A Hybrid Algorithm of Raster Conversion for Circle Based on Pattern Analysis

Haiwen Feng
Shenyang University of Technology/School of Software, Shenyang, China
Email:fhw19770704@sina.com.cn

Lianqiang Niu, Bowen Fu and Ling Zhong
Shenyang University of Technology / School of Software, Shenyang, China
Email: niulq@sut.edu.cn; fubw@sut.edu.cn; zhongl@sut.edu.cn

***Abstract*—A hybrid algorithm which combines multi-point movement, pixel movement and run-length technique is proposed in order to improve the speed of raster converion for circle arc. In the algorithm, the 1/8 circle arc is divided into 4 adjacent regions according to different raster circle pattern, and the method of 4-point movement and run-length I/O, double-point movement and run-length I/O, mixed 2-point and 3-point movement, 2-point movement along diagonal direction are adopted corresponding to different regions respectively. Different strategy is suitable for the feature of pixel pattern in every region. So the calculation numbers of each step is greatly reduced, and I/O operation numbers is decreased also by the use of run-length output. The experiment result shows that the efficiencies are increased 80% and 62% respectively comparing with the classic single-point algorithm and typical double-point algorithm. Compared with the existing run-length algorithm, the derivation of our algorithm is simple, and is made up of basic operations, so it is apt to be implemented by use of hardware.**

***Index Terms*— Circle Drawing, Pattern Analysis, Hybrid Algorithm, Run-Length Algorithm, Raster Conversion**

I. INTRODUCTION

Due to the important role of the basic graphic elements in computer graphics system, the generation algorithms for them have received wide attentions. So far, line generation algorithms have been studied deeply and the Bresenham Algorithm, the Middle-Point Algorithm proposed by Foley, double-point algorithm, run-length algorithm and multi-level run-length algorithm have been proposed[1-7]. However, a process of raster conversion for curve is complex because of its non-linear feature. As a result, fast algorithms are also rare besides the Bresenham Algorithm[8] and the Middle-Point Algorithm[9]. For fast generating circle arc, double-point algorithms and run-length algorithms were presented in [10] to [15], in which a double-point algorithm was that it draws two pixels in each loop operation, and a run-length algorithm was that it calculates directly the lengths of segments composing the raster circle arc and outputs those run-lengths in one time. Wu & Rokne[10] proposed a double-point generation algorithm for circle arc. In this algorithm raster pixels were divided into 4 regional patterns, the midpoint between the points (x+2, y) and (x+2, y-1), and that between the points (x+2, y-1) and (x+2, y-2) after the current point (x, y) were employed to compare with the really position of circle arc and make sure whether subsequent two points may be drawn directly. Since the calculation of intermediate point was eliminated, the efficiency of the algorithm was enhanced. However, because the pattern 2 and pattern 3 can not be distinguished, it leaded to wrong pixels selection. Based on the midpoint technique, Liu [11] redefined the decision-parameters in the midpoint between the two points (x+2, y) and (x+2, y-1) after the current point (x, y). Afterwards, he selected one of the relative four patterns by detecting the relationship between decision-parameters and the positions of three midpoints closer to current point. 3~4 times operations were only consumed for plotting one pixel and the efficiency of raster conversion was improved.

The representative run-length algorithms came from the work of Hsu[12] and Yao[13]. Hsu, Chow and Liu proposed an algorithm of scan-conversion for circle arc based on horizontal segments. We call it HCL Algorithm below. The algorithm was based on the following facts: The discrete loci of the second octant circle (from 45º to 90º) was composed of a series of horizontal segments which were made up adjacent pixels whose y-values were the same. This algorithm found out these horizontal segments and drawn them with the line drawing command. But, to a large number of horizontal segments with only a few pixels, the HCL Algorithm still adopted the line drawing command to draw them. Besides, the algorithm needed to adopt a square root operation to initialize decision-parameter, thus it's drawing speed was not very fast as a whole. Yao presented a hybrid approach for scan-conversion of circles [13]. This algorithm divided the second octant circle arc into three regions in which different calculation strategies as four-point movement, double-point movement and single-point movement were adopted for different regions respectively. For the first two regions, the run-lengths were calculated and were outputted by use of line drawing command, while for the last region the algorithm calculated every single pixel and

doi:10.4304/jsw.7.9.2053-2060

plotted them. Due to the reduction of the number of I/O operations and the number of basic calculation operations, the efficiency of scan conversion were improved. However, this algorithm done not accumulate the decision-parameters, only calculated the run-lengths while multiple-point movement occurs. Therefore, once square operation and multiplication operation must be invoked to calculate the initial value of decision-parameters before entering the later 1/16 circle arc. This influenced the efficiency of the algorithm to a certain extent and increased the complexity. In paper [15], we proposed an algorithm controlled by decision-parameters directly but with a strategy of double-point movement. This algorithm can be derived in a simple manner. However, its efficiency is more higher than that of single-point algorithms.

