A robust method for polygonal approximation by line simplification and smoothing

Pablo Barrera Sánchez, Iván Méndez Cruz, Miguel Raz Guzmán Macedo
Faculty of Sciences
Universidad Nacional Autónoma de México
Mexico City
e-mail: pablobarrera@ciencias.unam.mx, vanmc@ciencias.unam.mx, miguelraz@ciencias.unam.mx

***Abstract*—We present a new shape preserving method for simplifying polygonal curves with a high level of detail and a criteria for noise detection. The encouraging tests carried out on contours of water bodies and digital curves can be reproduced with our open source Julia package EditBoundary.jl**

***Index Terms*—line simplification, polygonal approximation, noise detection**

I. Introduction

In some problems of cartography and hydrology the contours of water bodies and coastlines are represented by polygonal curves with a high level of detail and thus require a large number of points. The visualization and processing of these contours may have an enormous computing demand on Geographic Information Systems due to such data scales; therefore shape preserving contours with fewer points are needed to deal with these issues.

Our motivation to approximate contours arises from the requirements of structured mesh generation: since meshing regions with noise or a high level of detail can be computationally expensive, the polygonal approximation of contours is indispensable for our mesh generator [30]. Moreover, some geospatial problems require mesh generation of complex geometries to carry out numerical simulations, and in that regard the faithful polygonal approximations of contours can drastically reduce the computational resources of meshing, and consequently the overall simulation without sacrificing robustness.

Collinearity criteria [21] and detection of dominant points [26], [27] were used in our mesh generation [29] to simplify contours. However, these methods may not preserve the shape of contours with the needed level of detail. In contrast, cartographers use line simplification methods like the Douglas-Peuker algorithm [10] and Visvalingam-Whyatt algorithm [32] for the interactive visualization of high resolution maps [5].

In other areas, such as computer vision and pattern recognition, digital curves with noise are extracted from images due to integer point segmentation preprocessing. Nevertheless, geospatial applications require smooth curves that preserve the initial shape. In that regard, the above line simplification methods are inadequate. Instead, the use of filters, splines [25], [28] or noise estimators [20] can generate shape preserving contours where such a digital curve can be smoothed over the same number of points, and then subsequently simplified.

In this paper we use a method for the polygonal approximation of contours with noise or that require a high level of detail. This method consists of three stages: elimination of collinear points, contour smoothing, and line simplification.

There are different algorithms in the literature for noise detection [20], curve smoothing [18] and line simplification [10], [33], so we highlight our main contributions: the triangle areas decay method is used for noise detection, and the radius method is introduced for line simplification. We also provide an implementation of our method as a free Julia software package called EditBoundary.jl.

The present paper is organized as follows:

- §II Explanation of our line simplification method.
- §III Noise detection and reduction.
- §IV Description of the main pipeline.
- §V Case study on two data sets
- §VI Conclusions and future work.

The description of the Julia package EditBoundary.jl is presented in the appendix.

II. Contour simplification

In this paper we examine contours given by vertex sequences $P = (v_1, \dots, v_n)$ joined by the line segments $v_1v_2, \dots, v_{n-1}v_n$ - these contours are known as polygonal chains. We only use anti-clockwise oriented polygonal chains without repeated vertices or crosses between the line segments, except for the case $v_n = v_1$ corresponding to simple polygons.

Our approximation of a polygonal chain P consists in finding another polygonal chain P' with fewer vertices so that it preserves the shape of P in the sense that P' still retains some of the main features of P by visual inspection. This means that P' has some dominant points of P, where the error measure is minimized or the error is bounded by a threshold.

The error measure should be a scale independent measure, where if the polygonal chains are simple polygons, we can use the relative error of the areas enclosed by P and P' as our intended metric.

When we simplify a contour, points are removed so that we get a shape preserving contour with the least number of vertices. This can be formulated as follows:

Min-# Problem [14]: Given a polygonal chain P and a threshold $\epsilon > 0$, find a polygonal chain P' from P with the

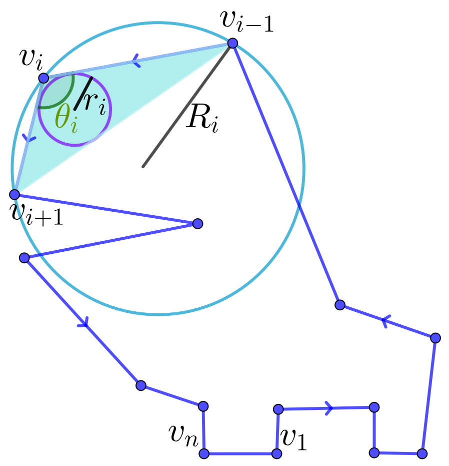


Fig. 1: Closed polygonal chain $v_1, \ldots, v_n$. We highlight the triangle $v_{i-1}v_iv_{i+1}$, its inradius r_i, circumradius R_i, and interior angle θ_i.

