Polygonal Approximation Using a Multiresolution Method and a Context-free Grammar

Hermilo Sánchez-Cruz[1](✉), Osvaldo A. Tapia-Dueñas[1], and Francisco Cuevas[2]

[1] Centro de Ciencias Básicas, Universidad Autónoma de Aguascalientes,
Av. Universidad 940, 20131 Aguascalientes, Aguascalientes, Mexico
hsanchez@correo.uaa.mx, black.osvo@gmail.com

[2] Departamento de Metrología Óptica, Centro de Investigaciones en Óptica, A. C.,
37150 León, Guanajuato, Mexico
fjcuevas@cio.mx

Abstract. An new method to obtain polygonal approximation for object contours is presented. The method consists of coding the contour with the Angle Freeman chain code (AF8), obtaining strings of eight symbols, and looking for patterns of substrings that represent slope changes along the contour. Our strategy for detecting dominant points is to look for the ends of discrete straight lines through patterns of AF8 symbols, which can be produced by a context-free grammar. With a multiresolution method, we present the polygonal approximation for noisy contours. A set of N dominant points is obtained, the integral square error (ISE) is calculated and, finally, based on lost pixels (LP) in decoding process a new error criterion, that we call lost ratio (LR), is proposed. We found that our method produces the lowest ISE, LP and LR regarding the state-of-the-art.

Keywords: Dominant points · Polygonal approximation · Angle Freeman chain code · Contour shapes · Context-free grammar

1 Introduction

The search for optimal methods to find descriptors is a constant task in computer vision and pattern recognition. Particularly regarding the search of methods with polygonal approximation to represent the shape of binary objects, in which the data of the vertices, significantly reduce the memory storage and facilitate the handling of the original shape information. In this work, each vertex is also called dominant point (DP, for short). Of course, it is inevitable to lose data, but this loss is bearable as long as the information of both the main shape features and its original topology are not affected in the least.

Attneave [2] already had noticed that a shape could be recognized when its contour was simplified by means of a set of straight lines. However, many

J. A. Carrasco-Ochoa et al. (Eds.): MCPR 2019, LNCS 11524, pp. 261–270, 2019.
https://doi.org/10.1007/978-3-030-21077-9_24

papers have been written to quantify the error between such straight lines and the original contour.

Given a contour of n cells 8-connected and listed in the clockwise direction, $C = \{(x_i, y_i), i = 1, ..., n\}$, the problem of finding the best polygon of m vertices (*i.e.*, one that allows a tolerable error) is considering the $C_m^n = n!/[m!(n-m)!]$ different possible polygons.

Teh and Chin [13] use Freeman chain code of eight directions [5], that we call F8, to represent contour shapes and proposed a non-parametric method for dominant point detection. In [12], Sarkar used F8 chain code seeking significant vertices by differentiating the code symbols, instead of taking any coordinate into account. Cronin [4] developed a symbolic algorithm using, also, F8 chain code to assign special symbols to detect DPs. Arrebola and Sandoval [1] proposed a hierarchical computation of a multiresolution structure on chain-coded contours allowing detection of shape details at different scales.

Some authors base their method on the iterative elimination of candidates, called *break points*, until final DPs are obtained [3,9,10].

An alternative way to obtain polygonal approximations is presented in this work, which is based on the recognition of chains that are part of a context-free grammar. Also, our method relies on not only to significantly decrease the number of vertices, *i.e.*, the dominant points, but to look for an error criterion that not only implies the integral square error or the compression ratio, but also, on the amount of information that is lost from the original contour, since there are pixels that can not be recovered in a decoding process.

This paper is organized as follows. In Sect. 2 we explain our proposed method by using a context-free garmmar and a multiresolution method to find DPs, whereas in Sect. 3 the proposal for a new error criterion is detailed. The application of our method is presented in Sect. 4. Finally, in Sect. 5 we give some conclusions and further work.

2 Method

Following definitions are used throughout the paper to understand our method.

