The Structure of Digital Straight Line Segments and Euclid's Algorithm

Ivo Považan
Institute of Control Theory and Robotics, SAS
Dúbravská cesta 9, 842 37 Bratislava
Slovak Republic
Phone: +421-7-3782985
e-mail: utrrpova@savba.savba.sk

Ľubomir Uher
Institute of Control Theory and Robotics, SAS
Dúbravská cesta 9, 842 37 Bratislava
Slovak Republic
Phone: +421-7-3782985
e-mail: utrrpova@savba.savba.sk

ABSTRACT

The presented paper describes the algorithm for systematic generating of all digital straight line segments starting from the S[0,0] point and ending at the E[h,v] point. The parameters are subdued to conditions $h \geq v > 0$ and $GCD(h, v) = 1$. The straight line segment is represented by a v–tuple of integer numbers. Euclidean algorithm is the skeleton of the forward running part of the algorithm. In backward running part r_{k-1}–tuples are generated sequentially representing digital straight line segment beginning at the point S[0,0] and ending at the E $[r_{k-2}, r_{k-1}]$ point, where $r_{k-2} \bmod r_{k-1} = r_k$.

Keywords: Algorithms, Euclidean algorithm, digital lines, structure of digital straight line, transition points

1 Introduction

Simple and effective algorithms for construction, recognition and sorting of digitized curves still attract interest in all fields related to image processing and computer graphics. In computer graphics emphasis is on the *generation* of digitized lines mostly determined by two points. A digital line segment is often used as the basic element in building up graphic constructions. In 1965 Bresenham [1] published the basic algorithm, which is often used in computer graphics till now. In the 80's and early 90's the following methods have been developed: Double-Step, Triple-step and N-step algorithms [12, 14, 15] as well as Slice algorithms [9, 13], starting from a certain implicit assumption that some internal structure of the straight line may be utilized to reduce the number of iterations. In image processing as a rule the measurement of characteristic parameters of a given cluster of points (arc), and solution of the reverse problem takes place. Namely, it should be found out whether the given cluster of points may be interpreted as a straight line segment, or as a segment of a circle or of another type of curve. As the straight line segment represents a subset of all possible digital lines between the given two points, some restricting linearity conditions must be added, specifying the necessary and sufficient conditions for a set of points to form the straight line segment. Rosenfeld [3] defined the straight line segment introducing the so called chord property. A digital arc is a digitization of a straight line segment if and only if it has the chord property. To describe a digital straight line on a lattice, Freeman's chain coding scheme can be used [2]. In a later paper three conditions are suggested which must be obeyed by a chain code to be a representation of the digital straight line. Analysis of the problem was continued in [4], where an algorithm is given with the output saying whether the analyzed chain code represent the straight line segment or not. In the paper [5] line properties of sets of points are analyzed proving that some properties of lines in real space do not hold for their digital analogs. A digital straight line segment, as well as a pixel, represents one of the basic (atomic) elements used in image processing as well as in computer graphics. The detection of straightness of digital curves is the subject of the paper [6]. The straight line segment is a special case of a line with some extra conditions supplied. Namely, in our case, the line connects two endpoints in the shortest way possible. The word "shortest" is used in the sense of some properly defined metrics characterizing a given type of space. Such a line may be represented by an analytic equation in real space. After the process which we call "digitization" or "mapping", the digitized analog of the original line is described by a set of points and then we call it a digital straight line. In

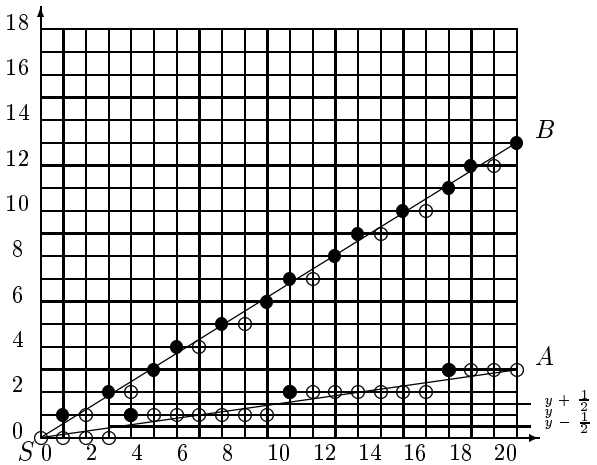

Figure 1:
In the figure approximations of two digital straight lines SA and SB are depicted. For both lines SA, SB the transition points are labeled with a bullets. Digitization scheme is based on the equation $y = \lfloor \frac{v}{h} \cdot x + \frac{a}{b} \rfloor$. The straight line SA obeys the Bresenham's approximation $\frac{a}{b} = \frac{1}{2}$ and the straight line SB differs from it by the more general form of $\frac{a}{b} \neq \frac{1}{2}$.

