STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **XLVII**, Number 1, 2002

THE CURVES ENCODING

VASILE PREJMEREAN AND SIMONA MOTOGNA

Abstract. In this paper we present an image description model that uses picture description language, and Bezier interpolation. We study 3-type images (represented by closed curves that conserve the critical points), giving modelling techniques, and also the coresponding algorithm. Our goal is to obtain a minimal description $\Pi_{\updownarrow}$-word of the smooth curves that aproximate the initial given curves.

1. Introduction

In this paper we present a method of encoding the 3-type images (according to the classification given in [7]) using $\Pi_{\updownarrow}$ -words to describe a set of critical points to which a Bezier interpolation is applied ([3,6,9]).

A 3-type image, described using a finite numbers of lines and curves, can be approximated through a 4-type image, described through a finite number of points in a cartesian rectangular system. The approximation consists of connecting the closest points to a curve $c \subset R^2$, points with integer coordinates, obtaining [5]:

$$M_{Pc}(c) = \{Apr(P) | P \in c\}, Apr : R^2 \to Z^2$$

These points can also be described by $\Pi_{\updownarrow}$-words.

Decoding means the operation of obtaining the curve closed to the initial one, and will be performed through interpolation, namely Bezier interpolation ([3,6,9]), of the points described by the $\Pi_{\updownarrow}$-words.

Bezier interpolation has been chosen because the resulting curves will be smooth, based on the two basic properties of these curves: they cross the initial and final points and are tangent to the initial and final segments determined by the first, respectively the last two points [7].

The curve we want to encode (and approximate) is divided in several curves, each of them initially described by a string of critical points, of length at most 7, according to the lemma 2.2 from [8]. In order to mark the breaking points of the curve, we will extend the description language Π^* (where the alphabet

2000 *Mathematics Subject Classification.* 68Q45, 68T10.

1998 *CR Categories and Descriptors.* I.5.2 [**Computing Methodologies**]: Pattern Recognition – *Design Methodology*.

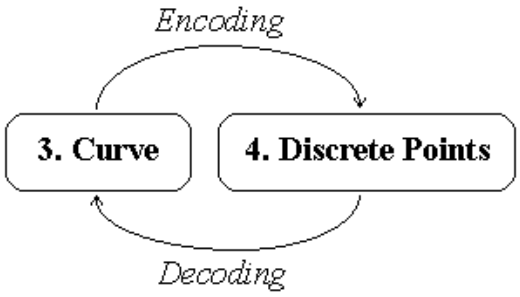

FIGURE 1. The transformations for the two image types

$\Pi = \{r, u, l, d\}$ stands for the commands *right, up, left, down*) to the language $\Pi_{|}^{*}$, where $\Pi_{|} = \Pi \cup \{|\}$, the command "|" representing the cut of the curve. These strings of length bigger than 7 will be reduced, eliminating some of them such that the curve resulted from interpolating the remaining points will preserve the critical points, thus approximating in an accurate way the given curve. The algorithm 2.1 tries to reduce the number of points such that it won't exceed 6. Since after elimination some points may no longer be neighbours, the description cannot be done through $\Pi_{|}$, and the language $\Pi_{\updownarrow} = \Pi_{|} \cup \{\uparrow, \downarrow\}$ will be needed, allowing points selection from a path traversed on the 4 directions.

2. The Set of Critical Points and Bezier Aproximation

From now on we will focus on 3-type images formed from closed curves, for example a system of level curves on a map as in Figure 2. The encoding of this image will be obtained by processing each curve; we will follow the process for the exterior curve, and the others will be treated similarly.

We will apply a rectangular network to the curve under study, such that every point from the curve (pixel from the image) will have two coordinates (x, y).

The string containing the centers of the squares that are crossed by the given curve will form the string of critical points (marked by small circles in Figure 3). The string of critical poits may start from anywhere, but it is more convenient to choose a starting point S such that together with its neighbouring points, they will be colinear. The explanation lies in the fact that S will also be the final point and the curve will be smooth, since a bezier curve is tangent to the starting and ending segments.