While aiming at improving the efficiency of algorithm, run-length algorithm is a development direction, this is partially because that the word pattern of hardware equipment can be utilized to reduce effectively output frequency and time. However it is difficult to calculate all lengths of run-lengths directly because of continues variation of the slope of tangent at curve. By analyzing the raster circle arcs, we know that the numbers of run-lengths whose length is larger than 4 accounts for only 35% in the second octant arc, and that of run-lengths whose length are between 2 or 3 accounts for 28%. Thus different regions should be calculated by different strategies. Based on the statistics for raster circle arcs, this paper aims at dividing a octant circle into 4 regions according to the difference between the lengths of horizontal segments. For the first three regions, we move with a four-point or double-point manner for suitable times according to the features of loci pattern at first, then convert the movement manner into single-point movement manner immediately to plot rest pixels. By this means, not only the advantage of the run-length algorithm is reserved, but also the number of calculation is reduced effectively.

II. DIVISION OF THE RASTER CIRCLE ARC

Here we only discuss the standard circles which are centered at original point of the coordinate system. Other circles can be gained by coordinate transformation. Because of the symmetry of circle, not loss of generality, we need only to discuss the second octant circle arc. Assume that the radius r of the circle is an integer.

Some basic properties about circle were argued in paper [13] and paper [14]. The results of theorem 1 and theorem 2 are given here.

Theorem 1. For any monotonic differentiable function curve $f(x)$, if there is $|f'(x)|\le 0.5$ in a continuous interval, the length l_h of each horizontal segment L_h of the raster conversion in this interval satisfies $l_h\ge 2$.

Theorem 2. For any monotonic differentiable function curve $f(x)$, if there is $|f'(x)| > 0.5$ in a continuous interval, the length l_h of each horizontal segment L_h of the raster conversion in this interval satisfies $l_h\le 2$.

Inference 1. Let C be a any subsection in the second 1/8 circle arc, if the length l of the horizontal run-length L contained in C satisfies $l\le 2$, for any diagonal run-length L_d, its length l_d satisfies $l_d\ge 2$. Here a diagonal run-length is a the line segment moving in direction of 45° angle.

The proof of above theorems may be shown in paper [13] and paper [14].

Theorem 3. For the raster conversion for second 1/8, let L_i and L_j be the horizontal run-lengths of the circle arc respectively. If the maximum of horizontal coordinate of L_i is less than the minimum of horizontal coordinate of L_j, the length l_i and l_j of L_i and L_j satisfy $l_j<l_i+2$.

Proof. Assuming that the length of the horizontal run-length L_i is n, and the X-coordinates of these points are $x_1^i, x_2^i, ..., x_n^i$, and the coordinates of intersection points of circle to these line $x=x_1^i$, $x=x_2^i$,..., $x=x_n^i$ are (x_1^i, y_1^i), (x_2^i, y_2^i), ..., (x_n^i, y_n^i); Assuming that the length of horizontal run-length L_j is $n+2$, and the X-coordinates of these points are x_0^j, x_1^j ..., x_{n+1}^j ,and the coordinates of intersection points of circle to these line $x=x_0^j$, $x=x_1^j$,..., $x=x_{n+1}^j$ are (x_0^j, y_0^j), (x_1^j, y_1^j), ..., (x_{n+1}^j, y_{n+1}^j), shown in Fig. 1. The coordinates of the last point of the run-length before L_i and the first point of the run-length after L_i are (x_0^i, y_0^i) and (x_{n+1}^i, y_{n+1}^i) respectively.

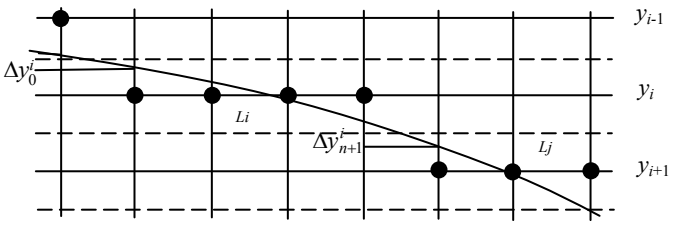

Figure 1. The relationship of lengths between a run-length L_i and a run-length L_j

Let $\Delta x_k^i = x_{k+1}^i - x_k^i$, $k = 0, 1, ..., n$,

And $\Delta y_k^i = y_{k+1}^i - y_k^i$, $k = 0,1,\ldots,n$.