Definition 1. A 2D *grid*, is a regular orthogonal array, denoted by $\mathbb{G}$, composed of r rows and c columns of resolution cells. A resolution cell is called a *pixel* denoted by p, if the following two properties are considered: its *Cartesian coordinates* (x, y) and its *intensity value* $I_p \in \{0, 1\}$. If $I_p = 0$, we say that the resolution cell is a 0-*pixel*; on the contrary, if $I_p = 1$, the resolution cell is a 1-*pixel*. Unless otherwise stated, and without causing confusion, in this work we often consider 1-pixel simply as pixel.

Definition 2. AF8 is a two-based vector code [6]: a reference and change vector, whose direction changes are labeled by the symbols of the alphabet $\Sigma_{AF8} = \{a, b, c, d, e, f, g, h\}$. See Fig. 1.

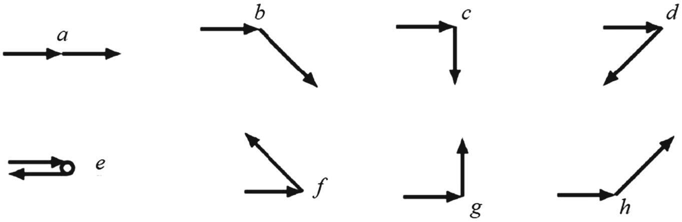


Fig. 1. *AF*8 symbols

We rely on the search for breakpoints where some others can be added carrying out another iteration, such that error criteria are the minimum as possible.

The steps of our multiresolution method are:

1. Consider two superimposed grids, $\mathbb{G}$ and $\mathbb{G}'$, so that $\mathbb{G}'$ can be scaled by a parameter $\alpha \geq 1$, with $\alpha \in \mathbb{Z}$. This scaling is done through the origin given by the centroid of the binary object. Start with α as a power of 2.
2. Obtain another contour in $\mathbb{G}'$ with the help of the original in $\mathbb{G}$ by visiting each cell of both grids, and follow the next steps.
   (a) From leftmost and upmost, find the first cell of $\mathbb{G}'$ that contains 1-pixels of the contour in $\mathbb{G}$ and mark it. Cover the set of marked cells in clockwise.
   (b) The next cell to mark in the 8-vicinity of $\mathbb{G}'$ is the one that has the greatest number of 1-pixels of $\mathbb{G}$.
   (c) Repeat last step until all 1-pixels of $\mathbb{G}$ have been covered.
   (d) Given the AF8 chain code of the contour, find strings from the set.

$$L = \{xa^p(bha^q)^r, xa^p(hba^q)^r \ |x \in \{a, b, c, d, e, f, g, h\}\}, \qquad (1)$$

   where p, q, r indicate the number of times the symbol or substring in parentheses is concatenated, x is the label for the breakpoints and $a, b, ..., h$ are symbols of the alphabet AF8.
   (e) Once a cell of $\mathbb{G}'$ has been defined as breakpoint, find the 1-pixel of $\mathbb{G}$ closest to the center of the cell of $\mathbb{G}'$ and define it as a breakpoint.
3. Given two breakpoints (x_k, y_k) and (x_{k+1}, y_{k+1}), a continuous-line segment is defined. The distance between this segment and the points of the contour cells is given by Eq. (2).

$$d^2(p_i, \overline{p_k p_{k+1}}) = \frac{((x_i - x_k)(y_{k+1} - y_k) - (y_i - y_k)(x_{k+1} - x_k))^2}{(x_i - x_{k+1})^2 + (y_k - y_{k+1})^2}. \qquad (2)$$

   If $\alpha \geq 1$ and, also, the error between line segments given by breakpoints and the contour is greater than a certain tolerable error, make $\alpha \rightarrow \alpha/2$ and go to step 2. Otherwise, consider all breakpoints as DPs and stop.

