[Re]
Curve-fitting with piecewise parametric cubics

Daniel, Davi

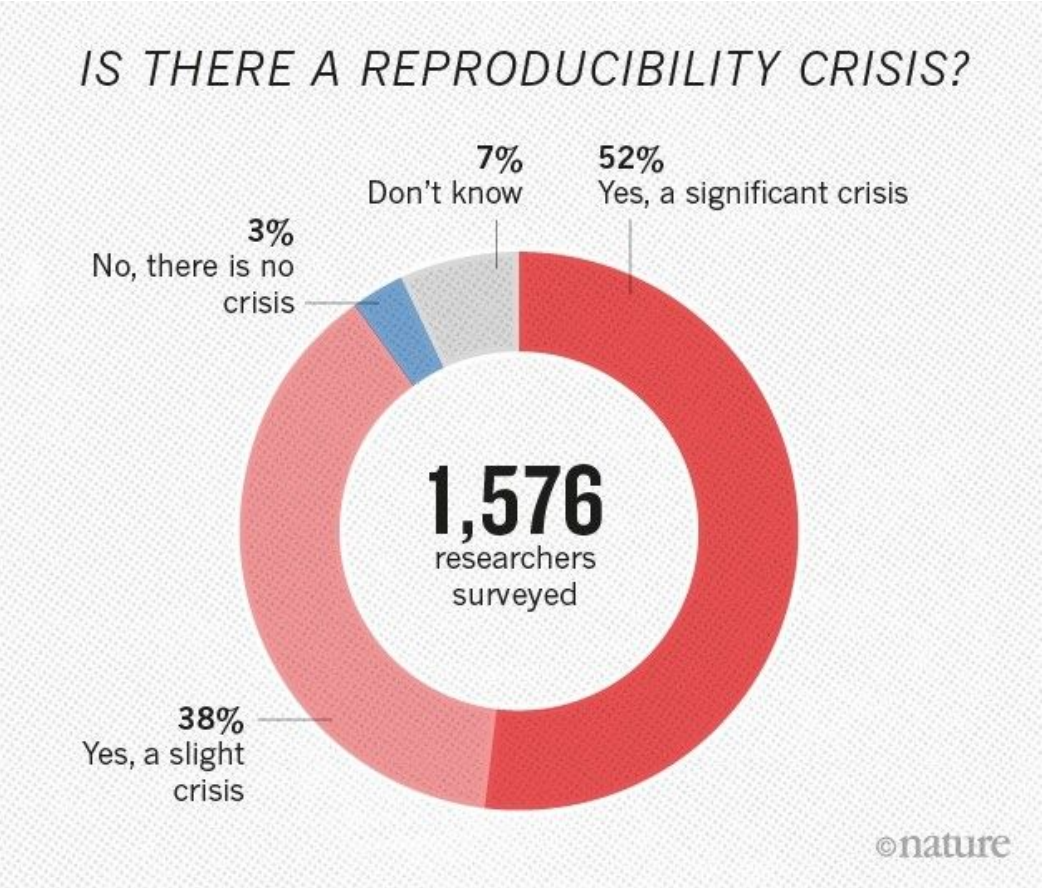
anos
impa
Instituto de
Matemática
Pura e Aplicada

1. Introduction

1. Introduction

- Scientific Method
  - Reproduction
  - Reproducibility crisis
  - ReScience

RESCIENCE C



1. Introduction

- The 80s - Exciting times
  - TeX released
  - Adobe PostScript
  - Desktop publishing

- Curve-fitting with piecewise parametric cubics
  - SIGGRAPH 1983

- Can this paper be reproduced?

Computer Graphics Volume 17, Number 3 July 1983

Curve-Fitting with Piecewise Parametric Cubics

Michael Plass and Maureen Stone
IMAGING SCIENCES LABORATORY
Xerox Palo Alto Research Center

Abstract: Parametric piecewise-cubic functions are used throughout the computer graphics industry to represent curved shapes. For many applications, it would be useful to be able to reliably derive this representation from a closely spaced set of points that approximate the desired curve, such as the input from a digitizing tablet or a scanner. This paper presents a solution to the problem of automatically generating efficient piecewise parametric cubic polynomial approximations to shapes from sampled data. We have developed an algorithm that takes a set of sample points, plus optional endpoint and tangent vector specifications, and iteratively derives a single parametric cubic polynomial that lies close to the data points as defined by an error metric based on least-squares. Combining this algorithm with dynamic programming techniques to determine the knot placement gives good results over a range of shapes and applications.