Obviously, there is

$$\Delta x_0^i = \Delta x_1^i = \cdots = \Delta x_n^i = 1,$$

And $\sum_{k=0}^{n} \Delta y_k^i > 1$.

From Rolle Theorem, for any k and i, $0\le k\le n$, $1\le i\le m$ (Integer m is number of run-length for a region), there is ξ_k^i, $x_k^i \le \xi_k^i \le x_{k+1}^i$,which satisfies

$$\Delta y_k^i / \Delta x_k^i = \left| f'(\xi_k^i) \right|.$$

Considering $\Delta x_k^i = 1$, we have

$$\Delta y_k^i = \left| f'(\xi_k^i) \right|,$$

And

$$\sum_{k=0}^{n} \left| f'(\xi_k^i) \right| > 1.$$

For the run-length L_j, Let $\Delta x_k^j = x_{k+1}^j - x_k^j$, $\Delta y_k^j = y_{k+1}^j - y_k^j$, $k = 0$，1，…，n. Thus we have

$$\Delta x_0^j = \Delta x_1^j = \cdots = \Delta x_{n+1}^j = 1,$$

And $\sum_{k=0}^{n} \Delta y_k^j \le 1$.

Similarly, for any k and j, $0 \le k \le n$, $1 \le j \le m$, there is ξ_k^j, $x_k^j \le \xi_k^j \le x_{k+1}^j$. Thus we have

$$\Delta y_k^j / \Delta x_k^j = \left| f'(\xi_k^j) \right|.$$

Considering $\Delta x_k^j = 1$, we gain

$$\Delta y_k^j = \left| f'(\xi_k^j) \right|.$$

Since the $|f'(x)|$ is monotonic increasing and $\xi_k^j > \xi_k^i$ is met in the second 1/8 circle arc, we have

$$\left| f'(\xi_k^j) \right| > \left| f'(\xi_k^i) \right|.$$

Therefore, we gain

$$\sum_{k=0}^{n} \Delta y_k^j = \sum_{k=0}^{n} \left| f'(\xi_k^j) \right| \ge \sum_{k=0}^{n} \left| f'(\xi_k^k) \right| > 1.$$

It is in contradiction with $\sum_{k=0}^{n} \Delta y_k^j \le 1$. This proves that the segment L_j can not contain n+2 or more pixels.

Based on the theorem 3, the following result may be obtained.

Inference 2. While the second 1/8 circle arc is raster converted, there will not be a horizontal run-length whose length is more than 4 after the first horizontal run-length whose length is 3 appears. Similarly, there will not be a horizontal run-length whose length is more than 3 after the first horizontal run-length whose length is 2 appears.

According to theorem 1 and theorem 2, the circle arc is divided into two parts C_1 and C_2, and the division point satisfies $|f'(x)| \le 0.5$. Region C_1 is only composed of the run-lengths whose length are not less than 2, and Region C_2 is only composed of the run-lengths of whose length are not more than 2. According to theorem 3 and inference 2, Region C_1 is further divided into 3 parts D_1, D_2 and D_3 at clockwise direction. The three Regions are composed of the run-lengths whose length are more than 4, that are 3 or 4, and that are 2 or 3 respectively, shown in Fig. 2.

While the algorithm is constructed, the region transition method is, if the first horizontal run-length whose length is less than 4 appears, the drawing of Region D_1 stops and we should enter Region D_2. Further, if the first horizontal run-length whose length is less than 3 appears, the drawing of Region D_2 stops and we should enter Region D_3.

III. CACULATION OF RUN-LENGTH

Fast calculation for length of run-length may be achieved in two ways. The first one is to iterate according to analysis for the equation of circle. The second one is to detect decision-parameter directly. The shortage of the former can not accumulate decision-parameter. Once different methods needs to be applied region C_1 and C_2, before the algorithm enters region C_2, the accumulation error must be calculated at once by using of several times multiplication operations. Therefore, in this paper, the strategies of four-point movement and double-point movement are adopted controlled by decision-parameter respectively for the above regions. The technique of single-point movement is applied for the rest pixels of a run-length which number is less than 4 or 2. The calculations of the run-length are finished completely.