The main idea of our method is to capture what visually seems to us a digital straight segment (DSS). Of course, there is an error if we consider the continuous straight segment. For contours with high noise, it is not convenient

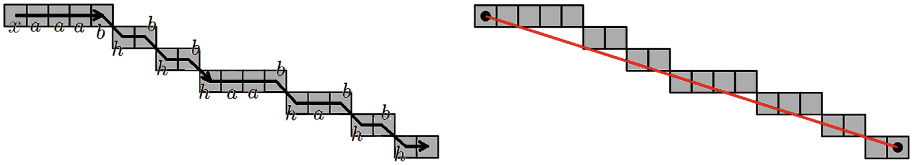
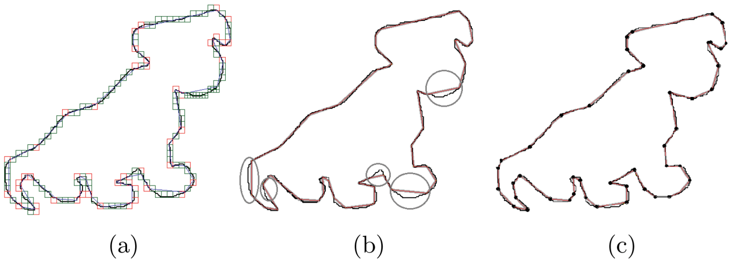


Fig. 2. Left: a discrete straight line coded with AF8 chain code. Right: a continuous straight line is adapted.

to adapt DSS for each pair of abrupt changes, that is why we proceed to expand $\mathbb{G}'$ doing $\alpha > 1$, with this we apply the algorithm on $\mathbb{G}'$ cells, ignoring the details of the noise. Figure 2 presents an example of a visual DSS. As can be observed, the AF8 chain code is $C_{AF8} = xaaabhbhbhaabhabhbh$, which can also be written as $C_{AF8} = xa^3\underbrace{bha^0}\underbrace{bha^0}\underbrace{bha^2}\underbrace{bha^1}\underbrace{bha^0}\underbrace{bha^0}$. Notice that it is on the form given by L in Eq. (1), where $p = 3$, $0 \leq q \leq 2$, and $r = 6$.

Figure 3 exemplifies our method. In Fig. 3(a) the contour is immersed in the grid $\mathbb{G}'$ scaled by $\alpha = 4$. The red cells represent the breakpoints. On the other hand, in Fig. 3(b), an approximating polygon was obtained in the first iteration, obtained by applying the CFG. The circumscribed regions are not under a tolerable error. Once our procedure has been carried out iteratively, a final set of DPs is obtained, as shown in Fig. 3(c).



(a) (b) (c)

Fig. 3. Example of the method: (a) the grid $\mathbb{G}'$, (b) first iteration, (c) final DPs.

Theorem 1. *L is a subset of a language generated by a context-free grammar,* CFG*.*

Proof. Let a 4-tuple $G = (V, \Sigma_{AF8}, S, P)$, where the variables V and terminal symbols Σ_{AF8} are disjoint sets, $S \in V$ and P is the set of productions given by the formulas below.

$$S \rightarrow \; xAB|xAC$$

$$B \rightarrow \; bhAB|\; \epsilon$$

$$C \rightarrow \; hbAC \;|\; \epsilon$$

$$A \rightarrow \; aA \mid \epsilon$$

where ϵ is the empty string. As can be noticed this 4-tuple defines a CFG and produces each of the strings given by L in Eq. (1).

3 Trade Off Between Common Error Criteria

A considerable number of papers have been written to find the best polygonal approximation, proposing a series of error criteria to evaluate the different methods. Some parameters commonly used for assessing the methods are given by the compression ratio (CR, Eq. (3)) and the integral square error (ISE, Eq. (4)).

$$\mathrm{CR} = \frac{n}{N}. \tag{3}$$

$$\mathrm{ISE} = \sum_{i=1}^{n} d_i^2. \tag{4}$$

where n is the number of pixels of the contour shape and N the number of DPs.

As noted by Masood and Haq in [10], the quality of the polygonal approximation should be measured in terms of the data reduction and in the similarity with the original contour, as well. Of course, another primary criterion is the number of DPs. However, sometimes this number is sacrified to obtain minor error distance. In this work we also propose to consider the number of pixels that are lost (LP) when a decoding is carried out to recover the shape. The reasons are given below. Once the DPs are found, if a decoding is performed, the lost pixels can be counted. The approximated polygon is obtained by considering the pixels that contain part of the continuous straight segments given by pairs of DPs. Starting with the first DP, the next pixel to decode is chosen when it contains the longest segment length. If the neighbor cell with the largest segment matches the 1-pixel of the original contour, then the pixel is not lost, otherwise it is a lost pixel.

