PACE: Polygonal Approximation of Thick Digital Curves Using Cellular Envelope

Partha Bhowmick[1], Arindam Biswas[1], and Bhargab B. Bhattacharya[2]

[1] Computer Science and Technology Department
Bengal Engineering and Science University, Shibpur, Howrah, India
{partha, abiswas}@becs.ac.in
[2] Advanced Computing and Microelectronics Unit
Indian Statistical Institute, Kolkata, India
bhargab@isical.ac.in

Abstract. A novel algorithm to derive an approximate cellular envelope of an arbitrarily thick digital curve on a 2D grid is proposed in this paper. The concept of "cellular envelope" is newly introduced in this paper, which is defined as the smallest set of cells containing the given curve, and hence bounded by two tightest (inner and outer) isothetic polygons on the grid. Contrary to the existing algorithms that use thinning as preprocessing for a digital curve with changing thickness, in our work, an optimal cellular envelope (smallest in the number of constituent cells) that entirely contains the given curve is constructed based on a combinatorial technique. The envelope, in turn, is further analyzed to determine polygonal approximation of the curve as a sequence of cells using certain attributes of digital straightness. Since a real-world curve/curve-shaped object with varying thickness and unexpected disconnectedness is unsuitable for the existing algorithms on polygonal approximation, the curve is encapsulated by the cellular envelope to enable the polygonal approximation. Owing to the implicit Euclidean-free metrics and combinatorial properties prevailing in the cellular plane, implementation of the proposed algorithm involves primitive integer operations only, leading to fast execution of the algorithm. Experimental results including CPU time reinforce the elegance and efficacy of the proposed algorithm.

1 Introduction

The subject on properties, characterizations, and representations of digital curves (DC) has been researched continuously since the debut of digitization of graphical objects and visual imageries [1], [2]. Nevertheless, in the abundance of various problems and their algorithms related with digital objects, polygonal approximation of a digital curve/object has received special attention for its efficient representation and for its potential applications in connection with analysis of digital images [3], [4], [5]. The set of straight edges of the concerned polygon carries a strong geometric property of the underlying objects, which can be used for efficient high level description of the objects and for finding the similarity among different objects in the digital plane.

Since an optimal solution of polygonal approximation targeted to minimize the number of vertices, and space thereof, is computationally intensive, several heuristic and meta-heuristic approaches based on certain optimality criterion have been proposed over the last few decades, and some of these that have come up in recent times may be seen in [6], [7], [8], [9], [10], [11], etc. Further, there also exist various studies and comparisons of the proposed techniques, e.g., [12], [10], [13], to cite a few. This entire collection of polygonal approximation algorithms, however, consider the input digital curve to be strictly "irreducible"[1] (and connected thereof), failing which the algorithm may produce undesired results pertaining to polygonal approximation.

Hence, in case of a thick DC, thinning is required to ensure the property of "irreducibility" to the input DC so that it can qualify for the subsequent process of polygonal approximation. A thinning procedure, being plagued by asymmetric erosion in thick regions and shifting of junction/end points, and being liable to slow down the overall run time of the approximation process, is susceptible to deteriorate the results of approximation. Furthermore, the result goes on worsening if there occurs some missing grid points (pixels) in the input DC — which splits, therefore, into multiple DC's — producing several approximate polygons instead of a single polygon, thereby giving rise to misleading impression, and more specifically, posing severe problems in the subsequent applications. These problems have been tackled in our method using the novel concept of cellular envelope of an arbitrary digital curve whose thickness may vary non-uniformly. In our method, we consider that all possible thicknesses — including 0 (missing pixel) and 1 (one pixel thick) — may occur in a DC[2] when it is subject to polygonal approximation. The idea of outer and inner boundaries of polygonal regions is also present in rounding the intersection of two polygonal regions [14] and simplification envelopes [15].

A brief outline of the paper is as follows. In Sec. 2, we present a combinatorial algorithm to derive the cellular envelope of an arbitrary DC (stage I) using its inner and outer isothetic polygons [16], [17], [18], [19]. Sec. 3 enumerates some digital geometric properties of cellular straight segments (CSS), followed by the motivation and underlying principle for their extraction (stage II) from the cellular envelope of the input DC obtained in stage I. In Sec. 4, we present our method **PACE** and the two algorithms corresponding to stage I and stage II. Sec. 5 exhibits some test results on a curve-shaped object with varying curve thickness. Finally in Sec. 6, we summarize its strength and point out the future scope of improvements.

---

[1] A digital curve $\mathcal{C}$ is said to be "irreducible" if and only if removal of any grid point p in $\mathcal{C}$ makes $\mathcal{C}$ disconnected.

[2] Henceforth, in this paper, we use the term "DC" to denote a digital curve (reducible or irreducible) as well as a curve-shaped object that may contain multiple disconnected segments producing the *impression* of a single object.