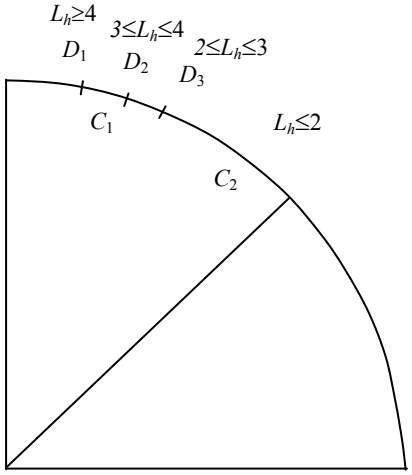

Figure 2. Region division of the 1/8 circle arc

A. The Length Calculation of Run-Length for Region D_1

Shown in Fig. 3. Let P_0 be the point before the first point of a new run-length. If the next run-length steps in four-point movement manner, the algorithm will arrive at the point P_4 after moving four points. The decision-parameter is updated as follows

$D = D + 2x_0 + 3 + 2x_1 + 3 + 2x_2 + 3 + 2x_3 + 3$
$= D + 8x_0 + 24$
$= D + 8x_3$.

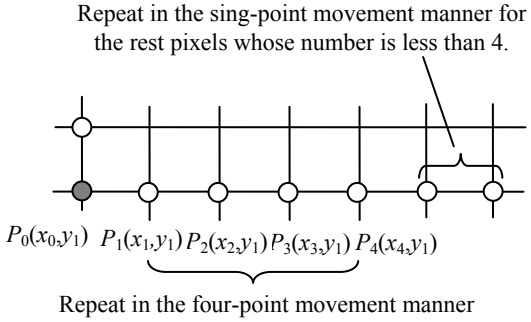

Figure 3 Lengh calculation pattern of run-length in Region D_1

Let $t = 8x_3$, $d = -D$, $x0 = x_1$. The algorithm should be implemented according to a progress below.

(1) At the first time, we detect whether the length of run-length is more than 4. If the condition is true, the iteration below will be executed.

```
if(d>t)
{  d = d - t;
```

$t = t + 32$;
$x = x + 4$;
}
Otherwise, go to (4).

(2) Continue to execute iteration operation with four-point movement manner for the rest part of the run-length.

while($d>t$)
{ $d = d - t$;
$t = t + 32$;
$x = x + 4$;
}

(3) Moving four points directly or adopting single-point movement manner to calculate and plot the rest pixels whose number is not more than 4.

Let N=-3-2x. If $d>t-N$, number of the rest pixels is 4, thus we may move four points forwards at once, and the decision-parameter should be updated as follows

$d = d - t$;
$x = x + 4$;

Otherwise, a loop with single-point movement manner should be executed for the rest part.

while($d>0$)
{ $d = d + N$;
$N = N - 2$; x++;
}

The length calculation for run-length is finished, and the segment should be drawn. After this, go to (1).

(4) If $d-t>-3-2x$, it shows that the length of the run-length is 4, and the segment can be drawn directly. Then go to (1). Otherwise, it shows that the length of the run-length is less than 4, and the drawing process of Region D_1 is finished.

B. The length calculation of run-Length for Region D_2

All lengths of run-lengths in Region D_2 are only 3 or 4, shown in Fig. 4. The calculation process in this region is similar to that in Region D_1, but only double-point movement is done first at once, and the decision-parameter should be updated as:

$D=D + 2(x_0 - y_0) + 5 + 2x_1 + 3$
$=D + 4x_0 - 2y_0 + 10$
$=D + 4x_2 - 2y_1$.

The iteration calculation and drawing process are described as follows:

(1) A double-step movement is executed.

$D = D + D+ 4x_2 - 2y_1$;
$x = x + 2$; $y=y-1$;

(2) If $D\geq0$, it shows that the length of the run-length is 2 (Referring to Fig. 4(b)), and the drawing process of Region D_2 is finished. Otherwise, stepping forwards one point and updating decision-parameter as:

$D = D + 2x + 1$;
$x = x+1$;

(3) If $D<0$, it shows that the run-length is composed of 4 pixels (Referring to Fig. 4(a)). In the case the algorithm continues to step one point forwards.

$D = D + 2x + 1$;
$x = x+1$;

Otherwise, this shows that the run-length has 3 pixels, thus the segment should be drawn directly. After this, go to (1).