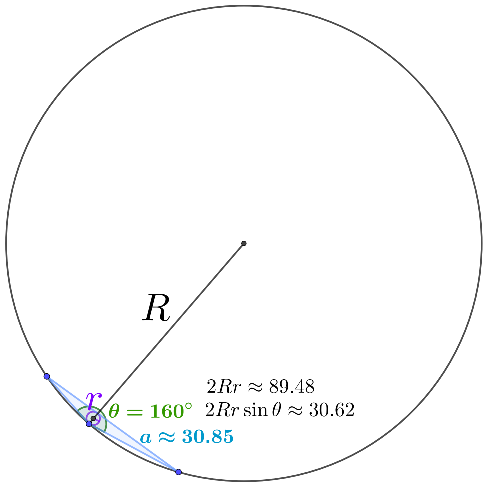


Fig. 2: In a triangle with obtuse angle θ, inradius r, and circumradius R, we compare the triangle's area a with $2Rr$ and $2Rr\sin\theta$.

least number of vertices so that that the error measure between P and P' is bounded by ϵ.

Viewed as an optimization problem, line simplification may have significant computational drawbacks [12], thus a non-starter for our interactive software given the number of points to consider.

Another idea is to examine the triangles $T_i = v_{i-1}v_iv_{j-1}$ generated by three consecutive vertices in order to leverage their geometrical properties such as area, angles, inradius, and circumradius as a modification criteria, see Fig. 1.

A. Area method

This method is a modification of the Visvalingam-Whyatt algorithm [32], [33]. The vertices are sequentially removed from the polygonal chain P based on the areas a_i of the triangles T_i. In order to get a scale independent method, the areas are divided by the average area of the triangles, denoted by $\overline{a}$.

Given a threshold $\epsilon > 0$, in each step we only remove the vertex v_j corresponding to smallest area triangle T_j if its scaled area is bounded by the threshold, that is, if

$$a_j \leq \overline{a} \cdot \epsilon. \tag{1}$$

The triangle areas can be weighted by the interior angles θ_j as described by [33] in order to get more control over the removed vertices, and in our experiments we pragmatically use $a_j \sin\theta_j$.

B. Radius method

In addition to the triangle areas, we measure the radius of the inscribed circles and circumcircles of the triangles to simplify the contours. Critically, the triangle area is bounded by the product of these radii and said bound is justified by Blundon's inequality [9], [35], namely, for every triangle with inradius r, circumradius R and perimeter p, the following inequality holds:

$$\frac{p}{2} \leq 2R + (3\sqrt{3} - 4)r, \tag{2}$$

additionally, the equality holds in (2) for equilateral triangles. Moreover, if the triangle does not have obtuse angles, then the following inequality holds:

$$2R + r \leq \frac{p}{2}. \tag{3}$$

In this case, the equality holds in (3) for right triangles.

Since the area a, the perimeter p, and the inradius r of a triangle are related by the identity

$$a = r \cdot \frac{p}{2}, \tag{4}$$

then from inequality (2) we get the following upper bound for the area:

$$a < r \cdot (2R + 1.2r), \tag{5}$$

and where, from the inequality (3), we also get the following lower bound for the area:

$$r \cdot (2R + r) \leq a. \tag{6}$$

If the triangle does not have obtuse angles and $r^2 \approx 0$, then both bounds (5)-(6) indicate that the triangle area is approximately twice the product of the inradius and the circumradius:

$$a \approx 2Rr. \tag{7}$$

We have noted that several of the triangles in our test contours have small inradius, but unfortunately, they also tend to have obtuse angles, therefore the approximation (7) may not be suitable. In order to compensate for this observation, the product of the radii is weighted by the sine of the obtuse angle as shown in Fig. 2. We then use the following formula:

$$a \approx 2Rr\sin\theta. \tag{8}$$

It is worthy to note that the double product of the inradius r and the circumradius R can be computed from the sizes ℓ_1, ℓ_2 and ℓ_3 of a triangle as follows [15]:

$$2Rr = \frac{\ell_1\ell_2\ell_3}{\ell_1 + \ell_2 + \ell_3}. \tag{9}$$

In the same manner as the area method, the polygonal chain P is sequentially simplified based on the weighted product of the radii. Let r_i be the inradius of the triangle T_i, let be R_i the circumradius of T_i, and let $\sin v_i$ be the sine of the interior angle at v_i. The weighted products of the radii $2R_i r_i \sin v_i$ are divided by their average, denoted by $\overline{\rho}$, so that we get a scale independent error measure.

Given a threshold $\epsilon > 0$, in each step we only remove the vertex v_j corresponding to the smallest weighted product of the radii if this quantity is bounded by the threshold, that is, if

$$2R_j r_j \sin v_j \leq \overline{\rho} \cdot \epsilon \tag{10}$$

Note that in each step it is enough to recompute the products of the radii for the smallest triangle and their two adjacent triangles.

In order to compare the area method against the radius method, the triangle areas are weighted by the sines of the interior angles. More points can be removed as we increase the threshold value, but the simplified contour may lose several details of the original contour.

III. Contour Smoothing

We can think of digital curves from image processing and pattern recognition applications as closed polygonal chains with integer coordinates, where some of these curves can have noise. In order to approximate these digital curves by smoothed polygonal chains, we must address noise detection and reduction.

A. Noise detection

If we know that a given polygonal chain has noise, then the contour approximation can be improved by applying a noise reduction algorithm before the contour simplification, which leaves the question of how to detect noise. To that end we propose to examine the triangles formed by three consecutive vertices and the decay distribution of said triangle areas.