accordance with [7], topological properties of original straight lines, namely connectivity, must be preserved in the mapping. From this point the ways of treatment of the digital line in image processing and in computer graphics continue in different ways.
The relations between Euclid's algorithm and the structure of a digital straight line segment were studied in [10, 11]. In the presented paper an algorithm is given for generating all possible straight line segments connecting two fixed points. In our previous paper [16] we have concentrated our attention to Bresenham's class of straight line segments.
The present article is organized as follows: in the next section some preliminary definitions and assumptions are given, where main stress is put on the definition of "transition points" and their use for representation of digital curves. An interesting example is given for the use of the technique of transition points for derivation of the classical Bresenham's algorithm. In chapter 3 the DSLS (Digital Straight Line Segment) algorithm is described, with its skeleton formed by Euclid's algorithm. The output of the algorithm is a data structure defining a set of transition points for all possible DSLS connecting the points $S[0,0]$, $E[r_{k-2}, r_{k-1}]$, with $r_{k-2} \bmod r_{k-1} = r_k$. In the last chapter some ideas are outlined showing possible directions of future research in relation to the just described algorithm.

2 Transition Points of the Digital Curves

In the following all variables and constants used are supposed to be integers. In this chapter we concentrate our attention on certain subclass of digital curves. After the specifications of the subclass are fixed, we introduce one of the possible data structures describing the subclass of digital curves. Then, we make use of the data structure of this subclass in construction of the DSLS algorithm. For digital line construction purposes let us introduce two operators, often used in digital curves processing. Given the initial point $P[x,y]$, let the operators O_x and O_{xy} be defined as follows:

$$O_x(P[x,y]) = P[x+1,y] \tag{1}$$

$$O_{xy}(P[x,y]) = P[x+1,y+1] \tag{2}$$

The specifications for the notion of "digital curves" as used further are:

1. initial point to each curve is $S[0,0]$.
2. for any point of the digital curve $P_2[x,y]$ (except of $S[0,0]$) there exist one point $P_1[x_1,y_1]$ for which either
$$O_x(P_1[x_1,y_1]) = P_2[x,y] \quad \text{or}$$
$$O_{xy}(P_1[x_1,y_1]) = P_2[x,y] \tag{3}$$
3. there exists a single point $E[h,v]$ on the curve for which there does not exist any point $P_2[x,y]$ observing either the condition
$$O_x(E[h,v]) = P_2[x,y] \quad \text{or}$$
$$O_{xy}(E[h,v]) = P_2[x,y] \tag{4}$$
We define the point $E[h,v]$ to be the endpoint of the digital curve, so that it has not a successor point.

We call the digital curves exhibiting the above three properties "non-descending digital curves".
Definition: We define the transition points for non-descending digital curves as follows: If for the given point $T[x,y]$ of the non-descending digital curve there exists a predecessor point $T[x-1,y-1]$ of the same curve, i.e. it observes the condition

$$O_{xy}(T[x-1,y-1]) = T[x,y], \tag{5}$$

then the point $T[x,y]$ is called the transition point. It is possible to define more data structures containing complete data representing the given non-descending digital curve. As it is shown below, in our case the Non-descending Digital Straight Line Segment with the initial point $S[0,0]$ and the endpoint $E[h,v]$ may be represented by the v-tuple

$$T \equiv T[1], T[2], \ldots T[i], \ldots T[v]$$

where $T[i]$ is the integer value of x-coordinate and i is the integer y-coordinate of the i–th transition point. Further we show that a proper digitization scheme may be derived yielding explicit formulas for

calculation of coordinates of transition points. The starting point of the digital straight line segment has coordinates $S = [0, 0]$ and the coordinates of the end point are $E = [h, v]$ (mnemonics: means h-orizontal, v-ertical). h and v are related by an inequality $0 \leq v \leq h$. An elegant definition of any integer function for the incremental linear interpolation using the ceiling operator has been suggested in [8]. The following two equations represent a generalization of the function known from [8], where instead of $\frac{1}{2}$ the generalized form $\frac{a}{b}$ is used with $a < b$.

$$y = \left\lfloor \frac{v}{h} \cdot x + \frac{a}{b} \right\rfloor \tag{6}$$

$$y = \left\lceil \frac{v}{h} \cdot x - \left(1 - \frac{a}{b}\right) \right\rceil \tag{7}$$

Let us rearrange the expression: $\frac{v \cdot x}{h} = q + \frac{r}{h}\ h > r \geq 0$. At a given point only one of the three possible cases can take place:

1. $\frac{r}{h} < 1 - \frac{a}{b}$ so that $\left\lfloor \frac{v}{h} \cdot x + \frac{a}{b} \right\rfloor = q$
   $\left\lceil \frac{v}{h} \cdot x - (1 - \frac{a}{b}) \right\rceil = q$
2. $\frac{r}{h} = 1 - \frac{a}{b}$ so that $\left\lfloor \frac{v}{h} \cdot x + \frac{a}{b} \right\rfloor = q + 1$
   $\left\lceil \frac{v}{h} \cdot x - (1 - \frac{a}{b}) \right\rceil = q$
3. $\frac{r}{h} > 1 - \frac{a}{b}$ so that $\left\lfloor \frac{v}{h} \cdot x + \frac{a}{b} \right\rfloor = q + 1$
   $\left\lceil \frac{v}{h} \cdot x - (1 - \frac{a}{b}) \right\rceil = q + 1$

I.e. if the expression $\frac{v}{h} \cdot x + \frac{a}{b}$ is not integer then:

$$\left\lfloor \frac{v}{h} \cdot x + \frac{a}{b} \right\rfloor = \left\lceil \frac{v}{h} \cdot x - \left(1 - \frac{a}{b}\right) \right\rceil \tag{8}$$

From the above it clearly results what is the difference between the Incremental Linear Interpolation represented by Eq.(7) and by Eq.(6)
It follows from the definition of the transition point that if we need to find the transition points coordinates, then for constant values of their y-coordinates $y_k = 1, 2 \ldots, v-1, v$ we should look for a minimum integer x observing the equation

$$y_k = \left\lfloor \frac{b \cdot v \cdot x + a \cdot h}{b \cdot h} \right\rfloor;$$

hence

$$y_k \leq \frac{bvx + ah}{bh}\ (a) \quad x \geq \frac{bhy_k - ah}{vb}\ (b)$$

and for such a minimum x,

$$x_k = \left\lceil \frac{(by_k - a)h}{vb} \right\rceil\ (c). \tag{9}$$

For the given y-values, from the last equation we can calculate x-coordinates of the above specified transition points. Especially for $a = 1$ and $b = 2$ one intermediate result from preceding modifications can be expressed in the form of an inequality:

$$2v \cdot x + h - 2h \cdot y_k \geq 0 \tag{10}$$

From the inequality (10) the Algorithm 1 may be derived directly [1]:

Algorithm 1 Bresenham 1965

Require: $h \geq v$; h,v is integer
1: $d \leftarrow 2v - h; i_1 \leftarrow 2v; i_2 \leftarrow 2v - 2h; x \leftarrow 0; y \leftarrow 0;$
2: $setpixel(x, y);$
3: **while** $x \neq h$ **do**
4: **if** $d \geq 0$ **then**
5: $d \leftarrow d + i_2; y \leftarrow y + 1;$
6: **else**
7: $d \leftarrow d + i_1;$
8: **end if**
9: $x \leftarrow x + 1; setpixel(x, y);$
10: **end while**

3 Digital Straight Line Segment (DSLS) algorithm

Algorithm 2 $transition_points(r_{k-2}, r_{k-1})$

Require: $GCD(r_{k-2}, r_{k-1}) = 1$
1: $q_k = r_{k-2}$ div r_{k-1}, $r_k = r_{k-2}$ mod r_{k-1}
2: **if** $r_k \neq 0$ **then**
3: $transition_points(r_{k-1}, r_k)$
4: **if** $T_{k+1}[1] = 1$ **then**
5: $j \leftarrow 2;$
6: $T_k[1] = q_{k_init}$ $\{q_{k_init} = \langle 1, \ldots q_k, q_k + 1\rangle\}$
7: **else**
8: $j \leftarrow 1;$
9: $T_k[1] = q_{k_init}$ $\{q_{k_init} = \langle 1, 2, \ldots q_k\rangle\}$
10: **end if**
11: **for** $i \leftarrow 2; i \leq r_{k-1}; i \leftarrow i + 1$ **do**
12: **if** $i = T_{k+1}[j]$ **then**
13: $T_k[i] \leftarrow T_k[i-1] + q_k + 1;\ j \leftarrow j + 1;$
14: **else**
15: $T_k[i] = T_k[i-1] + q_k;$
16: **end if**
17: **end for**
18: **else**
19: $T_k[1] = q_{k_init}$ $\{q_{k_init} = \langle 1, 2, \ldots q_k\rangle\}$
20: **end if**