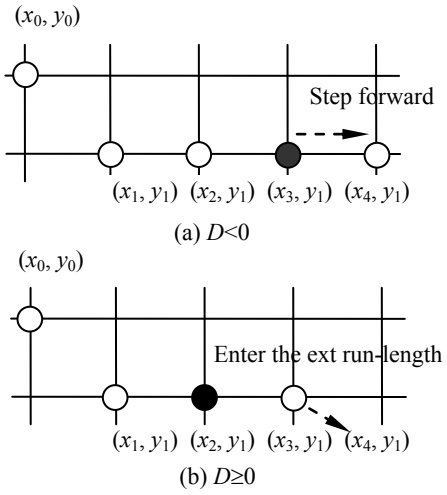

Figure 4. Step according to patterns in Region D_2

C. The Length Calculation of Run-Length for Region D_3

All lengths of run-lengths in Region D_3 are only 2 or 3. Before iteration is executed, operation of double-point movement should be executed at once, and the process will arrive at the point (x_1, y_0), shown in Fig. 5.

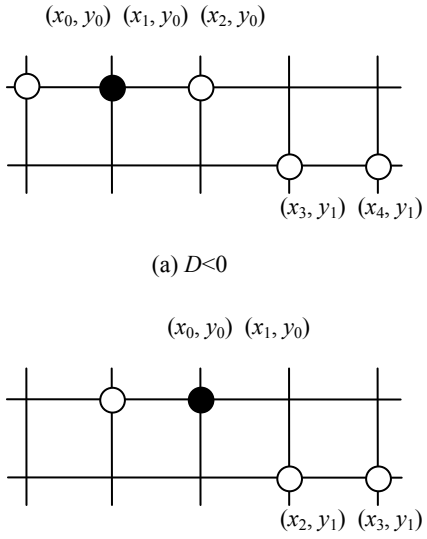

Figure 5. The run-length Patterns included in Region D_3

When the condition $2x<y$ is satisfied, the loop process is executed as follows.

(1) The decision-parameter D is detected. If $D<0$, the raster pattern is what illustrated with Fig. 5(a).

Sing-point movement is executed, and point (x_2,y_0) is selected. The current run-length is finished.

The operation enter next run-length, and the point (x_3,y_1) and point (x_4,y_1) are selected.

(2) If $D\geq0$ is satisfied, the raster pattern is shown in Fig. 5(b).

The current run-length is finished.

The operation enter the next run-length, and point (x_2,y_1) and point (x_3, y_1) are selected.

D. The Length Calculation of Run-Length in Region C_2

According to inference 1, all lengths of diagonal run-lengths in Region C_2 are more than or equal to 2. The operation of double-point movement in diagonal direction should be executed, if the decision-parameter of current point (x_0,y_0) satisfies $D\geq 0$. Then, one of three patterns is selected, and shown in Fig. 6. The iteration calculation and drawing process are described as follows:

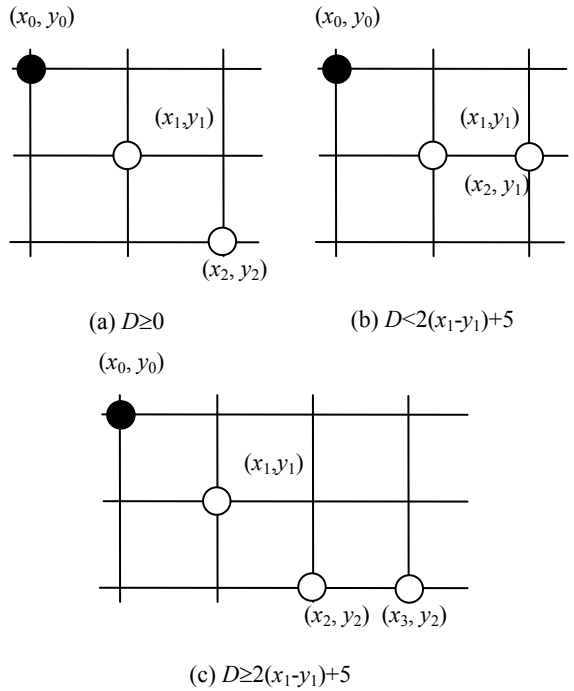

Figure 6. The calculations of run-lengths in Region C_2 for different raster pattern

(1) double-point movement in diagonal direction is executed, and the decision-parameter should be updated as follows

$D = D + 4(x - y) + 14$.

(2) If $D\geq 0$, the raster pattern is what illustrated with Fig. 6(a). The point (x_1,y_1) and the point (x_2, y_2) should be selected, and the decision-parameter of point (x_2, y_2) satisfied $D\geq 0$ this moment. After this, go to (1).

(3) If $D<2(x_1-y_1)+5$, the raster pattern is what illustrated with Fig. 6(b). it shows that the decision-parameter of the point (x_1, y_1) satisfies $D<0$, and this diagonal run-length is finished. Sing-point movement is executed in horizontal direction again, and the point (x_2, y_1) should be selected. The algorithm enters the next run-length, and the decision-parameter should be updated.