If this desirable situation is not possible, namely there aren't any three neighbour colinear points, then we will use a smoother network, halving the distance between horizontal and vertical lines and choosing one of the following situations:

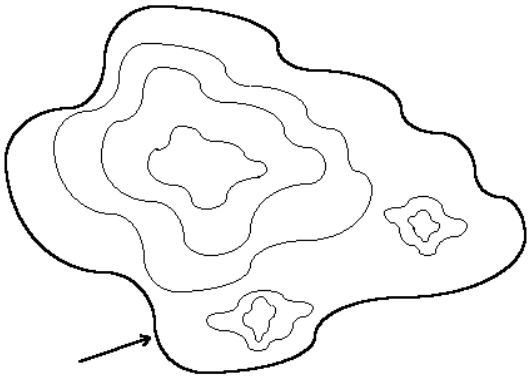

FIGURE 2. An example of a 3-type image

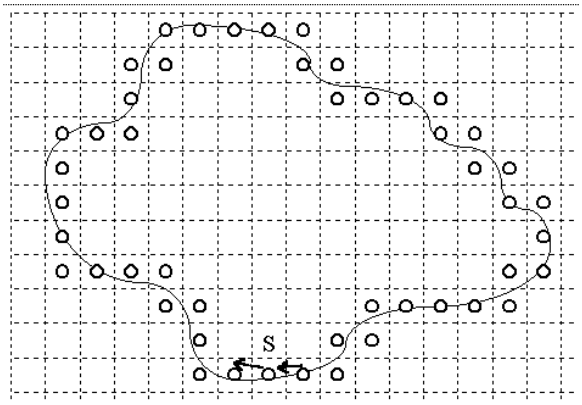

FIGURE 3. The string of critical points corresponding to a curve

a: re-compute the set of critical points and search a new starting point S;
b: insert a new point between any two neighbour points, at the middle distance between them; in this way any three succesive points will be colinear, and any point can be selected as start;

With this string of critical points we may approximate the given curve through a Bezier interpolation curve. The resulting curve (see Figure 4) will not be exactly what we've expected, since it does not approximate too well the initial curve. It does not preserve the critical points as shown in Figure 4.

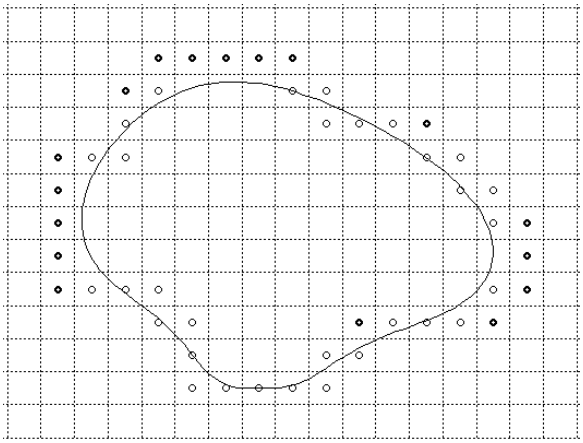

FIGURE 4. Initial curve and the one obtained through interpolation

Applying a new approximation we will get further to the initial curve C_0, although it will be desirable to obtain the same C_B, constant from now on.

Remark 2.1: One may notice that if from the set M_{P_C} of critical points we obtain the curve C_B applying a transformation T, and from the curve C_B we obtain the same set M_{P_C} of critical points applying a transformation T_1, then these two transformations represent each others inverses ($T_1 = T^{-1}$ and $T = T_1^{-1}$):

$$C_0 \to M_{P_C} \overset{\mathrm{T}}{\to} C_B \overset{\mathrm{T}^{-1}}{\to} M_{P_C} \overset{\mathrm{T}}{\to} C_B ...$$

The possibility of reducing the number of critical points is studied for each subcurve (the example from Figure 5 studies the curve between S_4 and S_1). In figure 6 you may notice that removing the 4 critical points denoted by "x" we obtain a Bezier approximation curve that satisfies the two proposed properties.