CR Categories and Subject Descriptors: G.1.2 [**Numerical Analysis**]: Approximation–Least squares approximation; Spline and piecewise polynomial approximation; G.1.5 [**Numerical Analysis**]: Roots of Nonlinear Equations–Iterative methods; I.2.8 [**Artificial Intelligence**]: Problem Solving, Control Methods and Search–Dynamic programming; I.3.3 [**Computer Graphics**]: Picture/Image Generation–Digitizing and scanning; I.3.5 [**Computer Graphics**]: Computational Geometry and Object Modeling–Curve, surface, solid, and object representations

General Terms: Algorithms

1. Introduction.

We are interested in the problems that arise when trying to use two-dimensional curved shapes in an interactive design environment. In particular, we are interested in finding the best ways for a designer to define such shapes. Our immediate focus is the production of publication-quality documents. The documents are designed using an interactive system with a raster display, then camera-ready copy is digitally produced at very high resolutions. Our design methodology is to represent all shapes analytically using piecewise parametric cubic polynomials [27]. There are many published methods for specifying such curves [2,12,15,23]. These methods all have in common the approach that the shape designer provides a small set of controlling points and parameters which act as handles to shape the curve. At Xerox PARC such algorithms, and similar ones for conics, have been used in design systems for making illustrations [4,3,16,26] and digital typefaces [5]. We conclude that it would be a powerful addition to the set of tools for specifying shapes to be able to reliably derive an efficient piecewise parametric cubic polynomial representation from a sequence of closely-spaced points that approximate the desired curve. The points could be produced in a variety of ways such as sketching with a digitizing pen, scanning and finding edges in an existing drawing, or writing a program to compute a set of data points. In this paper we will present a method deriving such an analytical representation from digitized data.

For the data in our curve-fitting work we have experimented with hand-tuned digital curves, optically-scanned images and simple hand-drawn sketches. We have found that the source of the digitized curve is relatively unimportant. The differences principally affect how tightly you want to follow the data; is it noisy, does it have pronounced rastering effects, are corners well defined? A scanned image will not have as sharp corners as a hand-tuned bitmap, for example. Also, the tolerance is a function of the image resolution. We feel our results are generally applicable to the problem of generating piecewise approximations to shapes from an ordered set of discrete data points.

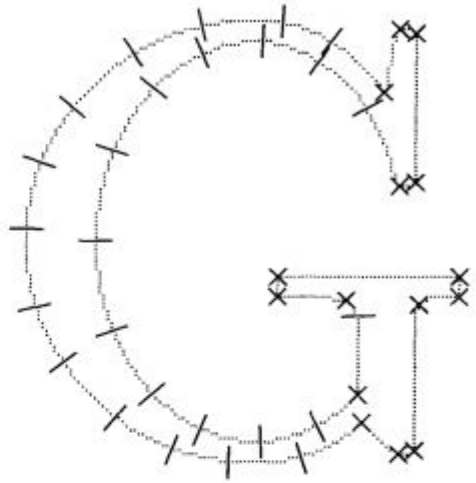
The heart of our method is an algorithm that takes a set of sample points and derives a parametric cubic curve that lies close to the data points; this algorithm is presented in section 4. A more elaborate version of the algorithm,

1. Introduction

- Main idea
  - Given a sample of points
  - Find cubic spline

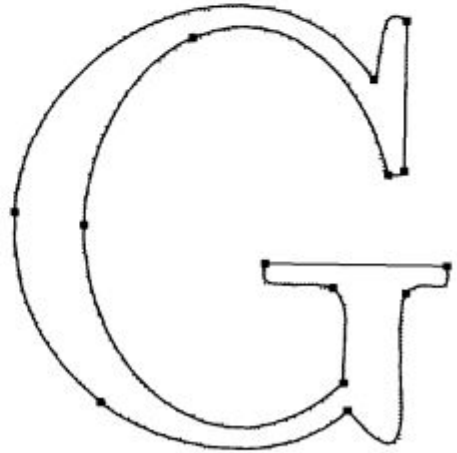
- Applications
  - Find SVG of input image
  - Digital typography

(a): Original data, approximately 150 by 160 bits.



(b): Samples, potential knots and normal vectors

(c): Result of curve fitting algorithm, 14 cubic pieces.



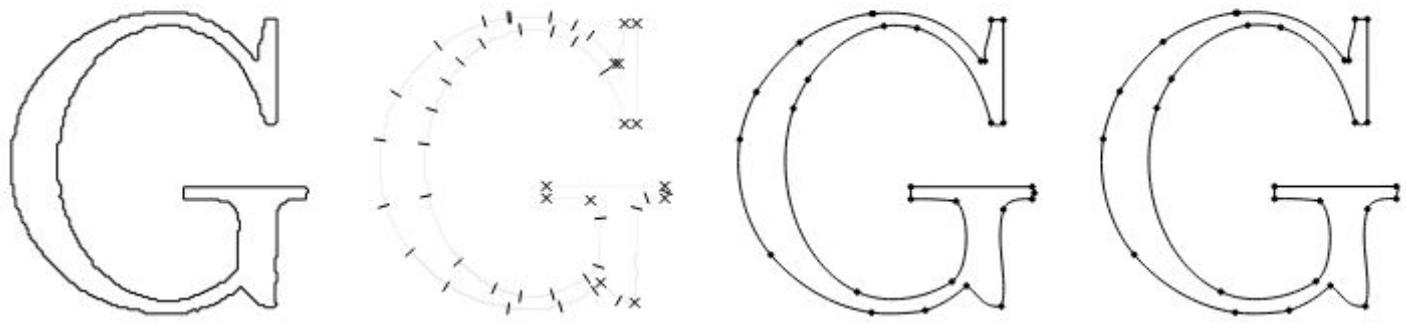
(d): Samples, curve and knots.