Figure 4 shows an example of lost pixels when decoding a segment between two DPs, which make a side of a polygonal approximation. Traversing the cells from top to bottom and from left to right, note that the 1-pixels labeled from 1 to 4 contain less length, of the continuous segment, than one of the neighbors (0-pixels) of the previous visited pixel, therefore they are pixels that are lost in decoding, that are mark in yellow. The gray pixels in the right of Fig. 4 are the final decoded approximating polygon. Note, also, that there is an error between the recovered pixels and the continuous segment, given by the coordinates in black dots.

Consider the case in which N DPs are found. Suppose the shape is recovered, and the exact original contour is obtained. In this case there is no loss of information and the method can be considered *lossless*. Something important to note (as depicted to the right of Fig. 4) is that this can happen even if ISE $\neq$ 0! If, on the other hand, those N DPs are found in such places where the recovered contour loses pixels, then the method is *lossy*.

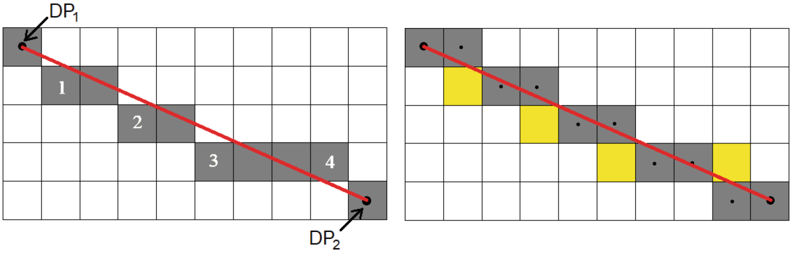


Fig. 4. Lost pixels in a decoding process. Red continous segment is a side of the approximating polygon. Left: some 1-pixels of the original contour in gray cells; right: lost pixels in yellow cells. (Color figure online)

Suppose two solution models (losy and lossles models) that give the same number of DPs, however distributed in different places. Of course, the value of CR is the same!

Once we have analyzed these ambiguities, we propose to consider the importance of N and ISE as a summation in a lost ratio (LR), but weighted by lost pixels (LP), fairly in a single equation, given by Eq. (5).

$$\mathrm{LR} = \frac{\mathrm{LP} * (N + \mathrm{ISE})}{n}, \tag{5}$$

where LP is the number of pixels lost in the decoding and n the number of pixels of the original contour. Thus, we propose to consider the number of lost pixels as part of the effectiveness of the method: the fewer pixels lose the method, the better. The same is valid for ISE and N, as expressed in Eq. (5).

4 Experiments

We applied our method to a set of samples that commonly appear in the literature. To select the values of the parameters p and q of our proposed L, each string of the AF8 chain code is read, and the maximum number of concatenated a's is obtained, while r is the result of finding repetitions of the form bha^q or hba^q.