In order to obtain the desired approximation curve we will apply *Bezier* algorithm for each subsequence, and the resulting curve is obtained unifying the s curves (in Figure 5 we have 4 curves divided by the points $S_1, ..., S_4$ determined by these subsequences: $C_B(P) = Bezier(P_1^1, ..., P_{n1}^1) \cup Bezier(P_1^2, ..., P_{n2}^2) \cup ... \cup Bezier(P_1^s, ..., P_{ns}^s)$, where $M_{P_C}^k = \{P_1^k, ..., P_{nk}^k\}$ is the set of critical points obtained for the curve k applying algorithms similar to the ones we will describe below.

The result can be further improved if we can reduce even more the number of points, even if we choose other points, not just removing the initial ones. In the following, we will study this posiibility, trying to reduce the number of remaining points (the nine points from figure 6 can be reduced to the five points from figure

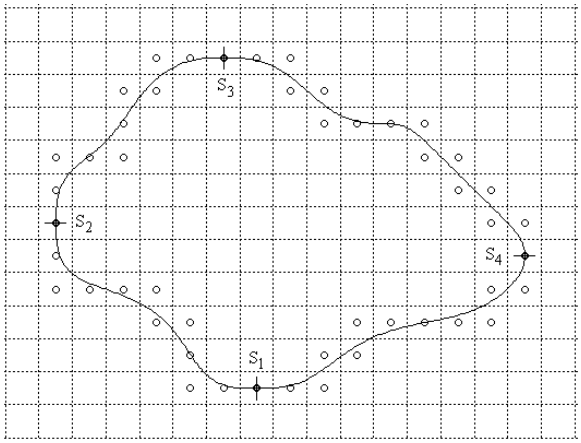

FIGURE 5. Dividing the curve

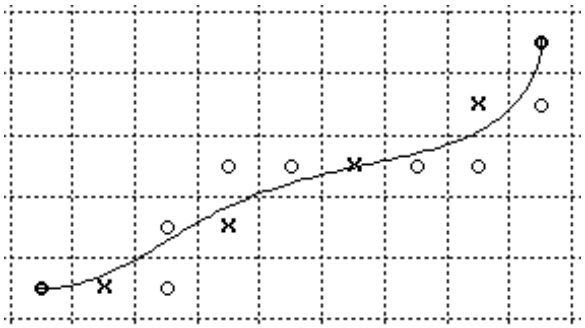

FIGURE 6. Reducing the number of critical points

7, denoted $P_1, P_2, ..., P_5$. On this specific example, the curve description can be obtained using only five interpolation points.

There are some remarks to be made analyzing figure 7: the initial point P_1 and the final point P_5 must be preserved, the second point P_2 and the penultimate point P_4 must be on the same direction with the old points (in order to satisfy the smoothness property). The point P_3 had been choosen such that the obtained curve preserves the critical points (meaning also that be obtain a valid approximation of the given curve).

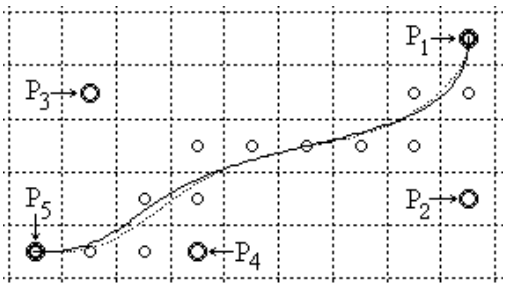

FIGURE 7. Approximation using 5 points

Algorithm 2.1

The following algorithm tries to find the points $P_1, ..., P_n$ that describe the Bezier approximation curve corresponding to the given critical points $Q_1, ..., Q_m$, starting from 2, 3,..., 6 points. If we don't succeed, the curve must be divided again (recommended to).