1. Introduction

- Typography
  - Digital font converted from bitmaps

1. Introduction

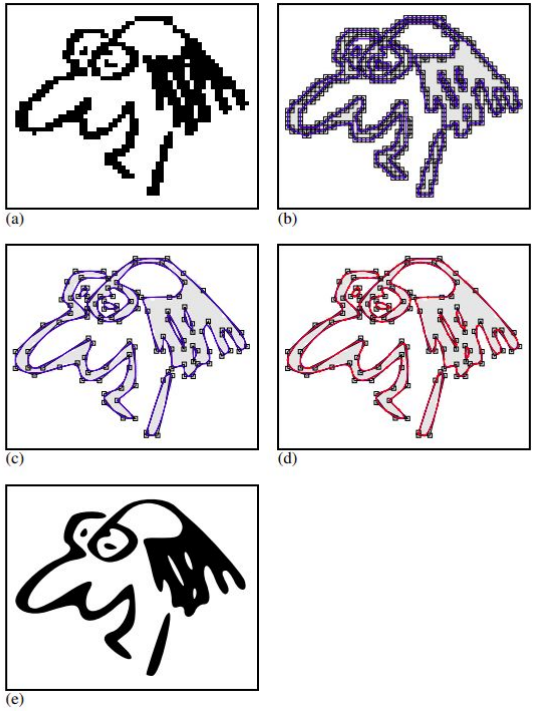
- Overview of the algorithm
  - Global and local
  - Perform least-squares to a set of knots and samples
  - Dynamic programming to find best set of knots