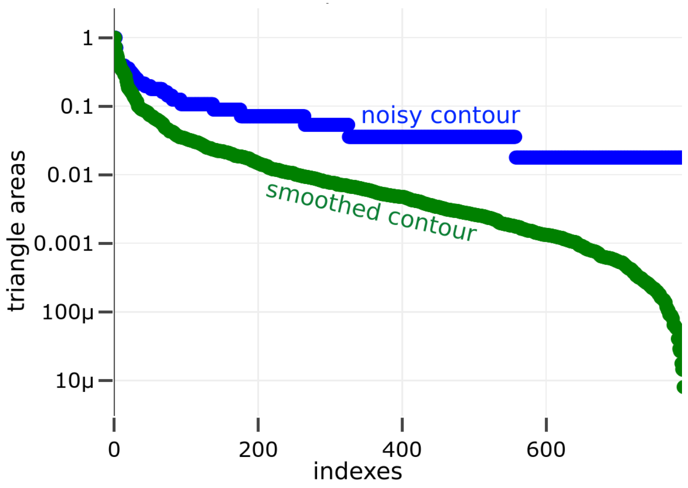
Let P be a polygonal chain of n vertices without collinear vertices, and let $\{a_i\}$ be the sequence of the corresponding triangle areas, then the areas are scaled by their largest value as follows:

$$\alpha_i = \frac{a_i}{\max_i a_i}, \quad i = 1, \dots, n \tag{11}$$

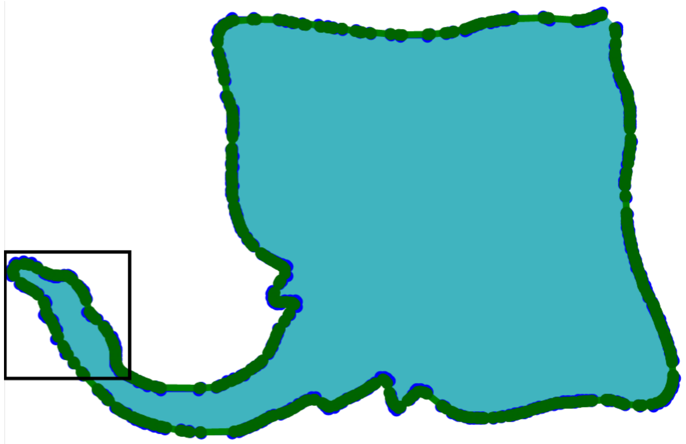
Next, these areas are sorted in decreasing order by a permutation σ of the set $\{1, \dots, n\}$. We examine the plot of the sorted triangle areas against their indexes in a logarithmic scale:

$$\left(i, \log_{10} \alpha_{\sigma(i)}\right), \quad i = 1, \dots, n. \tag{12}$$

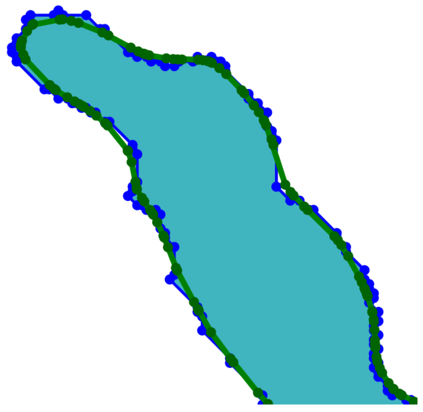
We name the plot (12) as the area plot. Our experiments suggest that the area plot for noisy polygonal chains has a staircase shape with some considerable gaps instead of the gradual decay we expect from polygonal chains without noise as shown in Fig. 3(a).



(a) Area plots for both the smoothed contour and the noisy contour of *ray10*.



(b) Comparison of the blue colored noisy contour against the green colored smoothed contour generated by (15) using the same number of points.



(c) Zoom at the tail of *ray10*: the blue colored contour has noise, while its approximation is the green colored contour.

Fig. 3: Noise detection and reduction for the digital curve *ray10*.

B. Noise reduction

Once a contour with noise is detected by visually inspecting the area plot, we apply noise reduction. In that regard, both the area and radius methods may be unsuitable for this task since the simplified contour may have unwanted noise or may not preserve the initial shape.

In order to generate shape preserving contours with the least possible noise, we use the idea of [18], which consists of moving the vertices so that the sharp borders are "straightened". More precisely, it generates the minimum length contour with the same number of points around the initial contour.

Given a polygonal chain $P = (v_1, \dots, v_n)$ of length ℓ, and a threshold $\delta > 0$, we denote by $N(P, \delta)$ the set of polygonal chains $P' = (u_1, \dots, u_n)$ such that:

$$\|v_i - u_i\|_\infty \leq \delta \cdot \ell, \quad i = 1, \dots, n, \tag{13}$$

where

$$\|(x, y)\|_\infty = \max(|x|, |y|). \tag{14}$$

The noise reduction problem is formulated in [27] as finding the minimum length polygonal chain in $N(P, \delta)$:

$$P_{\min} = \arg\min\{\text{length}(P') : P' \in N(P, \delta)\}. \tag{15}$$

The smoothing problem (15) is an non-linear optimization problem with box constraints, where the unknown variables are the coordinates of the vertices. It has a unique solution if the balls centered at v_i with radius $\delta\ell$ are disjoint sets [27]. Optimization algorithms such LBFGS-B [6] can be used in order to solve this problem.