$D = D + 2y_1 - 2$

$= D + 2y_2$.

After this, go to (1).

(4) If $D\leq 0$ and $D\geq 2(x_1-y_1)+5$, the raster pattern is what illustrated with 5(c). Then, the point (x_1,y_1) and the point (x_2,y_2) are selected, and the diagonal run-length is finished. Sing-point movement is executed in horizontal direction again, and the point (x_3, y_2) should be selected. The algorithm will enter the next run-length, and decision-parameter should be updated.

$D = D + 2x_2 + 3$.

After this, the process should go to (1).

Ⅳ. ALGORITHM DESCRIPTION

According to the method described above, a fast hybrid algorithm constructed with regions division is described by pseudo code as follows.

```
D←r, x←-1,y←r;
//Generating Region D1
Region1:
x++; x0←x; x←x+2; t←8x;
if(d>t){
  d←d-t; t←t+32; x++;
  while(d>t){
     d←d-t;
     t←t+32; x←x+4;
  }
  N←-3-2x;
  if(d>t-N)
  { d←d-t; x←x+4; }
  else
      while(d>0){
          d←d+N;
          N←N-2; x++;
      }
}
else{
  d←d-t;
  if(d > -3-2x)
     x++;
  else{
      x←x-3; d←d+t-2y;
      y++; d←-d;
      goto Region2;
  }
}
Line(x0, x, y);
y--; d←d+2y;
goto Region1
//Generating Region D2
Region2:
x++; x0←x; x++; y--;
D←D+4x-2y;
if(D ≥ 0){
    Line(x0,x,y);
    goto Region3;
}
else{
    x++; D←D+2x;
    if(D < -1){
        D←D+2x+4; x++;
    }
    else
       D++;
    Line(x0,x,y);
}
goto Region2;
```

```
//Generating Region D3
Region3:
x++;  y--;
D←d+2(x-y)+1;
if(D < 0){
   x0←x;  D←d+2x+3;  x++;
   M←4x-2y;  T←2x-y+6;
   while(T < 0){
      if(D<0){
         x++;
         Line(x0, x, y);
         D←D+M+2x+15;
         x++;  x0←x;
         x++;  y--;
         M←M+14; T←T+7;
      }
      else{
         Line(x0, x, y);
         x++;  x0←x;  x++;  y--;
         M←M+10;  D←D+M;  T←T+5;
      }
   }
   Line(x0, x, y);
}
else{
    Pixel (x, y);
}
//Generating Region D4
if(D < 0){
   D←D+2x+3; x++;
   Pixel(x, y);
}
N←4 (x-y) +14;
while(x < y-4){
    D←D+N;
    N←N+16;
    if(D ≥ 0){
      x++;  y--;
      Pixel(x, y);
      x++;  y--;
      Pixel(x, y);
   }
   else{
      if(d < N/2-6){
         x++;  y--;
         Pixel(x, y);
         x++;
         Pixel(x, y);
         D←D+2y-2;  N←N-4;
      }
      else{
         x++;  y--;
         Pixel(x, y);
         x++;  y--;
         Pixel(x, y);
         x++;
         Pixel(x, y);
         D←D+2x+1;  N←N + 4;
      }
   }
}
while(x < y){
    Pixel(x, y);
    if(d < 0)
        D←D+2x+3;
    else{
        D←D+2(x-y)+5;  y--;
    }
    x++;
}
```

In the algorithm, the function of Pixel (x, y) is to draw a pixel at point (x, y), the function of Line (x_0, x, y) is to draw the horizontal segments from point (x_0, y) to point (x, y).

Ⅴ. Analysis And Discussion

Here we will discuss the efficiency of our algorithm and compare it to that of some typical algorithms. Considering that the actual execution time of a algorithm is influenced by many factors, the basic calculation number of the algorithm is adopted as a evaluation standard. The operation usually included in a algorithm of raster conversion addition, shift, sign test and comparison which are regarded as the same basic operation spending one unit of time. The numbers of coordinate movement and assignment operation have only small differences in proposed algorithms and therefore is not calculated.

Five and six times basic operation will be needed for plotting one pixel along horizontal direction and diagonal direction respectively in the classic Bresenham Algorithm and the Middle-Point Algorithm. While the number of calculation in our algorithm changes on the different regions and needs to be calculated separately.

Referring to section II, the 1/8 circle arc is divided into 4 regions, so we calculate their times in this way.

(1) Region D_1. This region is only composed of run-lengths whose lengths are not less than 4. They can be divided into two categories according to whether they are integer multiple of 4.