2. Data

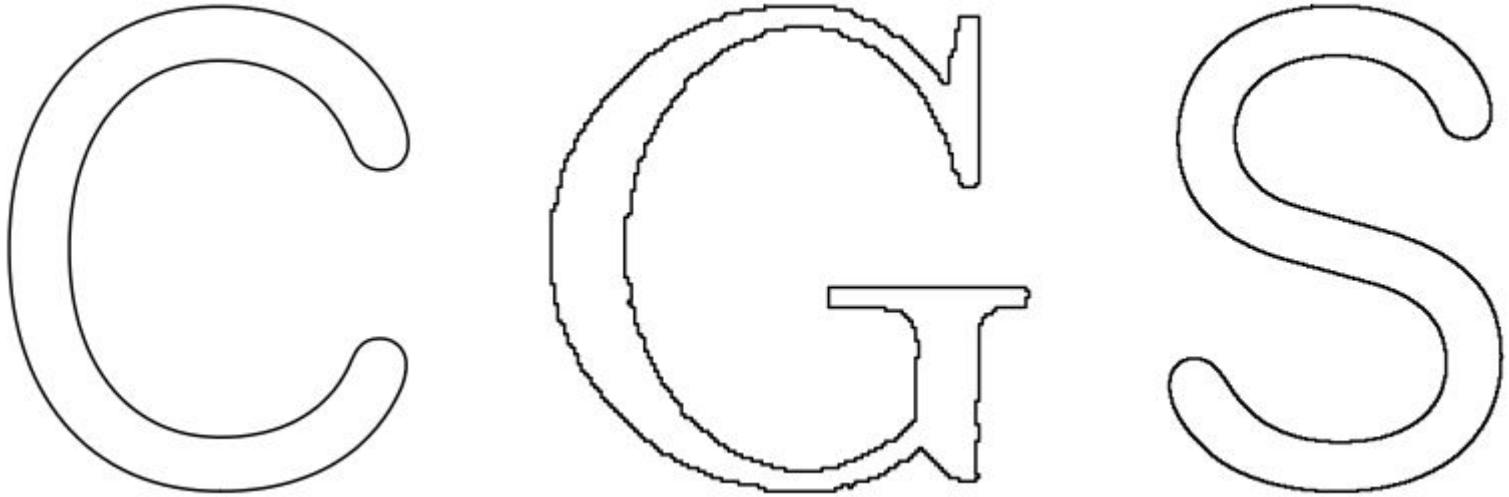
2. Data

- Hand-drawn images
- Glyph images



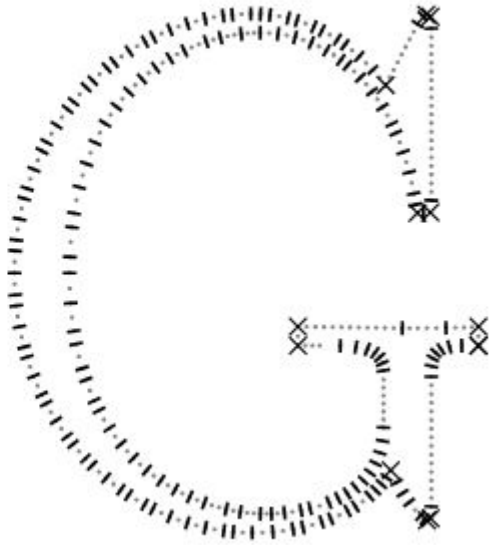
2. Our Data

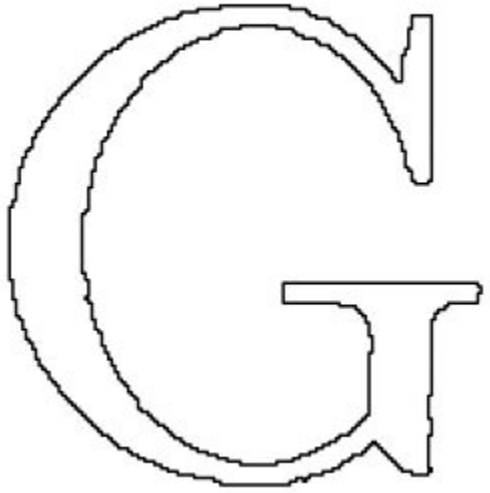
- Extracted sequences of points from glyphs
  - From outline
    - Exact
  - From image
    - Noisy

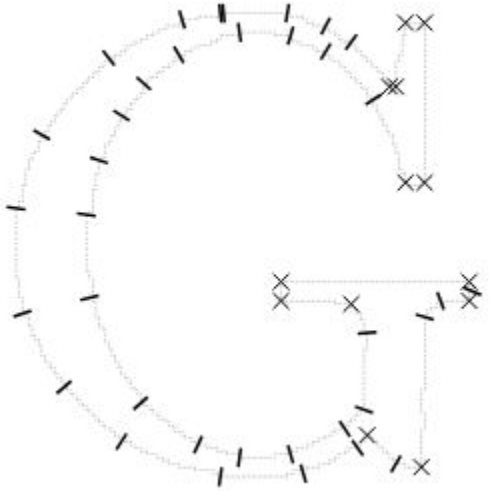


2. Our Data

- Potential knots and corners
  - Corners angle selection
  - Polygon simplification
    - Ramer–Douglas–Peucker