Begin
Input $Q_1, Q_2, ..., Q_m$;
if (Q_1, Q_m) *can describe the curve* **then** n:=2;
Output Q_1, Q_m
else
if *exists a point R such that* (Q_1, R, Q_m) *can describe the curve*
then n:=3;
Output Q_1, R, Q_m
else
if *exist 2 points R,S such that* (Q_1, R, S, Q_m) *can describe the curve*
then n:=4;
Output Q_1, R, S, Q_m
else
n:=5;
$P_1 := Q_1; P_n := Q_m$;
Find (P_2 *on direction* $Q_1 \to Q_2$) **and** (P_{n-1} *on direction* $Q_m \to Q_{m-1}$) **and**
Find (P_3 *in the domain* (P_1, P_n) *depending on* (P_2, P_{n-1}));
if $Ok(Q_1, Q_2, ..., Q_m, P_1, P_2, ..., P_n))$ **then Output** $P_1, ..., P_n$
else
n:=6;
$P_1 := Q_1; P_n := Q_m$;
Find (P_2 *on direction* $Q_1 \to Q_2$) **and** (P_{n-1} *on direction* $Q_m \to Q_{m-1}$) **and**
Find (P_3 *and* P_4 *in the domain* (P_1, P_n) *depending on* (P_2, P_{n-1}));
if $Ok(Q_1, Q_2, ..., Q_m, P_1, P_2, ..., P_n))$ **then Output** $P_1, ..., P_n$

else write('The curve must be divided')

End.

We present an analysis of the performed steps:

a) For $n = 2$ - it is very simple, because it is easy to verify if the segment Q_1Q_m is horizontal or vertical and traverse the critical point.

b) For $n = 3$ - we must verify if the triangle Q_1RQ_m is rectangular or not, and the critical points are preserved (there are two possibilities $R(x_1, y_m)$ or $R(x_m, y_1)$).

c) For $n = 4$ - is not dificult to find the points $R(= P_2)$ and $S(= P_3)$, because R must have the coordinates (x_1, y) or (x, y_1) and S must have the coordinates (x_m, y) or (x, y_m), where $x \in (x_1, x_m)$ and $y \in (y_1, y_m)$. There are the four possibilities presented in figure 8.

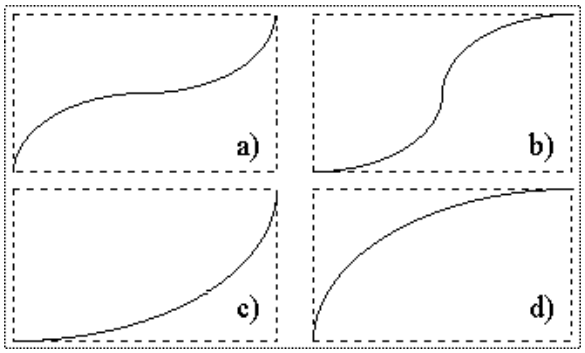

FIGURE 8. The 4 cases using 4 approximation points

d) For $n = 5$ - one can easy see that if we use only the four points P_1, P_2, P_4, P_5 to approximate the curve from figure 7 we obtain a curve like the one from figure 8 c). Then, we need a point (like P_3, from figure 7) to "drag" the curve. Finally, the points P1, P2, P3, P4, P5 approximate correctly the given curve. If we consider the case c) from figure 8, finding $P_2(x_1, y)$ on direction $Q_1 \rightarrow Q_2$ means to find a value $y < y_1$, because Q_1 have the coordinates (x_1, y_1) and Q_2 have the coordinates $(x_2 = x_1, y_2)$ and analogues, finding $P_{n-1}(x, y_m)$ on direction $Q_m \rightarrow Q_{m-1}$ means to find a value $x > x_m$, because Q_m have the coordinates (x_m, y_m) and Q_{m-1} have the coordinates $(x_{m-1}, y_{m-1} = y_m)$. The search of the desired points P_1 and P_{n-1} is performed alternatively increasing (or decreasing) the coordinate y starting from the initial value $y_1 \pm 1$, and the coordinate x starting from $x_m \pm 1$ until the desired points are obtained or an imposed maximum value is overflow (i.e., x_1, respectively y_m). Finding $P_3(x, y)$ in the domain $(P_1(x_1, y_1), P_n(x_m, y_m))$ means to find a value x such

that $x_m < x < x_1$ and a value y such that $y_m < y < y_1$ (for our case c) from figure 8).

e) For $n = 6$ - the points P_1, P_2, P_5, P_6 will be constructed like P_1, P_2, P_4, P_5 (see case n=5) and after that we need two points $P_3(x, y)$ and $P_4(x', y')$ in the domain $(P_1(x_1, y_1), P_n(x_m, y_m))$ with the same properties like above (like P_3 for n=5).

f) For $n > 6$ - it is more efficient to cut the curve and to apply this procedure on each of the parts.