As we have shown in previous chapter, the digital segment of a straight line connecting the start point $S[0, 0]$ with the end point $E[h, v]$ may be uniquely described by the v-tuple $T_0 \equiv T_0[1], T_0[2], T_0[i], ..T_0[v]$ of x-coordinates of transition points, the variable i being

the y-coordinate of the i-th transition point. Another abstract interpretation of the v-tuple is possible as well, but this is beyond the scope of our paper. Further we suppose that the given integer numbers h and v obey the relations $h \geq v > 0$ and $GCD(h, v) = 1$. The case $GCD(h, v) = g, g \neq 1$ can be transformed into the previous case : $GCD((h \operatorname{div} g), (v \operatorname{div} g)) = 1$. Algorithm 2 generates separate elements of the v-tuple T_0. Fundamental ideas on which the algorithm is built up are:

1. The skeleton of the algorithm is the Euclid's algorithm, running in the forward part. In Fig. 2 an example is given of the Euclid's algorithm for $h = 21$ and $v = 13$.

2. The second substantial part is a cycle for, see Algorithm 2 rows $11 - 17$. The cycle is based on the idea of generalization and extension of the known fact that the digital segment contains, beside the starting and ending sections, more segments of the length $q_0 = h \operatorname{div} v$ or $q_0 + 1$, the y-coordinate being constant along each of them. Then, the number of sections of the length $q_0 + 1$ is $r_0 = h \bmod v$. r_0 – tuple T_1 contains conditions – see the row 12 of the algorithm – when either $T_0[i+1] - T_0[i] = q_0$ or $T_0[i+1] - T_0[i] = q_0 + 1$ takes place. In Fig. 2 this condition is depicted by means of pointers. If a pointer points to a given cell then the difference of the content of the cell from the preceding cell is $q_0 + 1$. If no pointer points to a cell then the difference from the previous cell is q_0.

3. The third point, also important, is the initialization of all the first cells in $T_k[1]$ for any k. When Euclid's algorithm is finished we have

$$r_{n-1} \operatorname{div} 1 = q_{n+1} \quad r_{n-1} \bmod 1 = 0. \tag{11}$$

$T_{n+1}[1]$ is equal to some element of the set

$$q_{n_init} = \langle 1, 2, \ldots q_{n+1} \rangle \tag{12}$$

see row 19 of Algorithm 2. In the remaining cases two possibilities can take place: see rows 4 to 10 of Algorithm 2. Every initialization follows from the conservation of the condition for the element $T_0[v] \leq h$.

$$T_0[v] = q_{0_init} + q_0(v-1) + \begin{cases} r_0 \\ \\ r_0 - 1 \end{cases} \tag{13}$$

(a) if the element $T_{k+1}[1]$ points to the element $T_k[1]$ then

$$q_{k_init} = \langle 1, 2, \ldots q_k, q_k + 1 \rangle \tag{14}$$

(b) if no element points to the element $T_k[1]$ then

$$q_{k_init} = \langle 1, 2, \ldots q_k \rangle \tag{15}$$

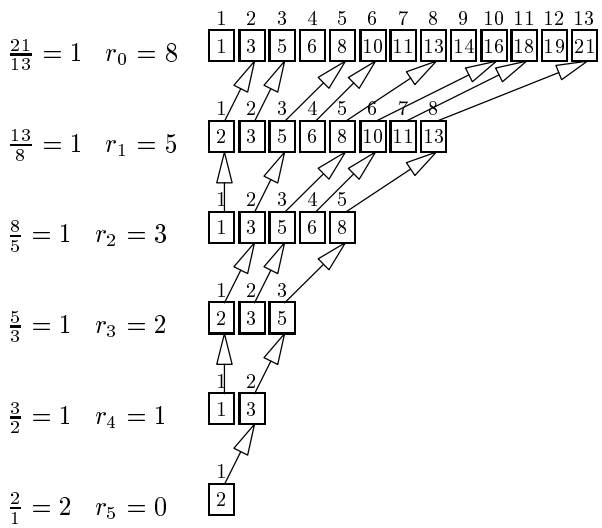

Figure 2:

Data structure of the DSLS algorithm.

In the backward running branch of Algorithm 2 we count the separate elements of the k-tuple T_k using the values in separate elements of (k+1)-tuple T_{k+1}. The contents of the separate elements of the k-tuple T_k may be interpreted as x-coordinates of transition points of digital segments interconnecting the points $S[0,0]$ and $E[r_{k-2}, r_{k-1}]$. The given algorithm may be considered as a definition of all possible digital straight line segments interconnecting the points $S[0,0]$, $E[h,v]$.