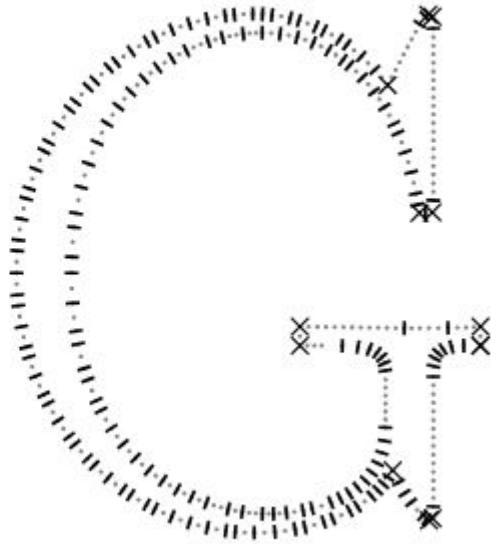


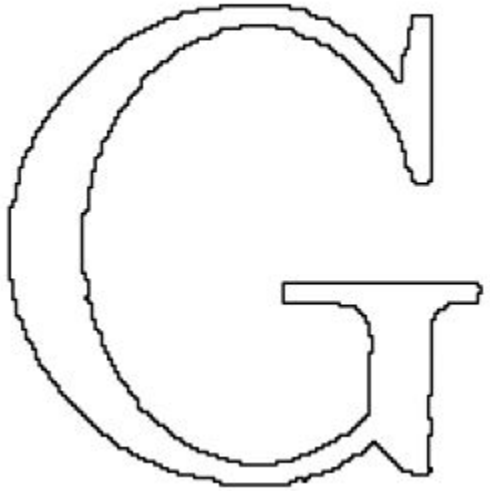


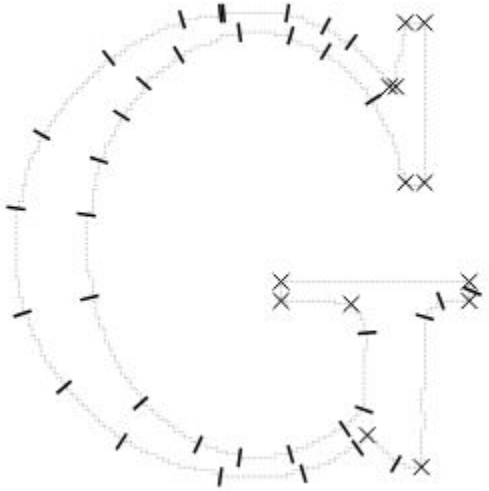


2. Data Preprocessing

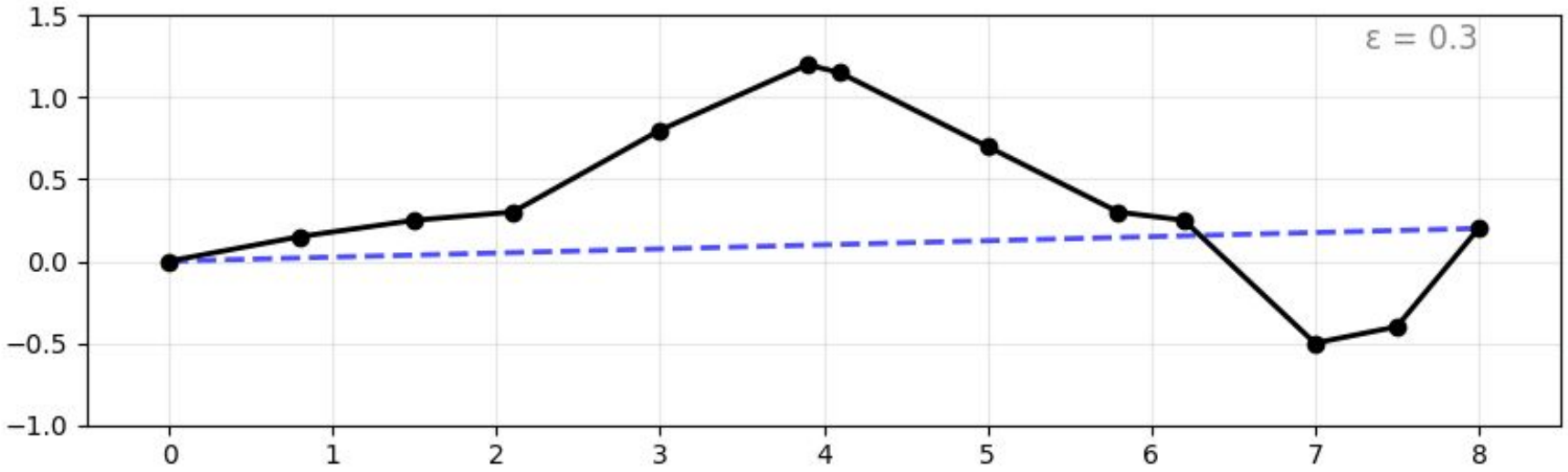
- Potential knots and corners
  - Knot selection
    - Polygon simplification
    - Ramer–Douglas–Peucker
  - Corners selection
    - Angle < 135°
  - For knots
    - Compute tangent
    - Local least-squares