The number of curves in which the initial curve is decomposed can be reduced in the following way (as in Figure 9):

- eliminating some cutting points: existing colinear points that would solve the problem correctly;
- searching certain critical points even outside the domain (less points).

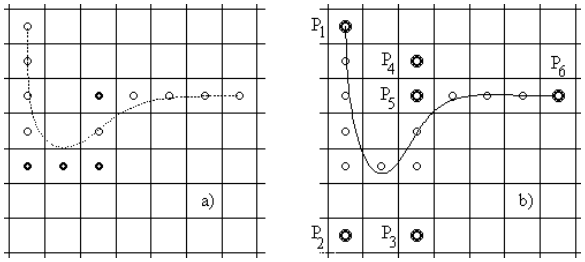

FIGURE 9. The curves obtained with 6 points

As shown in Figure 9a, if during interpolation we consider all points, then the obtain curve does not preserve the critical points (the three colinear points from the bottom), but the six points from Figure 9b preserve the curve.

The two interpolation pointsfrom the bottom are needed in order to "drag" the curve through the uncovered critical points. The point P_3 is inside the domain and is needed in order to cover the critical point that remained uncoverd by the curve from figure 9a.

Searching the points P_2, P_3, P_4, P_5 is quite a difficult problem, if we take into account the eficiency of the algorithm, because the points P_2 and P_5 must be searched in the form: $P_2(x_1, y_1 + Dy), P_5(x_m - Dx, y_1)$ and the points $P_3(x_3, y_3)$ and $P_4(x_4, y_4)$ must be searched in the domain:

$$(min(X), max(X)) \times (min(Y), max(Y)),$$

$$\text{where } X = \{x_1, x_2, x_5, x_6\} \text{ and } Y = \{y_1, y_2, y_5, y_6\}$$

Then, the searching algorithm has the following structure:

Algorithm 2.2

(1) **for** Dx:= 1 **to** $LimX$ **do**
(2) **for** Dy:=1 **to** $LimY$ **do**
(3) **for** x_3 := *min(X)* **to** *max(X)* **do**
(4) **for** y_3 := *min(Y)* **to** *max(Y)* **do**
(5) **for** x_4 := *min(X)* **to** *max(X)* **do**
(6) **for** y_4 := *min(Y)* **to** *max(Y)* **do**
(7) **if** *Ok(P,Q)* **then output** P

where $\mathbf{P} = (P_1, P_2, ..., P_6), \mathbf{Q} = (Q_1, Q_2, ...Q_m)$

This algorithm determines all solutions, but is inefficient due to the six loops. The optimizing of this algorithm can take into consideration that we may be satisfied with only some of the solutions or even with one solution in exachang to a more efficient search.

A first direction for optimization is to unify the loops from lines 3 and 4, respectively from lines 5 and 6. In this way, the search of the points P_3, P_4 is restricted on the directions in which the critical points haven't been covered (in our example, to the rigth, respectively upwards). Now, the lines 3 to 6 will be replaced with:

For Dr:=1 To Lr Do
For Du:=1 To Lu Do

In this situation, in our example, the points P_3 and P_4 will be search in the form $P_3(x_2 + Dr, y_1)$, respectively $P_4(x_5, y_5 - Du)$. Of course, applying such a strategy does not assure that all solutions will be obtained. This example will generate two solutions: the one in figure 9b and from figure 10a. The other three solutions from figure 10 (b,c,d) are not on that direction and we have to extend the search to a neighbourhood (inside the domain) of the points P_3 and P_4, if we haven't obtained any solution.