A band around P is given by the inequality (13), and since its width depends on the length ℓ, the smoothing method is scale independent and also depends on the threshold δ to control the smoothing level. As the value of the threshold δ increases, the noise level decreases, but, we have noted in our experiments that if $\delta > 10^{-2}$, the smoothed contour may not preserve the initial shape.

We exemplify the above noise detection and reduction by smoothing the contour *ray10* with 1618 points from [19]. First, 827 collinear points are deleted by the radius method, then we show the area plot in Fig. 3(a). Since the area plot has a staircase shape, we generate the minimum length polygonal chain (15) using $\delta = 10^{-3}$ for noise reduction. The staircase shape comes from the consecutive triangles taking collinear integer coordinates due to image segmentation preprocessing algorithms that digital imaging relies on; a large number of triangles will likely occupy a small number of distinct coordinate distance combinations, thus many will share the same area. A comparison of the contour without collinear points against the smoothed contour is shown in Figs. 3(b)-(c), both contours have 791 points.

IV. Main Pipeline

Based on the previous discussion, we describe our method for approximation of polygonal chains:

1) Remove collinear points using the radius method (10)
2) Use the area plot (12) for noise detection.
3) If the polygonal chain has noise, smooth it by generating the minimum length polygonal chain (15).
4) Simplify the contour using the radius method (10).

We recommend choosing the smoothing threshold δ so that the area plot (12) decays gradually and selecting the simplification threshold ϵ so that the area score (16) is smaller than 1. In that regard, we have found experimentally suitable value ranges for the thresholds: $10^{-5} \leq \delta \leq 10^{-3}$ and $0.1 \leq \epsilon \leq 100$. Moreover, the radius method with threshold $\epsilon = 0.1$ can remove collinear points and preserve the initial shape.

Since the double product of the inradius and the circumradius approximates the area of a triangle, both the radius and area methods can use the same value ranges for the threshold to get approximately the same level of simplification. So, the radius method can be replaced by the area method in the above procedure.

In order to know that we have a shape preserving approximation, the area of the simplified polygon P' is compared with respect to the area of the initial polygon P using the following area score:

$$\alpha(P) = 10^3 \times \frac{|\text{area}(P) - \text{area}(P')|}{\text{area}(P)}. \tag{16}$$

In our experiments we have considered at least three significant digits for the accuracy of the area score, so we scale the relative error in the areas by the factor 10^3.

As we increase the threshold of the line simplification, the area score also increases, albeit this score is not a monotonous function. Plots of the area score in terms of this threshold are shown in Figs. 4 and 8.

In our examples we have observed that if the simplification threshold is smaller than one, then the area score can also be smaller than one. Moreover, our simplified contours still preserve the initial shape if their area scores are smaller than one. With this in mind, we can identify shape preserving contours by their area scores. Contours with noise and digital curves are first smoothed before the line simplification procedure, so we measure the area score of the simplified contour with respect to the smoothed contour.

V. Experiments

We consider two types of polygonal contours to showcase our method, namely the Mexican water bodies data base of AmeriGeo [1] and the digital curves of MPEG7CS [19] and Polyseg [22]. These contours are approximated depending of the noise level and the level of detail.

A. Mexican water bodies contours

The Mexican water bodies of the data base [1] do not have noise but some of them have a high level of detail, such as the case of the *La Amistad* water dam, located in the Mexican state of Coahuila. This contour has 10445 points, see Fig. 5. First, the contour is simplified using the radius method with $\epsilon = 10$. In Fig. 6, we compare the simplified contour with respect to the initial shape. The simplified contour preserves the shape

TABLE I: Line simplification of the dam *La Amistad* (10445 points)

Radius Method		
Threshold	Simplification	Area Score
0.1	61 %	0.072
1	83 %	0.687
10	93 %	0.822
Area Method		
Threshold	Simplification	Area Score
0.1	73 %	0.334
1	86 %	2.191
10	93 %	5.809

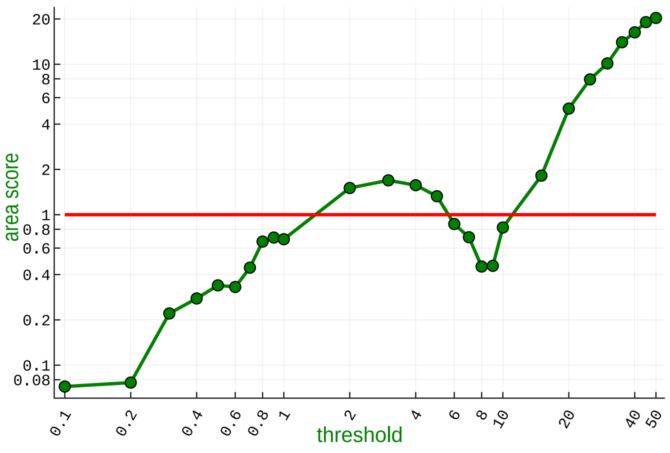


Fig. 4: Log-scale plot of the area score for the *La Amistad* water dam using the Radius Method.