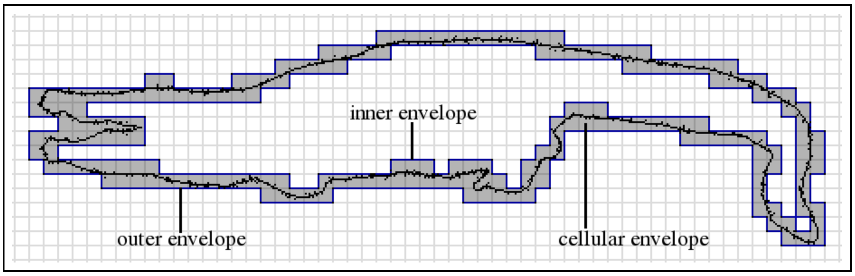


Fig. 1. Cellular envelope $\mathcal{E}(\mathcal{C},\mathcal{G})$ of a real-world (thick, rough, and reducible) curve-shaped object $\mathcal{C}$ for cell size $g = 8$

2 Cellular Envelope

If $\mathcal{C}$ be a given DC, and $\mathcal{G} = (\mathcal{H},\mathcal{V},g)$ be a set of uniformly spaced horizontal grid lines ($\mathcal{H}$) and vertical grid lines ($\mathcal{V}$) with spacing g, then the cellular envelope of $\mathcal{C}$, corresponding to the cellular plane defined by $\mathcal{G}$, is given by

$$\begin{aligned}\mathcal{E}(\mathcal{C},\mathcal{G}) &= \mathcal{E}_{out}(\mathcal{C},\mathcal{G}) \setminus \mathcal{E}_{in}(\mathcal{C},\mathcal{G}) \text{ if } \mathcal{C} \text{ is closed or contains a closed part,}\\ &= \mathcal{E}_{out}(\mathcal{C},\mathcal{G}) \text{ if } \mathcal{C} \text{ is open,}\end{aligned} \tag{1}$$

where $\mathcal{E}_{out}(\mathcal{C},\mathcal{G})$ and $\mathcal{E}_{in}(\mathcal{C},\mathcal{G})$ represent the respective outer and inner envelopes of $\mathcal{C}$ w.r.t. $\mathcal{G}$, such that (i) each point $p \in \mathcal{C}$ should lie inside $\mathcal{E}_{out}(\mathcal{C},\mathcal{G})$ and outside $\mathcal{E}_{in}(\mathcal{C},\mathcal{G})$; (ii) each vertex of $\mathcal{E}(\mathcal{C},\mathcal{G})$ (and of $\mathcal{E}_{out}(\mathcal{C},\mathcal{G})$ and $\mathcal{E}_{in}(\mathcal{C},\mathcal{G})$, thereof) is in $\mathcal{G}$; and (iii) area of $\mathcal{E}(\mathcal{C},\mathcal{G})$ is minimized.

The cellular envelope of a DC (curve-shaped object) $\mathcal{C}$, which is rough, not irreducible, and disconnected (since it has uneven thickness and stray pixels) has been shown in Fig. 1. Note that, the cellular envelope $\mathcal{E}(\mathcal{C},\mathcal{G})$ shown in this figure is for cell size $g = 8$, and the envelope "tightly encloses" all the points of $\mathcal{C}$ with no points lying outside $\mathcal{E}(\mathcal{C},\mathcal{G})$.

2.1 Combinatorial Properties of a Cellular Envelope

Let $\mathcal{I}$ be the 2D image plane having height h and width w, and containing the entire object $\mathcal{C}$. Let $\alpha(i,j)$ be the point of intersection of the horizontal grid line $l_H : x = i \in \mathcal{H}$ and the vertical grid line $l_V : y = j \in \mathcal{V}$. Let $\mathcal{S}_{\text{LT}}$, $\mathcal{S}_{\text{RT}}$, $\mathcal{S}_{\text{LB}}$, and $\mathcal{S}_{\text{RB}}$ be the respective left-top, right-top, left-bottom, and right-bottom square cells with the common grid point $\alpha(i,j)$, and let $\alpha'(i,j+g)$ and $\alpha''(i+g,j)$ be the respective grid points lying immediate right and lying immediate below α, as shown in Fig. 2. We construct a binary matrix A_e (*edge matrix*) that contains $\big((w/g)(h/g+1)\big) \times \big((h/g)(w/g+1)\big)$ entries, each entry being in one-to-one correspondence with a particular edge of a particular cell. If an edge $e(\alpha,\beta)$ connecting two neighbor grid points α and β is intersected by the object $\mathcal{C}$, then the corresponding entry in A_e is '1', otherwise '0'.