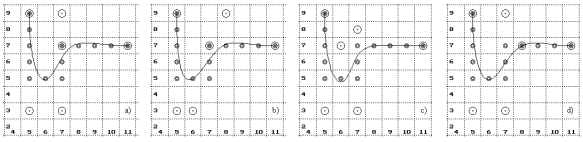

FIGURE 10. Other curves obtained with 6 points

In order to appreciate how close a Bezier curve $Bezier(P)$ is to the set of critical points Q, we define the distance $\delta(P, Q)$ as:

$$\delta(P, Q) = Nr.Minus + Nr.Plus$$

where:

- $Nr.Minus$ represents the number of critical points uncovered by the approximation curve $Bezier(P)$

- $Nr.Plus$ represents the number of extra points covered by the approximation curve ($\notin Q$)

Then, the distance $\delta(P, Q)$ is the cardinal of the simetric difference (Δ) between the set of *representative* points for the Bezier curve (C_B) corresponding to the determined points P) and the set of given critical points ($M_{P_C} = Q$):

$$\delta(P, Q) = |\{Apr(B)|B \in Bezier(P)\} \ \Delta \ Q|$$

For example, if we consider the approximation from figure 11, obtained for $Dx = 4(P_5(7, 7)), Dy = 5(P_2(5, 4)), Dr = 2(P_3(7, 4))$ and $Du = 1(P_4(7, 8))$, the distance is $\delta(P, Q) = 2 + 1 = 3$, since the points (5,5) and (7,5) are not covered, and the point (6,6) should not be included.

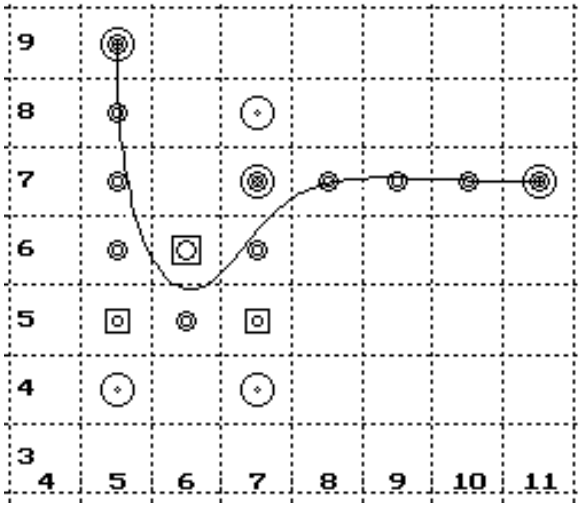

FIGURE 11. An example of approximation

If we study the minimal values obtained for the pairs (Dy, Dx) from table 2.1 we can notice that for $(Dy, Dx) \in \{(6, 3), (6, 4), (6, 5)\}$ there exist solutions, because the minimal values for the distance δ are zero. These minimal values are obtained choosing P_3 in the neighbourhood of P_2, respectively choosing P_4 in the neighbourhood of P_5. Another remark is that these values are grouped in a zone, to which if we get farther, then the values increase.

This remark gives the possibility to limit the values $LimX$ and $LimY$ from the loops (1) and (2), since the more we get farther from the zero positions, the more the values of distances increase.

Even more, if we re not interesting in obtaining all solutions and one soltuion is enough (as the approximation problem was initially stated), for example the

Min(δ)	*Dx:* **1**	**2**	**3**	**4**	**5**	**6**
Dy: **1**	12	10	10	8	8	8
2	8	8	8	6	6	6
3	6	6	6	6	6	6
4	4	4	4	4	5	5
5	6	4	3	2	3	4
6	2	2	0	0	0	4
7	8	3	2	1	1	3

TABLE 1. The distances obtained for the given curve

position (6,3), then the search should not parse the entire matrix, but the domain $(1,1) \times (6,3)$, so $LimX = 3$ and $LimY = 6$. The search can be performed succesively adding square matrixes, in the order given in table 2.2. One may notice that in the fifth column (or even earlier) the values of δ increases upwards, so it is possible to quit searching the solution in that zone and move to the next line.