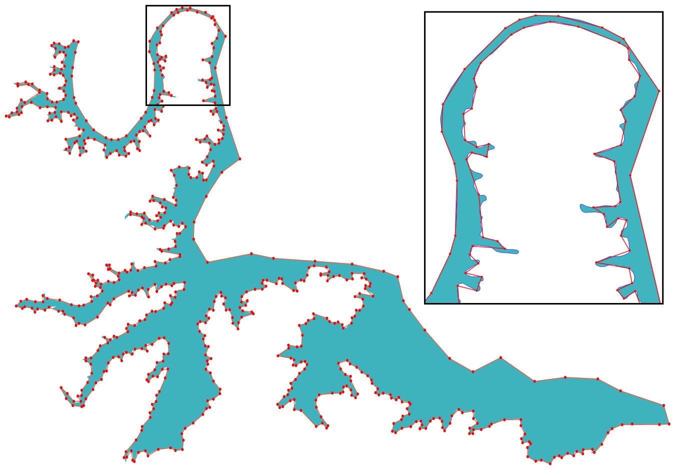


Fig. 5: Line simplification of 93% for the *La Amistad* water dam using the radius method with threshold $\epsilon = 10$. The simplified contour is red colored.

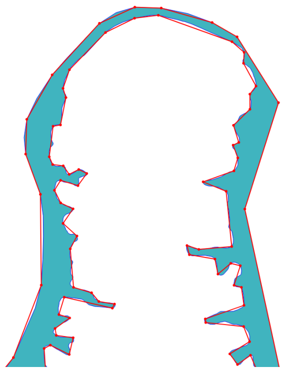


Fig. 6: Zoom of the *La Amistad* water dam. The red contour is the line simplification of 93% using the area method with threshold $\epsilon = 10$.

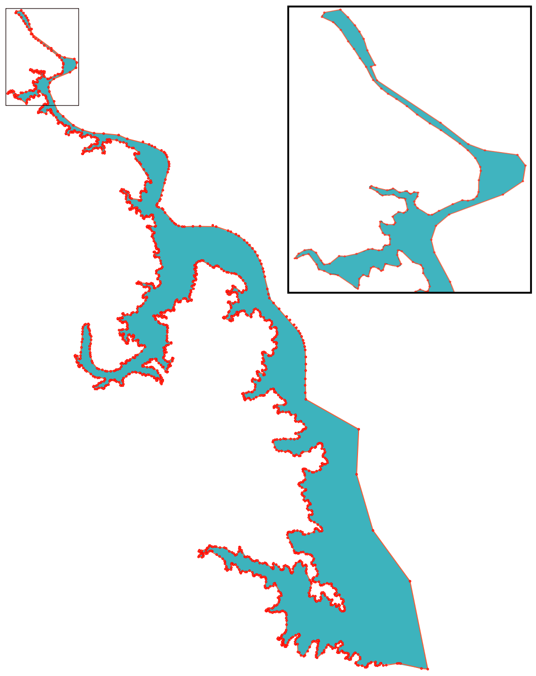


Fig. 7: Line simplification of 87% for the *Falcon* water dam using the radius method with threshold $\epsilon = 1$. The simplified contour is colored red.

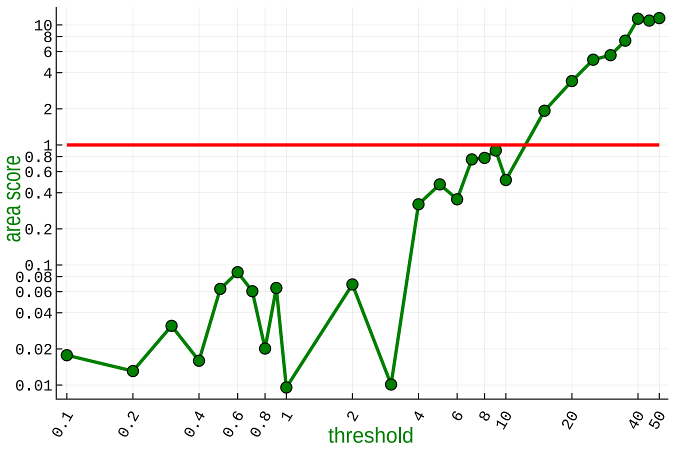
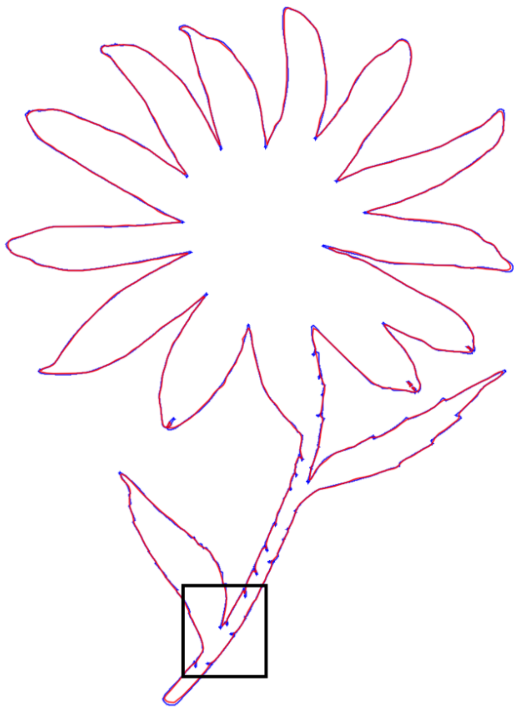


Fig. 8: Log-scale plot of the area score for the *Falcon* water dam.