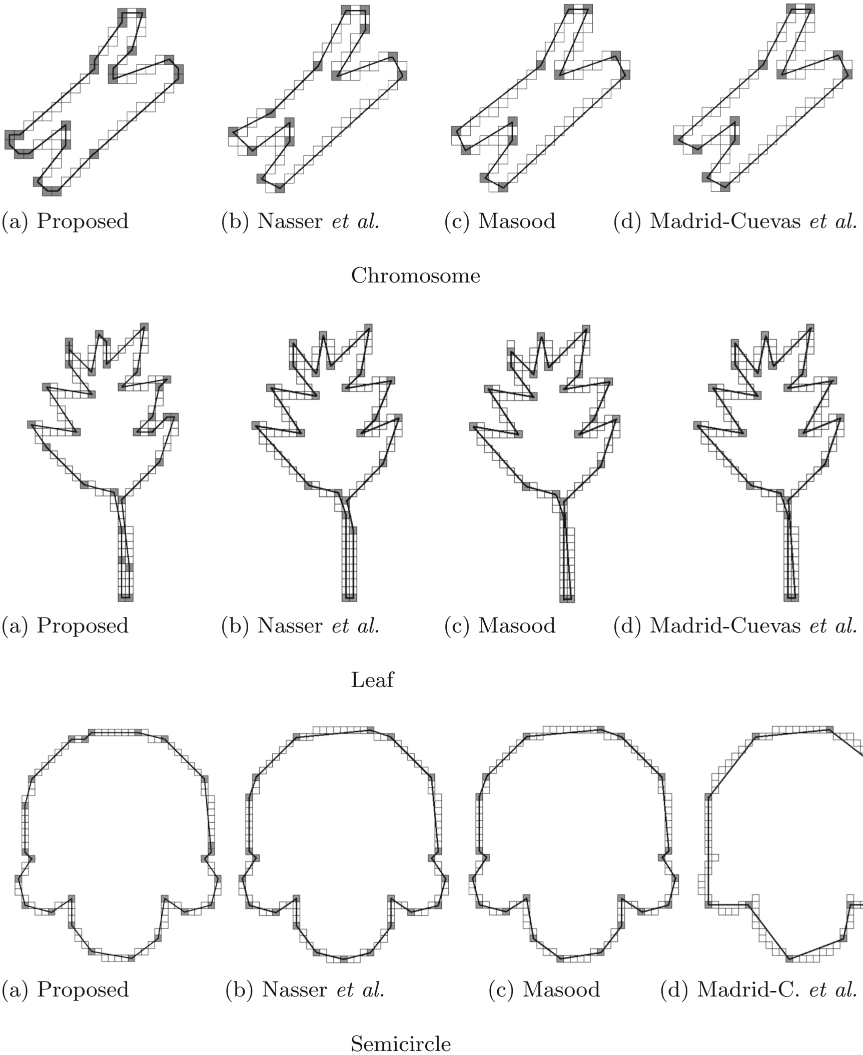
4.1 First Set

In this first part, the parameter $\alpha = 1$, *i.e.* no scale is performed due to the very low resolution of sample test. The chain codes of each sample are as presented in Table 1.

Our proposed method was compared and implemented, taking our tolerable errors from those found by Naser *et al.* [11], Masood [8] and Madrid-Cuevas *et al.* [7] methods, using parameters $(p, q, r) = (4, 4, 2)$ for Chromosoma and Leaf and (6,6,1) for Semicircle polygonal approximations, respectively.

Table 1. Chain codes of the sample shapes.

Shape	Chain code
Chromosome	cacabhbahgahbhbbbbaabhaaaaaaaabbcahbhafabhbbbchabhaaaahbahbh
Semicircle	baaaaaaabhabaaaaabahbaaaaaaabgbhbabhchbabhbfaaabhbabhbaaab hbbabhbaaafbhbabhchbabhbgbaaaaaaabhabaaaaabah
Leaf	daabhabafaahbadhabhbafahabcabhbhbaaaahaahbaaaaaaaaccaaaaaaaa aahbagabhbaaaaabhdbhaaafabhbhdbhaaafabhbaaehaafaabhdbaafaaaa


(a) Proposed (b) Nasser *et al.* (c) Masood (d) Madrid-Cuevas *et al.*

Chromosome

(a) Proposed (b) Nasser *et al.* (c) Masood (d) Madrid-Cuevas *et al.*

Leaf

(a) Proposed (b) Nasser *et al.* (c) Masood (d) Madrid-C. *et al.*

Semicircle

Fig. 5. Dominant points of three shapes.

In our experiments, we found an interesting result: the number of pixels that are lost when decoding the shape is lower with our method than with those of the literature. Table 2 shows the results of applying our method comparing with the above mentioned other recent polygonal approximation methods, whereas in Fig. 5 a visual comparison of the different methods is presented.

Table 2. Quantitative comparisons with other polygonal approximation methods.