Now, from A_e, we construct another binary matrix A_c (*cell matrix*) of size $(h/g) \times (w/g)$, in which each entry corresponds to a unique cell — the entry is

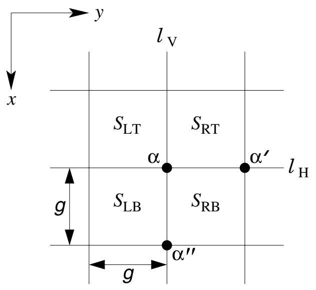


Fig. 2. Four cells with common vertex α

'1' if at least one of the four edges of the concerned cell is intersected by the object $\mathcal{C}$, and is '0' otherwise — which is checked from the correspondence of its edge information in A_e.

Next, the candidature of α as a vertex of the (inner or outer) envelope is checked by looking at the combinatorial arrangements (w.r.t. object containments) of the four cells having common vertex α. There exist $2^4 = 16$ different arrangements of these four cells, since each cell has 2 possibilities ('0'/'1'). These 16 arrangements can be further reduced to 5 cases, where, a particular case $\mathbf{C}_q$, $q = 0, 1, \ldots, 4$, includes all the arrangements where exactly q out of these four cells has/have object containments (i.e., contain(s) parts of the object $\mathcal{C}$), and the remaining (i.e., $4 - q$) ones have not. That is, the case in which the sum of the 4 bits in the corresponding entries in A_c is equal to q is represented by $\mathbf{C}_q$. Further, out of these 5 cases, cases $\mathbf{C}_1$ and $\mathbf{C}_3$ *always* and case $\mathbf{C}_2$ *conditionally* produce vertices of the inner/outer envelope, as explained below.

Case $\mathbf{C}_1$. $\binom{4}{1} = 4$ arrangements are possible where only one cell with vertex α contains $\mathcal{C}$, i.e., exactly one of the corresponding four entries in A_c is '1' and each other is '0'. The envelope will have its one edge ending at α and the next edge starting from α. Hence, if α lies inside $\mathcal{C}$, then it is a 270^0 vertex of $\mathcal{E}_{in}(\mathcal{C}, \mathcal{G})$, and if α lies outside $\mathcal{C}$, then it is a 90^0 vertex of $\mathcal{E}_{out}(\mathcal{C}, \mathcal{G})$ (the angle $90^0/270^0$ of a vertex means its internal angle in the corresponding envelope).

Case $\mathbf{C}_2$. $\binom{4}{2} = 6$ arrangements are possible in which exactly two of the four cells contain $\mathcal{C}$. If the cells containing $\mathcal{C}$ are diagonally opposite (2 out of 6 arrangements), then α is a vertex (90^0 for $\mathcal{E}_{in}(\mathcal{C}, \mathcal{G})$ and 270^0 for $\mathcal{E}_{out}(\mathcal{C}, \mathcal{G})$); otherwise α is an ordinary point on the envelope perimeter.

Case $\mathbf{C}_3$. $\binom{4}{3} = 4$ arrangements are possible for $q = 3$, where, out of the four cells, only one cell is free. In each such arrangement, α would be a 90^0 vertex for $\mathcal{E}_{in}(\mathcal{C}, \mathcal{G})$) and a 270^0 vertex for $\mathcal{E}_{out}(\mathcal{C}, \mathcal{G})$).

For case $\mathbf{C}_0$: $\binom{4}{0} = 1$ arrangement, α is just an ordinary grid point lying outside $\mathcal{E}_{out}(\mathcal{C}, \mathcal{G})$ or inside $\mathcal{E}_{in}(\mathcal{C}, \mathcal{G})$), whereas for case $\mathbf{C}_4$: $\binom{4}{4} = 1$ arrangement, α is a grid point included in $\mathcal{C}$ (since no two traversable edges are incident on it).

3 Cellular Straight Segments

There exist several works on constructs, properties, and applications of cell complexes and cellular straight segments (CSS), e.g., [20], [21], [22], [2], [23], in which the primal as well as many alternative definitions of CSS are found. For example, as indicated in [2], a CSS **C** can be defined as the minimal set of cells c specified by a real straight line segment **L** such that

$$\mathbf{L} \cap c \neq \emptyset, \ \forall\, c \in \mathbf{C}; \tag{2}$$

$$\text{and } \ \mathbf{L} \subset \mathbf{C}, \tag{3}$$

which makes its primal definition.

Another definition of CSS involving the Euclidean metric space is given in [20], in which it has been shown that a cellular curve **C** is a CSS if and only if there exists a direction θ and a pair of (parallel) lines in the real plane (tangential to and) containing **C**, such that the distance between, and measured in the direction (say, $\theta_\perp$) perpendicular to, this pair of lines does not exceed the distance (along $\theta_\perp$) between the closest pair of parallel lines containing the square formed by $(2 \times 2 =)\, 4$ cells sharing a common vertex.