Let length of a horizontal run-length be l_h. If the length l_h of the run-length is integer multiple of 4, when this run-length calculation is finished, $0.75l_h+6$ times basic operations are consumed. So, basic operation number consumed for plotting one point is $0.75+l_h/6$ times on average.

If the length l_h of the run-length is not integer multiple of 4, the $0.75l_h+3(l_h \ MOD \ 4)+4$ times basic operations are consumed after the whole run-length calculation is finished. So, the basic operation numbers consumed for plotting one point is $0.75+(3(l_h \ MOD \ 4)+4)/l_h$ on average. Here, identifier *MOD* in formula means modulo operation.

(2) Region D_2. All lengths of the run-lengths are 3 or 4 in the region. When this run-length calculation is finished, 3 times or 4 times basic operations are consumed respectively. So, the basic operation numbers consumed for plotting one pixel are 2.66 or 3 respectively on average.

(3) Region D_3. All lengths of the run-lengths are 2 or 3 in the region. The number of calculations consumed is decided by pattern (a) or pattern (b) shown in Fig. 5. To

pattern (a), the basic operation number consumed of three-point movement is 8 and each point consumes 2.66 basic operations on average. To pattern (b), the basic operation number consumed of double-point movement is 5 and each point consumes 2.5 times basic operations on average.

(4) Region C_2. The number of calculations consumed is decided by three patterns shown in Fig. 6. To pattern (a), the basic operations number of double-point movement is 5 and each point consumes 2.5 times basic operations on average. To pattern (b), the basic operations number of double-point movement is 12 and each point consumes 6 times basic operations on average. To pattern (c), the basic operations number of three-point movement is 12 and each point consumes 4 times basic operations on average.

Considering that the division points of adjacent regions are not easy to be determined precisely, there is a little great error in approximate calculation. In this paper, some groups of representative circles are chosen, and basic operation numbers consumed by each group for generating the second 1/8 arc in different algorithms are tracked and are statistically calculated. In Table 1, the comparison result of calculation numbers between our algorithm and several typical algorithms is shown.

As is shown in Table 1, the experimental results show that the efficiency of our algorithm has been enhanced about 70% and 50% respectively compared to that of the Middle-Point Algorithm and the double-point algorithm proposed by Liu. If we only think about the basic operation numbers, the efficiency of our algorithm is little higher than that of Yao algorithm. However, the Yao algorithm contains twice multiplications. By contrast, our algorithm only uses simple operations and construction. Besides, consistent strategies are adopted to generate different regions with a little difference of movement account rather than widely different method invoked for different regions as Yao Algorithm. Fig. 7 shows the comparison result of efficiencies among our algorithm, the Middle-Point Algorithm and the algorithm proposed by Liu for generating some typical circles.

arcs whose radii satisfy $100 \le r \le 500$, Table 2 shows the total numbers of calculations after converting output time into number of calculation. It shows that the efficiency has been enhanced 80% and 62% respectively contrast to the Middle-Point Algorithm and the Liu Algorithm.

Table 2. Three Algorithms Comparisons of Circle$100 \le R \le 500$

Algorithm	Baisc Operation	Output	Scale Total Number	Average Number for One Pixel
This Paper Algorithm	264891	44670	398901	4.69
Middle-Point Algorithm	461556	85164	717048	8.43
Yao Algorithm	310777	44066	442975	5.21
Liu Algoithmn	390852	85164	646344	7.60

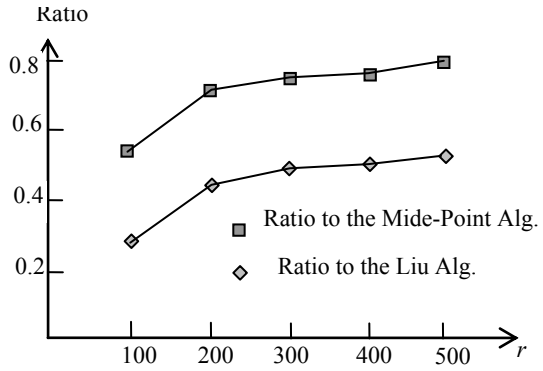

Figure 7. Ratio of computation of our algorithm to that of others

VI. CONCLUSION

A fast hybrid algorithm for generating circle arc based on the pattern analysis is proposed in this paper. By dividing 1/8 circle into 4 regions, and applying different strategieses to different regions, the algorithm can treate all kind of pixel pattern effectively, and reduce calculation number substantially. The other major feature of this algorithmn is that decision parameters are employed to implement iteration operation directly, which eliminates the shortage to acculumate errors needed for general run-length algorithms at one time before drawing the last region. Overall, the derivation of the new algorithm and operations are both simple. Its efficiency has been enhanced more than 80% and 60% respectively compared to the classic single-point