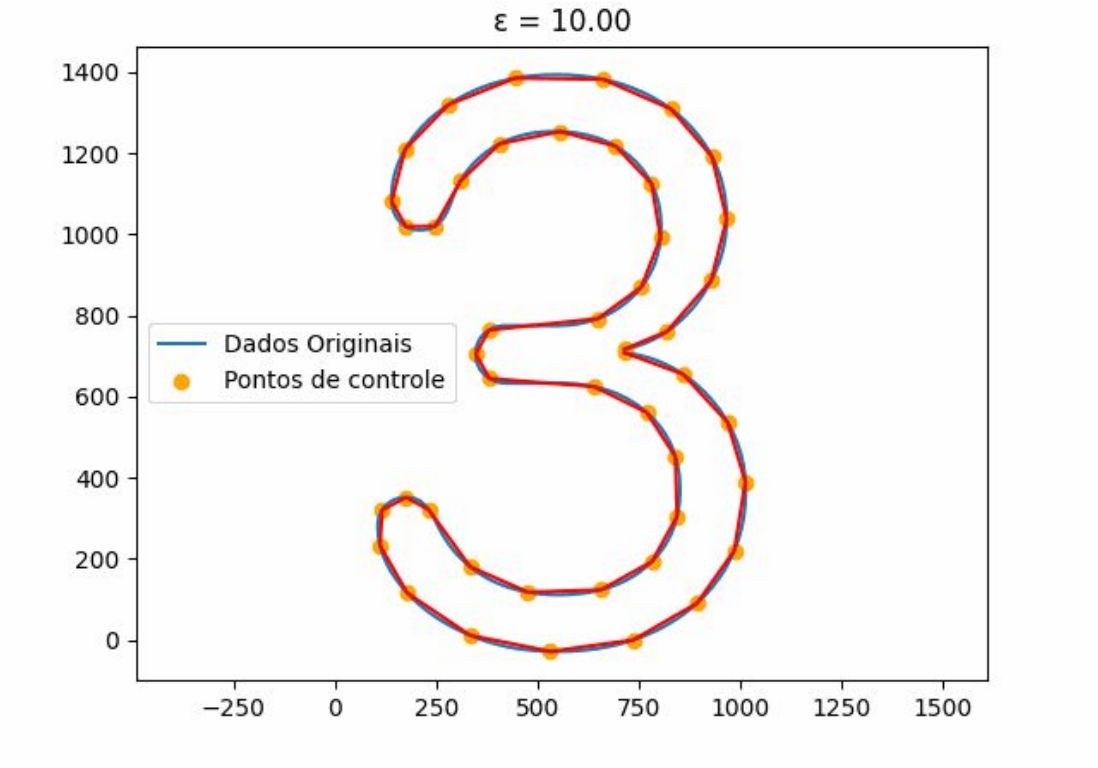




2. Ramer–Douglas–Peucker



2. Ramer–Douglas–Peucker



3. Fitting

3. Fitting

- Goal
  - Parametric curve *γ(t)* approximates input points

$$\gamma(t_i) \approx p_i$$

- Steps
  - Initialize values of t_i
  - Least-squares fitting
  - Update t_i

3. Fitting

- How do we initialize the values of t_i for each point p_i?
  - Take t_i as the normalized cumulative length of the chords up to p_i

$$t_i = \frac{s_i}{s_n}, \quad \text{where} \quad s_i = \sum_{k=1}^{i-1} \|p_{k+1} - p_k\|, \quad i = 1, \ldots, n.$$

3. Fitting

Bézier curve

$$\gamma(t) = (1-t)^3 B_0 + 3(1-t)^2 t B_1 + 3(1-t) t^2 B_2 + t^3 B_3$$

- Where does our curve begin and where does it end?
- Are we dealing with corners or ordinary knots?

3. Fitting

Approximation error

$$\sum_{i=1}^{n} \|\gamma(t_i) - p_i\|^2$$

- We set the endpoints at $B_0 = p_1$ and $B_3 = p_n$
- Let $B_1 = (x_1, y_1)$ and $B_2 = (x_2, y_2)$
- We use least squares to find B_1 and B_2, but we must consider three cases