In a recent work [21], an Euclidean-free definition of CSS has been given in terms of "fully partitioned (finite) strings" ($S^{(0)}$) and "higher order derived strings" ($S^{(j)} : j \geq 1$), the latter being derived iteratively from the preceding string (i.e., $S^{(j-1)}$) by replacing the majority symbol substrings of $S^{(j-1)}$ by its length, and by deleting the minority symbols of $S^{(j-1)}$. Subsequently, it has been shown that a string $S\,(= S^{(0)})$ represents a CSS, provided the jth order derived string of S exists for all $j \geq 0$.

Alternatively, in the perspective of digital straightness, if we consider the center points of these edge-connected cells as grid points, then it follows that a family of cells is edge-connected if and only if the set of center points of these cells is 4-connected. Thus CSS provides a suitable option — apart from that provided by digital straight line segments (DSS) [24] — for adjudging the straightness of a curve in the digital plane, as indicated in a contemporary work [2]. A linear off-line algorithm for CSS recognition, based on convex hull construction, is briefly sketched in [22]. In our work, we have designed an on-line algorithm to derive the set of CSS's from the cellular envelope of a curve-shaped object, which cannot be subject to direct DSS extraction/polygonal approximation due to its inherent nature of possessing varying thickness, as mentioned in Sec. 1.

We have considered the center of each cell for extracting the longest line segment iteratively in (a part of) a cellular envelope $\mathcal{E}(\mathcal{C}, \mathcal{G})$ corresponding to the given curve $\mathcal{C}$ and given cell size g imposed by the grid $\mathcal{G}$. We have used some digital geometric properties of DSS formulated and explained in [2], [24]. Before explaining our algorithm, the DSS properties (defined w.r.t. chain codes [25]) relevant to our work, which were established in [24], and later (see [2]) correlated with the other straightness options such as cellular straightness, are mentioned below.

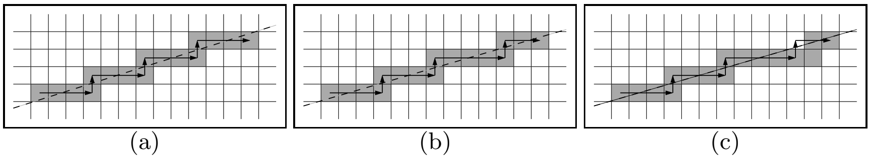


Fig. 3. Examples of cellular curves considered to explain the significance of straightness properties (R1)–(R4). Note that the directed path that traces the ordered set of centers of the cells shows the digital curve (DC) corresponding to a cellular curve. The curves in (a) and (b) are CSS's (the dashed lines show the corresponding real lines); but the curve in (c) is not, since there does not exist any real line that can pass through the set of cells defining this curve (see text for explanation).

(R1) The runs have at most two directions, differing by 90^0,[3] and for one of these directions, the run length must be 1.
(R2) The runs can have only two lengths, which are consecutive integers.
(R3) One of the run lengths can occur only once at a time.
(R4) For the run length that occurs in runs, these runs can themselves have only two lengths, which are consecutive integers; and so on.

Few examples of cellular curves/envelopes are shown in Fig. 3 to explain the significance of properties (R1)–(R4). For the curve in (a), if we consider the center of each cell as a grid point, as mentioned earlier, then its chain code is $\mathsf{000200020002000} = \mathsf{0^3 2 0^3 2 0^3 2 0^3}$, which consists of codes $\mathsf{0}$ and $\mathsf{2}$ only, and contains consecutive $\mathsf{0}$'s but no two consecutive $\mathsf{2}$'s, thereby satisfying property (R1). Regarding (R2), (R3), and (R4), since there is only one run length (of $\mathsf{0}$'s), this curve trivially satisfies these three properties, and becomes a CSS. Similarly, since the curve in (b) has chain code $\mathsf{0^3 2 0^3 2 0^3 2 0^2}$, which obeys (R1)–(R4), it is a CSS. On the contrary, the curve in (c) has chain code $\mathsf{0^3 2 0^3 2 0^5 2 0^1}$, which satisfies (R1), but violates (R2) as $\mathsf{0}$ has non-consecutive run lengths (3 and 5) — even if we do not consider the leftmost and the rightmost run lengths (which are 3 and 1, respectively), and so it is not a CSS.

In our method for extraction of CSS from the cellular envelope $\mathcal{E}(\mathcal{C}, \mathcal{G})$, we have adhered to the properties (R1–R4). In addition, we have considered that also the leftmost and the rightmost run lengths of a CSS should follow property (R2) (which is not mandatory as suggested in [24]).

4 Proposed Method (PACE)

The method on finding the (cellular) polygonal approximation of an object $\mathcal{C}$ consists of stage I and stage II. In stage I, we construct the cellular envelope $\mathcal{E}(\mathcal{C}, \mathcal{G})$ based on the combinatorial arrangement of the cells containing $\mathcal{C}$ (Sec. 2).