TABLE II: Line simplification of the dam *Falcon* (12565 points)

Radius Method		
Threshold	Simplification	Area Score
0.5	83 %	0.063
5	93 %	0.469
50	98 %	11.390
Area Method		
Threshold	Simplification	Area Score
0.5	86 %	0.241
5	93 %	0.707
50	98 %	4.619


(a) The initial blue colored noisy contour compared against our red colored approximation.

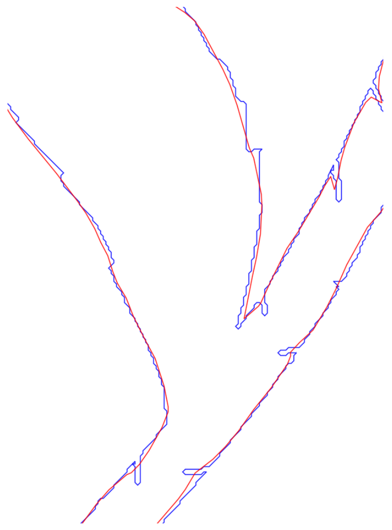

(b) Zoom at *flower* showing both the blue colored noisy contour and its red colored approximation.

Fig. 9: Approximation of the noisy contour *flower* using our method.

visually and its area score is 0.822. The plot of its area score in terms of the threshold ϵ is shown in Fig. 4. Note that the area score is larger than one for threshold values larger than one. In addition, we also simplify the contour using the area method with the same threshold value and the simplified contour is depicted in Fig. 6. Additionally, in Table I we compare our line simplification methods by their area score and level of simplification measured as the percentage of points deleted.

Another example we find noteworthy is the line simplification of the *Falcon* water dam, located between the U.S. state of Texas and the Mexican state of Tamaulipas. In Fig. 7 we show the Mexican side of the dam and its simplified contour by the radius method with threshold $\epsilon = 1$. The simplified contour preserves the initial shape visually and its area score is 0.01. The plot of its area score in terms of the threshold ϵ is shown in Fig. 8. Note that the area score is larger than one for thresholds values larger than 10. In Table II we compare our line simplification methods by their level of simplification and area score.

B. *Image segmentation contours*

Now, we consider the contour *flower* of 5788 points from the data base Polyseg [22]. This contour has holes, noise, and collinear points. First we remove 1389 collinear points, then we smooth the contour using (15) with factor $\delta = 5 \times 10^{-4}$. Afterwards, we simplify it using the radius method with threshold $\epsilon = 10$. The smoothed and simplified contour of 1045 points is shown in Fig. 9 together with the initial shape.

Interactive and high resolution visualizations of the above examples and more can be found in our website [30].