Shape	Method	N	CR	ISE	LP	LR
Chromosome	Proposed	22	2.73	**3.78**	**8**	3.44
$n = 60$	Nasser *et al.* (2018)	14	4.28	4.97	10	**3.16**
	Masood (2008)	12	5	7.76	15	4.94
	Madrid-Cuevas (2016)	12	0.2	5.82	13	3.86
Leaf	Proposed	31	3.87	**8.42**	**11**	**3.61**
$n = 120$	Nasser	24	4.95	9.96	15	4.25
	Masood	23	5.22	10.61	18	5.04
	Madrid-Cuevas	22	5.45	11.16	18	4.97
Semicircle	Proposed	25	4.08	**6.32**	**6**	**1.84**
$n = 102$	Nasser	23	4.43	7.63	11	3.30
	Masood	22	4.64	8.61	17	5.10
	Madrid-Cuevas	10	10.2	40.79	47	23.40

4.2 Second Set

In this subsection, we show the application of our method, for objects with greater length in contour shapes. We compare our proposed method with Algorithm 1, APS (applying automatic simplification process) and FDP (fixing the desired number of dominant points) reported recently by Nasser *et al.* [11].

Using Eq. (1), parameters were found. For Shark: $(p, q, r) = (20, 20, 7)$ for Cup: $(p, q, r) = (19, 19, 1)$ and for Stingray: $(p, q, r) = (4, 4, 1)$. Table 3 shows the results in error criteria defined. Cup and Stingray are highly noisy shapes, and a multiresolution process was applied, by using the method iteratively from $\alpha = 4$, to $\alpha = 1$.

Figure 6(a) shows the regions where multiresolution was used, while Fig. 6(b) shows a comparison of our method with those of Nasser *et al.*

Table 3. Quantitative comparisons of second set with other polygonal approximation methods.

Shape	Method	N	ISE	CR	LP	LR
Shark	Proposed	37	**45.46**	7.92	**73**	**20.54**
$n = 293$	Algo 1	23	79.41	12.73	137	47.89
$\alpha = 1$	APS	21	75.52	13.95	133	43.81
	FDP	19	105.09	15.42	135	57.18
Cup	Proposed	20	**93.22**	20.25	**105**	**29.35**
$n = 405$	Algo 1	21	160.70	19.29	167	74.92
$\alpha = 4, 2$	APS	11	238.45	36.82	220	135.50
	FDP	17	159.56	23.82	180	78.47
Stingray	Proposed	37	**84.36**	8.86	**122**	**45.14**
$n = 328$	Algo 1	25	118.66	13.12	180	78.84
$\alpha = 4, 2, 1$	APS	23	121.38	14.26	160	70.43
	FDP	20	165.75	16.40	180	101.94