[3] In our work, we have considered 4-connectivity of a DSS, i.e., having chain codes lying in the set $\{\mathsf{0}, \mathsf{2}, \mathsf{4}, \mathsf{6}\}$, since the cells in the cellular envelope $\mathcal{E}(\mathcal{C}, \mathcal{G})$ obtained for the curve $\mathcal{C}$ (Sec. 2) are connected in 4-neighborhood. In a DSS with 8-connectivity, however, the runs would have directions differing by 45^0 as stated in [24].

STEP 1. Initialize each entry in A_e and each entry in A_c with '0'.

STEP 2. DFS-VISIT on $\mathcal{C}$ starting from p using 8-connectivity to reach the nearest cell edge e_p of $\mathcal{G}$.

STEP 3. DFS-VISIT on A_e starting from the entry $A_e(e_p)$ corresponding to e_p in A_e using 4-connectivity (of '1's in A_e) to assign:
'1' to the entry in A_e corresponding to each cell edge e intersected by $\mathcal{C}$, and
'1' to the entry in A_c corresponding to each of the two cells with e as the common edge.

STEP 4. DFS-VISIT on A_c starting from some cell (e.g., c_p, the left adjacent cell of e_p) of the cellular envelope formed by the '1's obtained in step 3 using 4-connectivity (of '1's in A_c); and check whether the entry $A_c(c)$ corresponding to the cell c currently under DFS-VISIT satisfies at least one of the following two conditions:
(i) both the left and the right adjacent entries of $A_c(c)$ are '1's;
(ii) both the bottom and the top adjacent entries of $A_c(c)$ are '1's.
If (i) or/and (ii) is/are true, then terminate the DFS-VISIT, since the current cell c lies either on a horizontal edge/part (when (i) satisfies) or on a vertical edge/part (when (ii) satisfies) of the cellular envelope of $\mathcal{C}$; and declare c as the seed cell c_0 for stage II.

STEP 5. If no seed cell c_0 is found in step 4, then the cell size is not sufficiently large compared to the (minimum) thickness of the input curve $\mathcal{C}$. Hence the user may be asked to increase the cell size (i.e., grid separation g); alternatively, an arbitrary cell of the envelope may be considered to be the seed cell c_0.

Fig. 4. Algorithm FIND-CELLULAR-ENVELOPE($\mathcal{C}, \mathcal{G}, p$) in stage I

In stage II, we analyze the cells of $\mathcal{E}(\mathcal{C}, \mathcal{G})$ to extract the straight pieces from $\mathcal{E}(\mathcal{C}, \mathcal{G})$, considering the center of each cell of $\mathcal{E}(\mathcal{C}, \mathcal{G})$ as a grid point and using the straightness properties (Sec. 3).

4.1 Stage I: Finding the Cellular Envelope

We consider any point $p \in \mathcal{C}$ as the start point defining the object $\mathcal{C}$. For the time being, consider that $\mathcal{C}$ is connected in 8-neighborhood. Then using DFS-VISIT (Depth First Search algorithm [26]), we can reach the nearest edge e_p of a cell that intersects $\mathcal{C}$. Starting from e_p, using DFS-VISIT on the edges of the cells, we visit those cell edges that are intersected by $\mathcal{E}$; this procedure helps us in constructing the edge matrix A_e and the cell matrix A_c (Sec. 2), which are finally used to obtain $\mathcal{E}(\mathcal{C}, \mathcal{G})$. The major steps of the algorithm FIND-CELLULAR-ENVELOPE($\mathcal{C}, \mathcal{G}, p$) to find the cellular envelope of a connected (and of uniform or non-uniform thickness) object $\mathcal{C}$ w.r.t. the cellular array imposed by the grid $\mathcal{G}$ is given in Fig. 4.

In case $\mathcal{C}$ has some missing points/pixels, i.e., possesses disconnectedness, then it may happen that none of the edges of a cell is intersected by $\mathcal{C}$, although $\mathcal{C}$ is

```
STEP 1. Traverse (cell-wise) towards left and towards right from $c_0$ to extract all
        possible pairs of CSS starting from $c_0$, such that
        (i) the chain code of each CSS, and
        (ii) the combined chain code of the two CSS's
        in each pair are in conformity with properties (R1)–(R4);
STEP 2. Find a/the pair of CSS that has maximum sum of lengths;
        merge this pair into a single CSS, namely $C_1$;
        declare $c_0$ and $c_1$ as the left and the right terminal cells of $C_1$;
        store (the centers of) $c_0$ and $c_1$ in the ordered set $T$.
STEP 3. Start from $c_1$ to extract the next (longest) CSS, $C_2 := (c_1, c_2)$, with
        terminal cells $c_1$ and $c_2$;
        store $c_2$ in T; and mark the cells defining $C_2$ as VISITED.
STEP 4. Repeat STEP 3 starting from the last entry (i.e., terminal cell) in $T$ to get
        the CSS's defining $\mathcal{E}$ until all cells of $\mathcal{E}$ are VISITED (using DFS-VISIT).
        Note: (i) If a CSS has both its terminal cells in the 4-neighborhood of
        another (longer) CSS, then the former (shorter) CSS is not included in
        $T$ (Fig. 6(a)). (ii) For a bifurcating/branching CSS, we store both its
        terminal cells in $T$ (Fig. 6(b)).
STEP 5. Declare $T$ as the polygonal approximation of the cellular envelope $\mathcal{E}$.
```