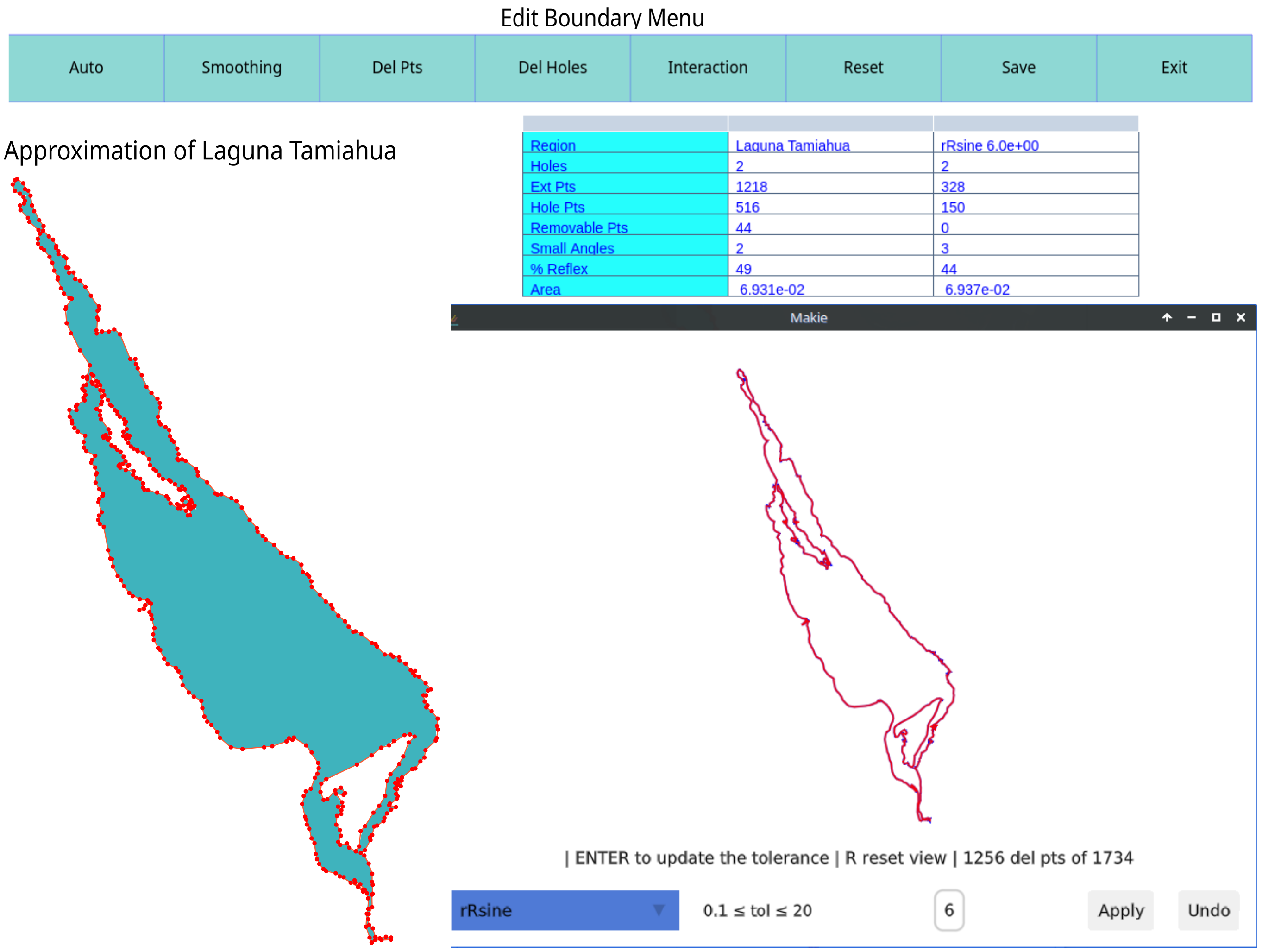
VI. Conclusions and Future Work

We have described a robust method for the approximation of polygonal contours where the area plot is used for noise detection. Our radius method simplifies the contour, and the shape preserving approximations have small area scores. Moreover, we provide EditBoundary.jl, a Julia software package, for reproducing our results. The tests conducted by this open source software bolster our claims that the proposed method can be applied to contours with different levels of noise and detail.

Our method can be extended to polygons with holes as shown in Fig. 10 by applying it to each hole separately. Later, we plan to approximate the contours of polygon decomposition using our method. A particular challenge is preserving intersecting lines in the point elimination procedure. Identifying and fixing crossings and the automatic selection of threshold values are issues we still need to incorporate into the method.

Acknowledgment

We wish to thank the reviewers for their pertinent comments which allowed us to improve this manuscript.



Region	Laguna Tamiahua	rRsine 6.0e+00
Holes	2	2
Ext Pts	1218	328
Hole Pts	516	150
Removable Pts	44	0
Small Angles	2	3
% Reflex	49	44
Area	6.931e-02	6.937e-02

| ENTER to update the tolerance | R reset view | 1256 del pts of 1734

rRsine 0.1 ≤ tol ≤ 20 6

Fig. 10: Screenshot of the main menu of `EditBundary.jl` together with a window to simplify the contour *Laguna de Tamiahua* of [1].

Appendix

EditBoundary.jl is our open source module for polygonal approximation of contours implemented in the Julia programming language [4]. It has a collection of routines for interactive line simplification, curve smoothing, and contour edition based on our methodology, see Fig. 10. This module is part of our mesh generator [30], but it can be used independently of the meshing software. The source code and test contours can be found in [30], and read-made demo version is provided at [11].

The version of [30] uses the Julia packages [8], [16] for interactive graphics and is executed inside a Jupyter notebook [13] running a Julia kernel. Users can create their polygonal contours and inspect the area plot, but only XYZ files are supported for input and output in the demo version of the code. Future versions will handle more standard geospatial formats like $.geojson$ and $.shp$.

References

[1] AmeriGeo: https://data.amerigeoss.org/dataset/mexican-water-bodies

[2] P. K. Agarwal, K. R. Varadarajan, “Efficient algorithms for approximating polygonal chains,” Discrete Comput. Geom. vol. 23, pp. 273–291, 2000.

[3] P. Barrera, J.J. Cortés, G. F. González, F. J. Domínguez, J. G. Tinoco, “Smoothness and convex area functionals revisted,” SIAM J. Sci. Comput. 2010, vol. 32, no. 4, pp. 1913–1928.

[4] J. Bezanson, A. Edelman, S. Karpinski, V. B. Shah, “Julia: a fresh approach to numerical computing,” SIAM Review 2017, vol 59, no. 1, pp. 65–98.

[5] M. Bloch, “An updated version of the original mapshaper tool v 0.2.0,” http://mapshaper.org/Visvalingam2014 (accessed 2022).

[6] R. H. Byrd, P. Lu, J. Nocedal, C. Zhu. “A limited memory algorithm for bound constrained optimization,” SIAM J. Sci. Comput. vol. 16, no. 5, pp. 1190–1208, 1995.