Searching order	*Dx:* **1**	**2**	**3**	**4**	**5**	**6**
Dy: **1**	1	4	9	16	25	-
2	2	3	8	15	24	-
3	5	6	7	14	23	-
4	10	11	12	13	22	-
5	17	18	19	20	21	-
6	26	27	**28**	...	...	-
7	-	-	-	-	-	-

TABLE 2. The searching order - first version

Of course, a *Branch and Bound* algorithm will be more suitable, especially since we have already defined a distance δ to the solution (if $\delta(P,Q) =)$ then P is a solution). We will characterize a state through the components Dx, Dy, d_1 and d_2. The ditance d_1 represents the computation step, and $d_2 = \delta(Dx, Dy)$. The initial state will be $(1,1,1,\delta(1,1))$, and the final state will be characterized by $\delta = 0(d_2 = 0)$.

From a current state (choosen from the set of active states, where the minimum of the sum $d_1 + d_2$ is obtained) we will generate two active states $(Dx+1, Dy, d_1 + 1, \delta(Dx+1, Dy))$ and $(Dx, Dy+1, d_1+1, \delta(Dx, Dy+1))$, and this current state will become pasive. The steps to be performed can be studied in table 2.3, and it is also easy to remark that the solution will be obtained in 8 steps. If the set of active states become empty, such that no new selection of a current state can

be performed, then the problem has no solution. In order to end the algorithm in such a case we must specify a limit $LimX$ for Dx and a limit $LimY$ for Dy.

Searching order	*Dx:* **1**	**2**	**3**	**4**	**5**	**6**
Dy: **1**	**1**+12	2+10	10	8	8	8
2	**2**+8	3+8	8	6	6	6
3	**3**+6	4+6	6	6	6	6
4	**4**+4	**5**+4	6+4	4	5	5
5	5+6	**6**+4	7+3	2	3	4
6	2	**7**+2	**8+0**	0	0	4
7	8	8+3	2	1	1	3

TABLE 3. The searching order - *Branch and Bound* approach

A last remark is that in the case in which there are more than one active state with the same minimal value for $d_1 + d_2$, we will choose the one for which d_2 is minimal. As shown in table 2.3, at the step 7 we have "preferred" the state (2,5,6,4) (or (3,4,6,4)) although the state (2,3,4,6) has the same value for $d_1 + d_2 = 10$, since is closer to the solution (d_2 approximates the distance between a state and the final state). Even if we would have choosen the other state, the algorithm gives the desired solution, but an extra step would have been performed, and then come back and choose the "preferred" state.

3. Using Π-words to describe pictures of 3-type

In this section, we present a model of approximating a curve using a picture description language. At the beginning, we will use the alphabet $\Pi = \{r, u, l, d\}$ to describe the movement of the pen on the four directions, then we will extend the commands alphabet to $\Pi_| = \Pi \cup \{|\}$ (where | denotes an interuption of the sequence ofcritical points) in order to define the cuts of a curve; eventually, this alphabet will be enhanced with two more commands *pen-up* and *pen-down* $\Pi_\updownarrow = \Pi_| \cup \{\uparrow, \downarrow\}$, in order to avoid (eliminate) certain points, reducing the number of critical points that will be interpolated obtaining an approximation curve.

If we consider the example from figure 5, then the Π-word $w \in \Pi^*$ is:

$$\boldsymbol{w = lluulullluu\ uurruururr\ rrdrdrrrdrdrdrd\ dldlllldldll}$$

The insertion positions, corresponding to the possible interuption points (points between two colinear points), can be anywhere between two identical commands(characters). If this is not possible or is not convenient, then the word w can be easily modified doubleing each character. In this manner, the curve can be described starting from any point and can be interrupted at any moment, since any three consecutive points are colinear. The description word becomes:

$$w' = l^4u^2(u^2l^2)^2l^4u^8r^4u^2(u^2r^2)^2r^6(d^2r^2)^2r^4(d^2r^2)^3d^4l^2d^2l^8(d^2l^2)^2l^2.$$

Since our example doesn't need any network or command word doubleing, we will work from now on with the word w (not w').