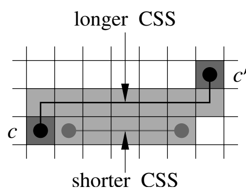
Fig. 5. Algorithm FIND-CSS($\mathcal{E}, c_0$) in stage II

contained in that cell. To circumvent this problem, we have to directly construct the cell matrix A_c, without constructing A_e, which would, however, increase the time complexity (and the run time, thereof) of stage I. Further, if the curve possesses too much gap/disconnectedness, so that the gap is even larger than the cell size, then this may result to gap (in the edge-connectivity) of the cells constituting the envelope $\mathcal{E}(\mathcal{C}, \mathcal{G})$, which gets fragmented into two or more pieces, thereby producing faulty results. Choosing an appropriate cell size is, therefore, necessary to obtain the desired cellular envelope of a disconnected DC in stage I.

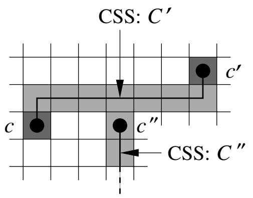
4.2 Stage II: Finding the Cellular Straight Segments

In stage II, the algorithm FIND-CSS($\mathcal{E}, c_0$)[4], given in Fig. 5, extracts the ordered set of CSS's from the cellular envelope $\mathcal{E}$ as follows. W.l.o.g., since in stage I, the seed cell c_0 lies on a horizontal part (or on a vertical part, or on a thick part) of $\mathcal{E}$, we negotiate two traversals (STEP 1) — one towards left and the other towards right of c_0 — to obtain two CSSs with complying straightness such that the sum of their lengths is maximal, and merge these two to get the first CSS, C_1, to be included in the ordered set T of terminal cells (STEP 2). The starting cell for extracting the next CSS (STEP 3) from the cellular envelope is, therefore, considered to be the right terminal cell c_1 of C_1. We use the algorithm DFS-VISIT [26] to explore the cells constituting the envelope and to extract the CSSs, whose terminal cells are finally reported in T.

[4] Now onwards, we denote the cellular envelope of $\mathcal{C}$ by $\mathcal{E}$ for simplicity.



(a) A short CSS with each of its terminal cells lying at 4-N of a longer CSS is not considered as a valid CSS (*Note* (i) of STEP 4 in Fig. 5).



(b) For a branching CSS, C'', each of its terminal cells (one is c'' and the other not shown) is stored in T (*Note* (ii) of STEP 4 in Fig. 5).

Fig. 6. Inclusion and exclusion of terminal cell(s) of CSS in T

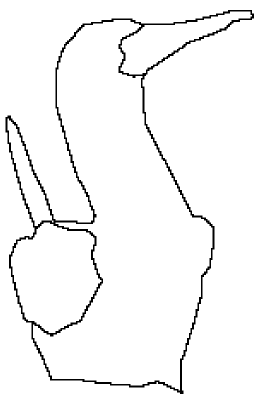
Time complexity. If N be the number of points defining the curve $\mathcal{C}$, then its envelope $\mathcal{E}$ consists of $O(N/g)$ cells. Due to DFS-VISITs, therefore, the time complexity in stage I is bounded by $O(N/g)$. In stage II, extraction of each CSS C_i takes $O(|C_i|)$ time, where $|C_i|$ is the number of cells defining C_i. Hence, the time complexity to extract all CSS's in step II is $O\left(\sum |C_i|\right) = O(N/g)$, which gives the total time complexity of **PACE** as $O(N/g)$.

5 Experiments and Results

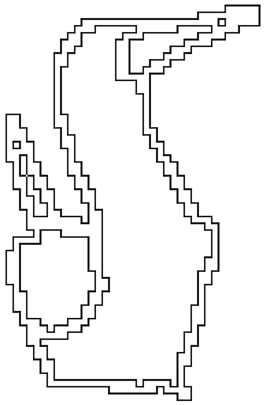
We have implemented the two algorithms, namely FIND-CELLULAR-ENVELOPE and FIND-CSS, that make the proposed method **PACE** for polygonal approximation of an arbitrarily thick DC, in C in SunOS Release 5.7 Generic of Sun_Ultra 5_10, Sparc, 233 MHz, and have tested various digital curves of arbitrary shape, changing thickness, and irregular connectedness. It may be mentioned here that, since the concept of a cellular polygon introduced in this work is entirely new, and no other work on cellular polygon exists at present, we could not have a comparative study of our method in this paper.