[7] C. Carreón, “Un módulo para el tratamiento de contornos en el sistema UNAMALLA,” Bachelor Thesis, UNAM, Mexico City, 2008.

[8] S. Danisch, J. Krumbiegel, “Makie.jl: flexible high-performance data visualization for julia,” J.O.S S., vol. 6, no. 65, 3349.

[9] G. Dospinescu, M. Lascu, C. Pohoata, M. Tetiva, “An elementary proof of Blundon’s inequality,” J. Inequal. Pure and Appl. Math. 2008, vol. 9, no. 4, 100.

[10] D. H. Douglas, T. K. Peucker, “Algorithms for the reduction of the number of points required to represent a digitized line or its caricature,” Cartographica, vol. 10, no. 2, pp. 112– 122, 1973.

[11] EditBoundary homepage: https://github.com/miguelraz/EditBoundary.jl

[12] M. Kreveld, M. Löffler, L. Wiratma. “On optimal polyline simplification using the Hausdorff and Fréchet distance,” 2018. https://doi.org/10.48550/arXiv.1803.03550

[13] Kluyver T. et al. “Jupyter notebooks: a publishing format for reproducible computational workflows,” In: Loizides F. and Scmidt B. (eds). Positioning and Power in Academic Publishing, IOS Press, pp. 87–90, 2010

[14] Kurozumi, Y. and Davis, W. A. “Polygonal approximation by the minimax method,” Comput. Graph. Image Process., vol. 19, no. 3, pp. 248–264, 1982.

[15] R. A. Johnson, “Modern geometry: an elementary treatise on the geometry of the triangle and the circle,” Houghton Mifflin Company, The Riverside Press, USA, 1929.

[16] S. Lyon, “PlotlyJS.jl: julia interface to plotly.js visualization library” https://github.com/JuliaPlots/PlotlyJS.jl (accessed 2022)

[17] A. Masood, “Dominant point detection by reverse polygonization of digital curves,” Image and Vision Computing vol. 26, no. 5, pp. 702–715, 2008.

[18] U. Montanari, “A note on minimal length polynomial approximation to a digitized contour,” Commun. ACM, vol. 13, no. 1, pp. 41–47, 1979.

[19] MPEG7 CS dataset: http://congyang.de/resources/dataset/MPEG7CS.zip

[20] Ngo, P., Debled-Rennesson, I., Kerautret, B. et al. “Analysis of Noisy Digital Contours with Adaptive Tangential Cover,” J. Math. Imaging Vis., vol 59, pp. 123–135, 2017. https://doi.org/10.1007/s10851-017-0723-7

[21] T. Pavlidis, “Curve fitting as a pattern recognition problem,”” Proc. 6th Conf. Pattern Recognition in IEEE Computer Society Press, pp. 853–859, 1982.

[22] PolySeg dataset: http://masc.cs.gmu.edu/wiki/PolySeg

[23] D. K. Prasad, M. K. H. Leung, C. Quek, S.Y. Cho, “A novel framework for making dominant point detection methods non-parametric,” Image and Vision Computing, vol 30, no. 11, pp. 843–859, 2012.

[24] M. Ramaiah, D. K. Prasad, “Polygonal approximation of digital planar curve using novel significant measure,” In: C. Volosencu, S. Kücük, J. Guerrero, O. Valero (eds), Automation and Control. IntechOpen 2020

[25] L. A. Ramírez, “Un módulo para el suavizamiento de contornos usando spline en tensión para el sistema Editboundary,” Bachelor Thesis, UNAM, Mexico City, 2011.

[26] B. K. Ray, K. S. Ray, “A non-parametric sequential method for polygonal approximation of digital curves,” Pattern Recognition Letters vol. 15, pp. 161–167, 1994.

[27] B. K. Ray, K. S. Ray, “Polygonal approximation and scale-space analysis of closed digital curves,” Apple Academic Press, Inc. 2013

[28] A. Rivera, “Un punto de vista sobre las cónicas y su uso en el suavizamiento de polígonos,” Bachelor Thesis, UNAM, Mexico City, 1999.

[29] UNAMalla 4 homepage: lya.fciencias.unam.mx/unamalla/

[30] UNAMalla 6 homepage: http://tikhonov.fciencias.unam.mx/unamalla/

[31] T. Tienaah, “Line simplification under spatial constraints”, Ph.D. dissertation, Department of Geodesy and Geomatics Engineering Technical Report No. 314, University of New Brunswick, Canada, 2018.

[32] M. Visvalingam, J. D. Whyatt, “Line generalisation by repeated elimination of points,” Cartographic Journal, vol. 30, no. 1, pp. 46–51, 1993.

[33] M. Visvalingam, J. D. Whyatt, “Implications of weighting metrics for line generalisation with Visvalingam’s algorithm,” The Cartographic Journal vol. 53, no. 3, pp. 253–267, 2014.

[34] M. Visvalingam, “The Visvalingam algorithm: metrics, measures and heuristics,” The Cartographic Journal vol. 53, no. 3, pp. 242–252, 2016.

[35] S. H. Wu, Y. M. Chu, “Geometric interpretation of Blundon’s inequality and Ciamberlini’s inequality,” J. Inequal. Appl. 2014, 381.