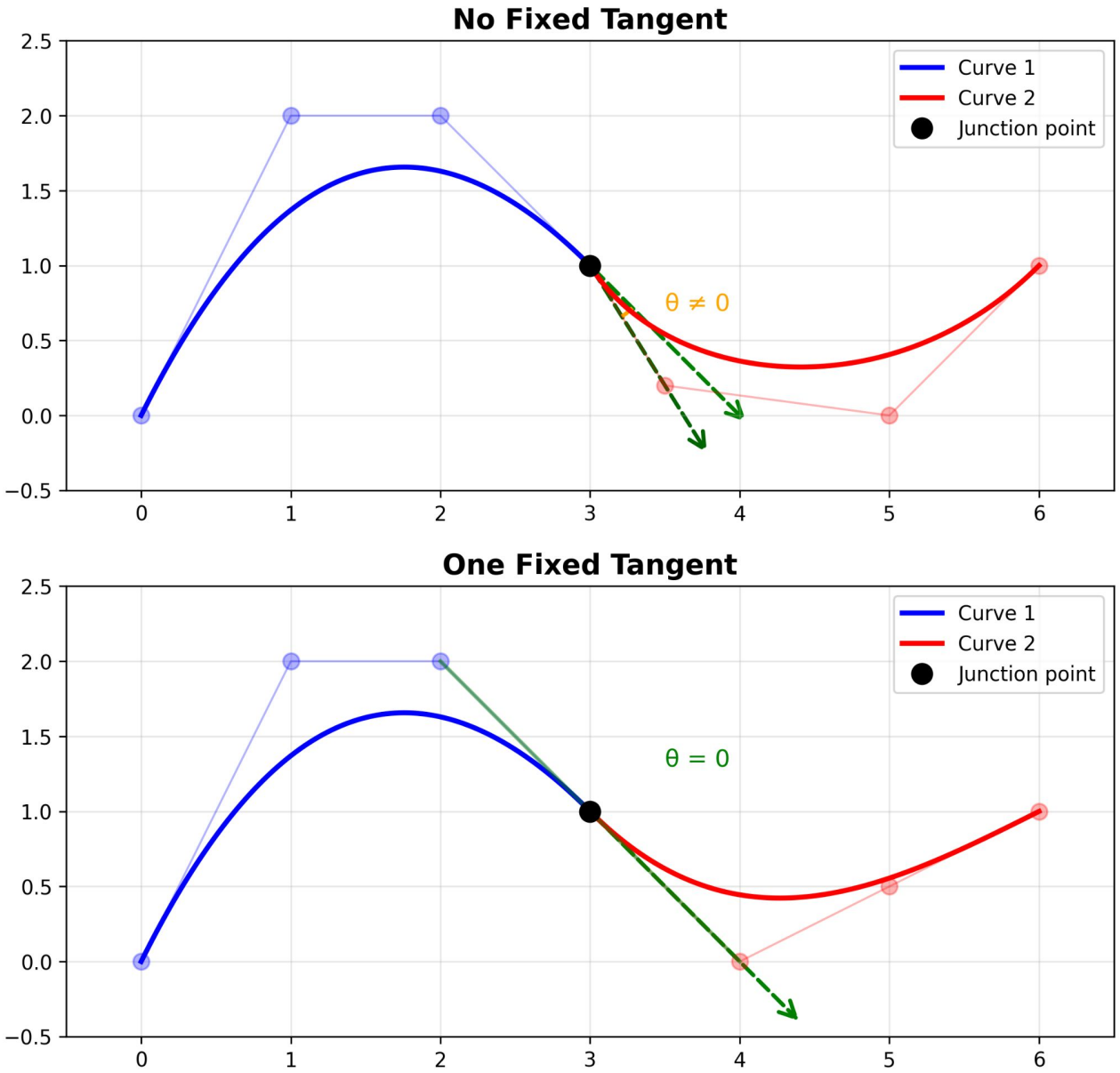
3. Fitting

- Our curve is between two corners: *No tangent directions.*
  The optimization variables are x_1, y_1, x_2, y_2.

- The curve is touching only one corner: *One tangent direction.*
  When the tangent direction is T_0, we have $B_1 = p_1 + \alpha_1 T_0$ and the optimization variables are x_2, y_2, α_1.
  When the tangent direction is T_3, we have $B_2 = p_n + \alpha_2 T_3$ and the optimization variables are x_1, y_1, α_2.

- The curve does not touch corners: *Two tangent directions.*
  We have $B_1 = p_1 + \alpha_1 T_0$ and $B_2 = p_n + \alpha_2 T_3$ and the optimization variables are α_1, α_2.

3. Fitting


No Fixed Tangent
Curve 1
Curve 2
Junction point
θ ≠ 0
One Fixed Tangent
Curve 1
Curve 2
Junction point
θ = 0

3. Fitting

- With the estimated curve, we update the values of t_i for each point p_i such that $\gamma(t_i)$ is the point on the curve closest to p_i

- To do this, we minimize

$$g_i(t) = \|\gamma(t) - p_i\|^2$$

$$f_i(t) = g_i'(t) = (\gamma(t) - P_i) \cdot \gamma'(t)$$

3. Fitting

- We use Newton-Raphson to find the root of f_i

$$t_i \leftarrow t_i - \frac{f_i(t)}{f_i'(t)}$$

$$f_i'(t) = \gamma'(t) \cdot \gamma'(t) + (\gamma(t) - P_i) \cdot \gamma''(t)$$

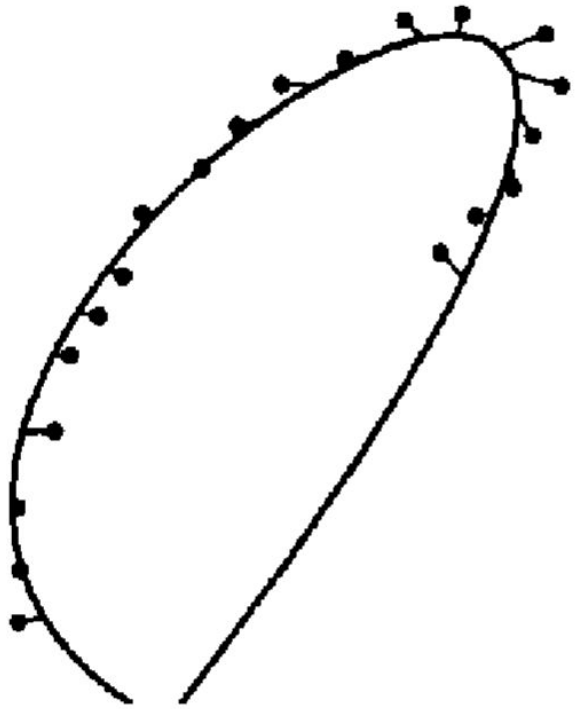
- 14 steps

3. Fitting

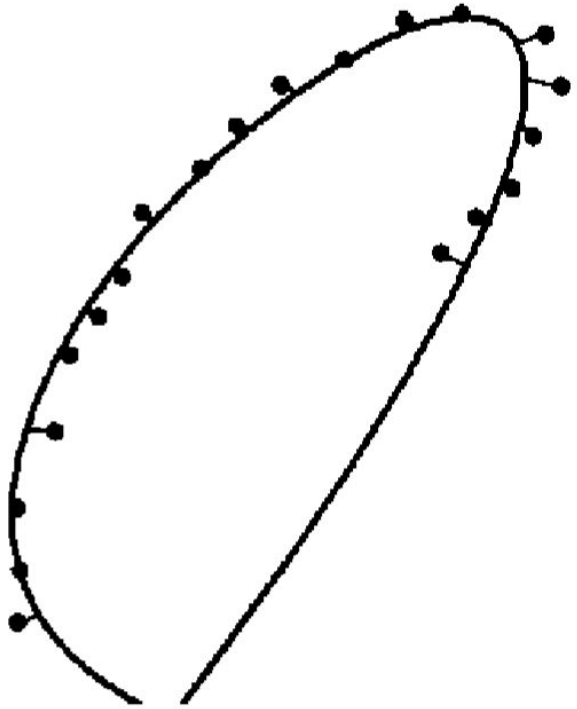
- And finally, we normalize the values of t_i to remain in the range [0, 1].