Table 1. The Calculation Number Comparsion between This Paper Algorithm and Other Classic Algorithm

Radis	This Paper Algorithm	The Middle-Point lgorithm	Improve Ratio	Yao Algorithm	Improve Ratio	Liu Algorithm	Improve Ratio
100	3.49	5.44	55.87%	3.85	10.17%	4.62	32.38%
200	3.18	5.42	70.44%	3.68	15.60%	4.58	44.03%
300	3.1	5.42	74.84%	3.66	18.08%	4.59	48.06%
400	3.08	5.42	75.97%	3.63	17.94%	4.59	49.03%
500	3.03	5.42	78.88%	3.61	19.05%	4.58	51.16%

The algorithm in this paper can directly output all the horizontal run-lengths whose lengths are more than 2 instead of single point. The time of one I/O operation is generally about three times that of a basic operation. If hardware device supports output with word pattern, the time consumed for outputting a horizontal segment will be the same as that of plotting a single pixel. According to the above discussion, to raster conversion for some 1/8

algorithm and general double-point algorithm. It is easy to be implemented by hardware and should be developed for scan conversion of circle in graphics device widely.

REFERENCES

[1] J. E. Bresenham. "Algorithm for computer control of digital plotter", *IBM System Journal*, vol. 4(1), pp. 25-30, 1965.
[2] P. Graham, S. Sitharama Iyengar. "Double-and-triple-step incremental linear interpolation", *IEEE Computer Graphics and Applications*, vol. 14(3), pp. 49-53, 1994.
[3] G. W. Gill. "N-step incremental straight-line algorithms", *IEEE Computer Graphics and Applications* , vol. 14 (3), pp. 66 -72, 1994.
[4] V. Boyer, J. J. Bourdin. "Auto-adaptive step straight-line algorithm", *IEEE Computer Graphics and Applications* , vol. 20 (5), pp.67-69, 2000.
[5] X. H. Lin, T. Zhang. "An adaptive multi-slice line drawing algorithm", *Journal of Computer-Aided Design & Computer Graphics*, vol. 18(8), pp. 1136-1141.
[6] Y. L. Jia, H. Zhang, Y. L. Jiang. "A modified Bresenham algorithm of line drawing", *Journal of Image and Graphics*, vol. 13(1), pp. 158-61, 2008.
[7] B. M. Huang, L. Zhang. "Self-adaptive step straight- line algorithms", *Journal of Tsinghua University(Sci & Tech)* , vol. 46(10), pp. 1719-1722, 2006.
[8] J. E. Bresenham. "A linear algorithm for incremental digital display of circular arcs", *Communications of the Association for Computing Machinery*, vol. 20(2), pp. 100-106, 1977.
[9] J. D. Foley, A. V. Dam, S. K. Feiner, et al.. "Computer Graphics: Principles and Practice". *MA: Addison-Wesley*, 1990.
[10] X. Wu, J. Rokne. "Double-step incremental generation of lines and circles", *Computer Vision: Graphics and Image Processing*, vol. 37(3), pp. 331-344, 1987.
[11] Y. K. Liu, J. Y. Shi. "Double-step circle drawing algorithm with and without grey scale", *Journel of Computer-aided Design & Computer Graphics*, vol. 17(1), pp. 34-41, 2005.
[12] S. Y. Hsu, L. R. Chow, C. H. Liu. "A new approach for the generation of circles", *Computer Graphics Forum*, vol. 12(2), pp. 105-109, 1993.
[13] C. Yao, J. G. Rokne, "Hybrid scan-conversion of circles", *IEEE Trans. on Computer Vision and Graphics*, vol. 1(4), pp. 311-318, 1995.
[14] J. Cheng, G. D. Liu, J. R. Tan. "A fast algorithm for the drawing of circles", *Journal of Software*, vol. 13(12), pp. 2275-2280, 2002.
[15] L. Q. Niu, B. H. Li, H. W. Feng, et al. "A double-stepping run-length algorithm for fast circle drawing", *ICIC Express Letters*, vol. 4(5), pp. 1695-1700, 2010.

Haiwen Feng holds a BSc in computer science and application. He is currently a PhD candidate in the specialty of electric machines and electric apparatus of Shenyang University of Technology. His research interests include computer graphics, computer-aided design, computer communication, and the application of these areas in industry.