a digital plotter. IBM System Journal 1965;4(1):25–30.
[3] Liu Yong-Kui. Generating, clipping of line and curve. Ph.D. thesis, Zhejiang University, Hangzhou Zhejiang, 1997 [in Chinese].
[4] Freeman H. Computer processing of line drawing images. ACM Computing Surveys 1974;6(1):57–97.
[5] van Lierop MLP, van Overveld CWAM, van de Wetering HMM. Line rasterization algorithms that satisfy the subset line property. CVGIP 1988;41:210–28.
[6] Rokne JG, Wyvill B, Wu X. Fast line scan-conversion. ACM Transactions on Graphics 1990;9(4):376–88.
[7] Angel E, Morrison D. Speeding up Bresenham's algorithm. IEEE Computer Graphics and Applications 1991; 11:16–7.
[8] Kuzmin YP. Bresenham's line generation algorithm with built-in clipping. Computer Graphics forum 1995;14(5): 275–80.
[9] Yao C, Rokne JG. Bi-directional incremental linear interpolation. Computer and Graphics 1996;20(2):295–305.
[10] Rokne J, Rao Y. Double-step incremental linear interpolation. ACM Transactions on Graphics 1992;11(2): 183–92.
[11] Graham P, Iyengar SS. Double and triple step incremental generation of lines. In IEEE Computer Graphics and Application 1994. p. 49–53.
[12] Bao P, Rokne J. Quadruple-step line generation. Computers and Graphics 1989;13(4):461–9.
[13] Gill G. N-Step Incremental straight-line algorithms. IEEE Computer Graphics and Applications 1994. p. 66–72.
[14] Bresenham JE. Run length slice algorithm for incremental lines. In: Fundamental algorithms in computer graphics. Berlin: Springer-Verlag, 1985. p. 59–104.
[15] Miao L, Liu X, Peng Q, Bao H. An adaptive multi-pixel line drawing algorithm based on displacement code. Journal of Software, to appear [in Chinese].
[16] Boyer V, Bourdin J-J. Auto-adaptive step straight-line algorithm. IEEE Computer Graphics and Applications 2000;20(5):67–9.
[17] Boyer V, Bourdin J-J. Fast lines: a Span by span method, Proceeding of Eurographics 99, Computer Graphics Forum Series, vol. 18 no. 3. Oxford, UK: Blackwell Publishers, September 1999 p. 377–84.