Since the command sequence is circular [1], we can execute it from any position, and then come back to the beginnign of the string and execute the remaining commands. The start command will be a character that has a predecessor (left neighbour) equal to it (the points must be colinear).

Even more, for a selected string, marked with interruption commands, denoted | and that will cut the string into substrings, corresponding to the curves that forms the given closed curve; for example, the $\Pi_{|}$-word that describe the curve from figure 5 is

$\boldsymbol{w = llulullluu|uurruururr|rrdrdrrxrdrdrdrd|dldllldldll}$

we can introduce avoiding commands (eliminating critical points), that will preserve the quality of the curve. (the smoothness and the critical points).

We can use the alphabet $\Pi_{\updownarrow}$ that allows the construction of $\Pi_{\updownarrow}$-words (containing two more symbols $\uparrow = pen - up$ and $\downarrow = pen - down$) : $\Pi_{\updownarrow} = \{r, u, l, d, |, \uparrow, \downarrow\}$. These words will describe a sequence of points that determines a curve by Bezier interpolation. This means that a set of $\Pi_{\updownarrow}$-words describes curves that compose a 3-type image. The $\Pi_{\updownarrow}$-word for the curve from figure 6 is $d \uparrow l \downarrow dl \uparrow l \downarrow ll \uparrow d \downarrow ld \uparrow l \downarrow l$, and for the curve from figure 7 the Π-word is $\uparrow d^2 \downarrow d \uparrow l^7 u \downarrow u \uparrow d^3 r \downarrow r \uparrow l^2 \downarrow l$. An even simpler description convention can be used, if the points are rare, as in our example, if we remove the sequence that denote a point of the form $\downarrow \tau \uparrow$ (where $\tau \in \Pi$) and replace it with a single command character (for example $\downarrow$), that attach the current point to the sequence of interpolation points. This means that, implicitely, the movement of the pen is done "without drawing", and when a $\downarrow$ command is met, the current point will be stored. We could also consider that these characters are present at the beginning and at the end of the description word. In the example from figure 7, the description word may be $\downarrow d^3 \downarrow l^7 u^2 \downarrow d^3 r^2 \downarrow l^2 \downarrow l^3 \downarrow$ or if we give up the first and last character, we obtain a reduced description: $w = d^3 \downarrow l^7 u^2 \downarrow d^3 r^2 \downarrow l^2 \downarrow l^3$.

Reducing the number of critical points also implies the preserving of the initial points, so the fixed point problem for a new approximation (curves convergence) is solved.

REFERENCES

1. F.J. Brandemburg, M.P. Chytil, On Picture Languages : Cycles and Syntax - Directed Transformations, Technische Berichte der Fakultat fur Mathematik und Informatik Universitat Passau, MIP-9020, 1990.
2. J. Dassow, F. Hinz, Decision problems and regular chain code picture languages, Discrete Applied Mathematics, no.45, 1993, pp. 29-49.
3. J.D. Foley, A.V. Dam, Fundamentals of Interactive Computer Graphics, Addison Wesley, London, 1982.

4. H.A. Maurer, G. Rozenberg, E. Welzl, Using String Languages to Describe Picture Languages, Information and Control, Vol.54, Nr.3, 1982, pp.115-185.
5. S.Motogna, V.Cioban, V.Prejmerean, Picture Approximation, Studia Univ. Babes-Bolyai, Vol.XLIII, Nr.2, 1998, pp.43-55.
6. T. Pavlidis, Algorithms for Graphics and Image Processing, Springer-Verlag, Berlin-Heidelberg, 1982.
7. A. Rosenfeld, Picture Processing by Computer, Academic Press, London, 1969.
8. I.H. Sudborough, E. Welzl, Complexity and Decidability for Chain Code Picture Languages, Universitat Graz, F125, 1983.
9. A. Watt, 3D Computer Graphics, Addison-Wesley, Great Britain, 1993.

Babeş-Bolyai University, Faculty of Mathematics and Computer Science, RO 3400 Cluj-Napoca, Str. Kogălniceanu 1, Romania
E-mail address: per|motogna@cs.ubbcluj.ro