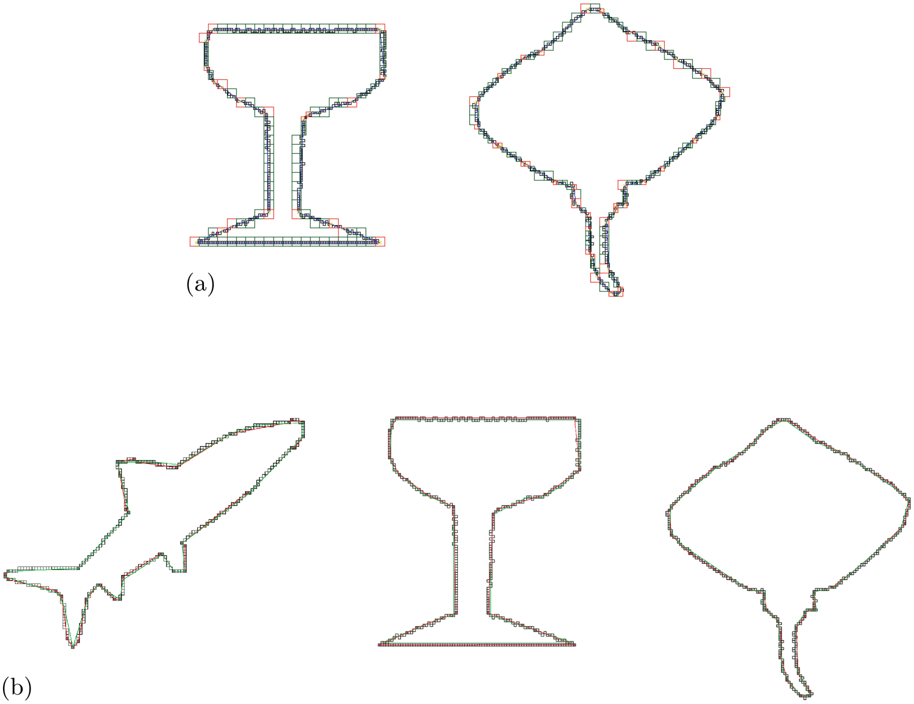


Fig. 6. (a) Mutirresolution method applied to Cup and Stringray shapes; (b) polygonal approximation in red is from our proposed method, whereas in green is given by Algorithm 1 from Nasser *et al.*. (Color figure online)

5 Conclusions and Further Work

Without any explicit analysis of curvature changes, we have proposed a new method for detecting dominant points and consequently a polygonal approximation, with an error that improves current models. Although the chain codes already implicitly contain the information of the angles and curvature changes, our method is based on the syntactic search of strings well established by a context-free grammar. In addition, a new evaluation criterion was proposed for the polygonal approach, based on lost pixels in decoding.

As a future work, it is suggested to apply our method to higher resolution shapes, and with greater amount of noise. On the other hand, we decided to find the closest pixel to the center of a $\mathbb{G}'$ cell, however it may not be the optimal. A study through metaheuristic techniques may be appropriate.

Acknowledgements. H. Sánchez-Cruz was supported by CONACyT and Universidad Autónoma de Aguascalientes, under Grant PII18-8.

References

1. Arrebola, F., Sandoval, F.: Corner detection and curve segmentation by multiresolution chain-code linking. Pattern Recogn. **38**(10), 1596–1614 (2005)
2. Attneave, F.: Some informational aspects of visual perception. Psychol. Rev. **61**(3), 183–193 (1954)
3. Carmona-Poyato, A., Madrid-Cuevas, F., Medina-Carnicer, R., Munoz-Salinas, R.: Polygonal approximation of digital planar curves through break point suppression. Pattern Recogn. **43**(1), 14–25 (2010)
4. Cronin, T.M.: A boundary concavity code to support dominant point detection. Pattern Recogn. Lett. **20**(6), 617–634 (1999)
5. Freeman, H.: On the encoding of arbitrary geometric configurations. IRE Trans. Electron. Comput. **10**(2), 260–268 (1961)
6. Liu, Y., Zalik, B.: An efficient chain code with huffman coding. Pattern Recogn. **38**(4), 553–557 (2005)
7. Madrid-Cuevas, F., Aguilera-Aguilera, E., Carmona-Poyato, A., Munoz-Salinas, R., Medina-Carnicer, R., Fernandez-Garcia, N.: An efficient unsupervised method for obtaining polygonal approximations of closed digital planar curves. J. Vis. Commu. Image Represent. **39**, 152–163 (2016)
8. Masood, A.: Dominant point detection by reverse polygonization of digital curves. Image Vis. Comput. **26**(5), 702–715 (2008)
9. Masood, A.: Optimized polygonal approximation by dominant point deletion. Pattern Recogn. **41**(1), 227–239 (2008)
10. Masood, A., Haq, S.A.: A novel approach to polygonal approximation of digital curves. J. Vis. Commun. Image Represent. **18**(3), 264–274 (2007)
11. Nasser, H., Ngo, P., Debled-Rennesson, I.: Dominant point detection based on discrete curve structure and applications. J. Comput. Syst. Sci. **95**, 177–192 (2018)
12. Sarkar, D.: A simple algorithm for detection of significant vertices for polygonal approximation of chain-coded curves. Pattern Recogn. Lett. **14**(12), 959–964 (1993)
13. Teh, C., Chin, R.T.: On the detection of dominant points on digital curves. IEEE Transactions on Pattern Analysis and Machine Intelligence **11**(8), 859–872 (1989)