4 Conclusion

The main part of the present paper was represented by chapter 3 where the recursive DSLS algorithm is described. The input to the algorithm is formed by consists of two integer numbers, observing the conditions $h \geq v > 0$ and $GCD(h, v) = 1$. The output is formed by consists of the n of r_{k-1}–tuples. The v-tuple T_0 contains transition points representing a digital straight line segment given by the end points $S[0,0]$, $E[v,h]$. It may be easily shown that the representation by a v-tuple T_0 from the first step of the DSLS algorithm is equivalent to the chain code representation as defined in [2].
The exact proof that a v-tuple T_0 contains digital coordinates of transition points of the digitized straight line segment is beyond the scope of the present paper and can be based on different assumptions. In the following we give an outline of the two versions of these assumptions.

- It start from three basic criteria for the chain code of a straight line segment according to Freeman [2] which was improved in [4]
- It show that in accordance with the "properly

chosen criterion" the transition points are displaced "close enough" to the straight line segment $y = \frac{v}{h} \cdot x$

We can note at least that the data structure from our chapter 3 may be accepted as an abstract mathematical structure, which was defined solely by means of the DSLS algorithm. The properties of the just mentioned abstract structure are still under research.
An inverse task may be defined: If the transition points of any digital curve in the form of v-tuple are given, the problem is in finding whether the digital curve is a digital straight line segment or not. An outline of the solution of this problem follows directly from Fig. 2.

References

[1] J. E. Bresenham, "Algorithm for computer control of a digital plotter", IBM Systems Journal, Vol. 4, No.1, January 1965, pp. 25-30.

[2] H. Freeman, "Boundary encoding and processing," B.S.Lipkin and A. Rosenfeld, eds., Picture Processing and Psychopictorics, Academic Press, New York, 1970, pp. 241-266.

[3] A. Rosenfeld, "Digital straight line segments", IEEE Transaction on Computers, **C-23**, no.12, 1974, pp. 1264-1269.

[4] L.D.Wu, "On the Chain Code of a Line, " IEEE Trans. Pattern Analysis and Machine Intelligence,No. 3. Vol. 4, May 1982, pp. 347-353.

[5] M.L.P. van Lierop, C.W.A.M. van Overveld, and H.M.M van de Wetering, "Line Rasterization Algorithms That Satisfy the Subset Line Property", Computer Vision, Graphics, and Image Processing 41, 210-228 (1988)

[6] W.G. Kropatsch and H. Tockner, "Detecting the Straightness of Digital Curves in $O(N)$ Steps", Computer Vision, Graphics, and Image Processing 45, 1-21 (1989)

[7] T.Y. Kong and A. Rosenfeld, "Digital Topology: Introduction and Survey", Computer Vision, Graphics, and Image Processing 48, 357-393 (1989)

[8] D. Field, "Incremental linear interpolation", ACM Transactions on Graphics, Vol. 4, No. 1, January 1985, pp. 1-11.

[9] J. E. Bresenham, "Run length slice algorithm for incremental lines", Fundamental Algorithms for Computer Graphics (New York), Springer Verlag, 1985, pp.59-104.

[10] M.L.V. Pitteway "The Relationship between Euclid's Algorithms and Run-Length Encoding.", Fundamental Algorithms for Computer Graphics (New York), Springer Verlag, 1985, pp.105-111.

[11] C.M.A. Castle and M.L.V. Pitteway "An Application of Euclid's Algorithm to Drawing Straight Lines.", Fundamental Algorithms for Computer Graphics (New York), Springer Verlag, 1985, pp.135-139.

[12] J. Rokne and Y. Rao, "Double-Step Incremental Linear Interpolation", ACM Transactions on Graphics, Vol. 11, No. 2, April 1992, pp. 183-192.

[13] K. Y. Fung, T. M. Nicholl and A. K. Dewdney, "Run-Length Slice Line Drawing Algorithm without Division Operations," Proceedings of EUROGRAPHICS '92, Vol. 11, No. 3, 1992, pp. C-267 - C-277.

[14] P. Graham and S. S. Iyengar, " Double– and Triple–Step Incremental Linear Interpolation," IEEE Computer Graphics & Applications, Vol. 14, No. 3, May 1994, pp. 49-53.

[15] G. W. Gill, " N–Step Incremental Straight–Line Algorithms" IEEE Computer Graphics & Applications, Vol. 14, No. 3, May 1994, pp. 66-72.

[16] I.Považan and T.Hrúz "GCD-Recursive Incremental Linear Interpolation" SCCG'98 – SPRING CONFERENCE ON COMPUTER GRAPHICS AND ITS APPLICATIONS, April 23 - 25 1998 in Budmerice, Bratislava, Slovakia, in press.