The result for an (non-thinned) edge map of a "duck" is shown in Fig. 7, which testifies the elegance of **PACE** in deriving the cellular polygon corresponding to a DC. It may may be noticed in this figure that, some of the cells in the envelope $\mathcal{E}$ have not been included in any CSS; because in the algorithm FIND-CSS, we have considered the (terminal cells of) each locally longest CSS to be included in P (see the *Note* in STEP 4). But when there is a bifurcation/self-intersection (e.g., in and around the root of its tail) or a sharp bend (e.g., at the tip of its beak), the cellular envelope (Fig. 7(b)) contains several cells across its thickness, which may cause error in the polygonal approximation as manifested in Fig. 7(d) in the part of the polygon corresponding to the region in and around the tail root. Hence a proper value of the cell size, g, is mandatory to ensure a good cellular envelope corresponding to a DC, and a good polygonal approximation thereof.

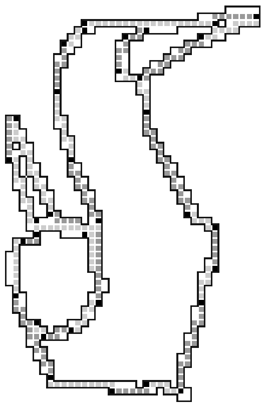
The major strength of the proposed method is the inherent nature of Euclidean-free metrics and operations involved in both the stages. This imparts high



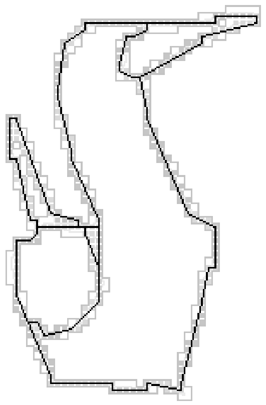
(a) Since the curve is not one-pixel thick, the conventional algorithms on polygonal approximation cannot be applied on it.



(b) Cellular envelope $\mathcal{E}(\mathcal{C},\mathcal{G})$ obtained in stage I of the algorithm **PACE**. The cells of the envelope are connected in 4-neighborhood, which are, therefore, 4-cells.



(c) The set of CSS's extracted in stage II of the algorithm **PACE** from the envelope $\mathcal{E}(\mathcal{C},\mathcal{G})$ shown in (b). The CSS's have been colored gray with the terminal cell of each CSS shown in black.



(d) Final polygonal approximation (in thin black lines) superimposed on the (faded) cellular envelope $\mathcal{E}(\mathcal{C},\mathcal{G})$.

Fig. 7. Results of algorithm **PACE** for cell size $g = 4$ on a curve-shaped digital object $\mathcal{C}$ of nonuniform thickness representing the edge map of a "duck"

execution speed to the implementation of **PACE**, which is reflected in the respective CPU times presented in Table 1. Further, with increase in the cell size g, the compression ratio (CR) improves consistently, but the quality of approximation deteriorates, as evidenced by the average errors (measured w.r.t. both

Table 1. Results of **PACE** on the "duck" image (1748 pixels), shown in Fig. 7, for different grid sizes (g)

				avg. error		CPU time (secs.)		
g	$\lvert\mathcal{E}\rvert$	$\lvert P\rvert$	CR	$d_{\perp}^{(\mathcal{E})}$	$d_{\perp}^{(P)}$	$\mathcal{E}$	P	total
2	703	130	0.074	0.87	0.92	0.026	0.227	0.253
3	445	93	0.053	1.25	1.34	0.019	0.142	0.161
4	382	53	0.030	1.49	1.97	0.014	0.129	0.143
8	191	22	0.013	2.85	3.36	0.006	0.108	0.114
12	125	18	0.010	4.03	5.58	0.004	0.071	0.075

$\lvert\mathcal{E}\rvert$ = number of cells in $\mathcal{E}$; $\lvert P\rvert$ = number of terminal cells in P; CR (= $\lvert P\rvert/\lvert\mathcal{C}\rvert$) = compression ratio; $d_{\perp}^{(\mathcal{E})}$ = isothetic error averaged over (centers of) all cells of $\mathcal{E}$ from (their corresponding nearest points of) $\mathcal{C}$; $d_{\perp}^{(P)}$ = isothetic error averaged over all terminal cells in P from $\mathcal{C}$, where $\max\{|x_1 - x_2|, |x_2, y_2|\}$ is the isothetic distance between two points (x_1, y_1) and (x_2, y_2).

$\mathcal{E}$ and P) of the curve $\mathcal{C}$ in this table. This again indicates that the cell size g should be suitably chosen to get an acceptable tradeoff in the approximation.

6 Conclusion and Future Work

We have presented here the novel concept of approximating a curve-shaped digital object by a cellular polygon. The algorithm is marked by its **(i)** indifference to change in thickness of the input DC, **(ii)** innovative combinatorial approach to construct the optimum cellular envelope for the given DC, **(iii)** use of straightness properties inherited from digital geometry, **(iv)** independency to Euclidean paradigm, and **(v)** realization without any floating point operation, which collectively make it robust, speedy, and efficient. Presently, we are experimenting on the nature of variation of the cellular envelope and the resulting polygon of a DC with its registration (both translation and rotation) w.r.t. grid, which will be reported shortly.

References

1. Klette, R., Rosenfeld, A.: Digital Geometry: Geometric Methods for Digital Image Analysis. Morgan Kaufmann (2004)
2. Klette, R., Rosenfeld, A.: Digital straightness: A review. Discrete Applied Mathematics **139** (2004) 197–230
3. Aken, J.R.V., Novak, M.: Curve-drawing algorithms for raster display. ACM Trans. Graphics **4** (1985) 147–169
4. Attneave, F.: Some informational aspects of visual perception. Psychological Review **61** (1954) 183–193
5. Imai, H., Iri, M.: Computational geometric methods for polygonal approximations of a curve. CVGIP **36** (1986) 31–41

6. Perez, J.C., Vidal, E.: Optimum polygonal approximation of digitized curves. PRL **15** (1994) 743–750
7. Schröder, K., Laurent, P.: Efficient polygon approximations for shape signatures. In: Proc. ICIP. (1999) 811–814
8. Schuster, G.M., Katsaggelos, A.K.: An optimal polygonal boundary encoding scheme in the rate distortion sense. IEEE Trans. Circuits and Systems for Video Technology **7** (1998) 13–26
9. Tanigawa, S., Katoh, N.: Polygonal curve approximation using grid points with application to a triangular mesh generation with small number of different edge lengths. In: Proc. AAIM 2006. (2006) 161–172
10. Teh, C.H., Chin, R.T.: On the detection of dominant points on digital curves. IEEE Trans. PAMI **2** (1989) 859–872
11. Yin, P.Y.: Ant colony search algorithms for optimal polygonal approximation of plane curves. Pattern Recognition **36** (2003) 1783–1797
12. Rosin, P.L.: Techniques for assessing polygonal approximation of curves. IEEE Trans. PAMI **19** (1997) 659–666
13. Yin, P.Y.: A new method for polygonal approximation using genetic algorithms. PRL **19** (1998) 1017–1026
14. Devillers, O.: Inner and outer rounding of set operations on lattice polygonal regions. In: Proc. 20th Ann. Symp. Computational Geometry (2004) 429–437
15. Cohen, J., et al.: Simplification Envelopes. In: Proc. SIGGRAPH (1996) 119–128
16. Bhattacharya, P., Rosenfeld, A.: Contour codes of isothetic polygons. CVGIP **50** (1990) 353–363
17. Bhowmick, P., Biswas, A., Bhattacharya, B.B.: Isothetic polygons of a 2D object on generalized grid. In: Proc. PReMI 2005. LNCS **3776**. 407-412
18. Biswas, A., Bhowmick, P., Bhattacharya, B.B.: **TIPS**: On finding a **T**ight **I**sothetic **P**olygonal **S**hape covering a 2d object. In: Proc. SCIA 2005. LNCS **3540**. 930–939
19. Yu, B., Lin, X., Wu, Y., Yuan, B.: Isothetic polygon representation for contours. CVGIP **56** (1992) 264–268
20. Fam, A., Sklansky, J.: Cellularly straight images and the hausdorff metric. In: Proc. Conf. on Pattern Recognition and Image Processing. (1977) 242–247
21. Geer, P., McLaughlin, H.W.: Cellular lines: An introduction. Discrete Mathematics and Theoretical Computer Science (2003) 167–178
22. Kim, C.E.: On cellular straight line segments. Computer Graphics Image Processing **18** (1982) 369–391
23. Klette, R.: Cell complexes through time. In: Proc. Vision Geometry. Volume IX of SPIE 4117. (2000) 134–145
24. Rosenfeld, A.: Digital straight line segments. IEEE Transactions on Computers **23** (1974) 1264–1268
25. Freeman, H.: On the encoding of arbitrary geometric configurations. IRE Trans. Electronic Computers **EC-10** (1961) 260–268
26. Cormen, T.H., Leiserson, C.E., Rivest, R.L.: Introduction to Algorithms. Prentice Hall of India Pvt. Ltd. (2000)