$$t \leftarrow \frac{t - t_{min}}{t_{max} - t_{min}}$$

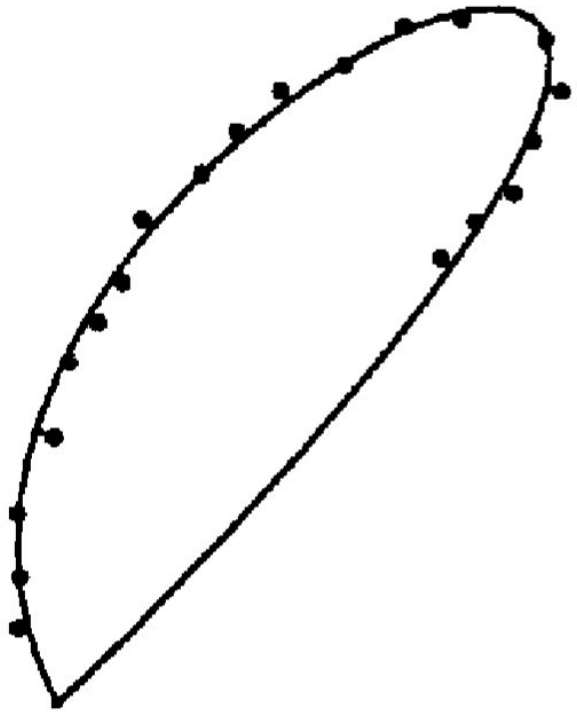
3. Fitting



Initial fit.



1 iteration.

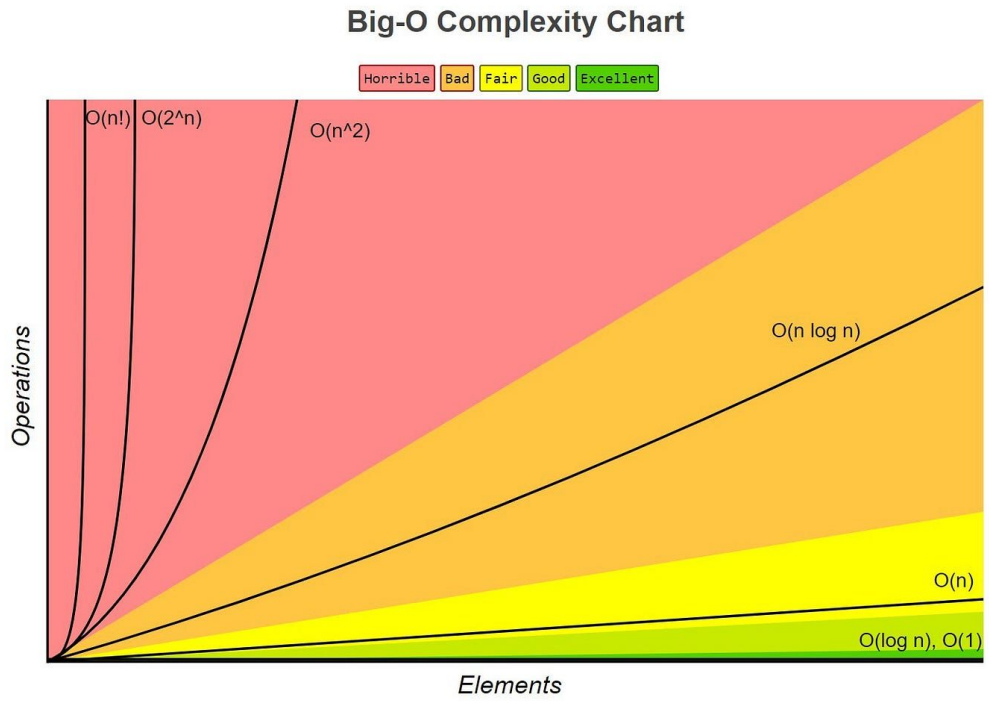


10 iterations.

4. Best Knots

4. Best Knots

- Given the set of knots
  - How can we get the best set of knots?
  - We could try every combination of knots
    - Exponential number of fittings



4. Best Knots

- Dynamic Programming
  - Use a table and grow the result
  - From a base case until our desired case

- Define error

$$e_{ij} = \sum_{k=i}^{j} \|\gamma(t_k) - p_k\|^2 + \tau$$

- For each curve between input points *i* and *j*

- Add a tolerance τ so that the curve does not degenerate

4. Best Knots

- Dynamic Programming
  - Use a table and grow the result
  - From a base case until our desired case

$$e_{ij} = \sum_{k=i}^{j} \|\gamma(t_k) - p_k\|^2 + \tau$$

error	1	2	3	4
1	0	15	20	30
2	x	0	10	20
3	x	x	0	4
4	x	x	x	0

4. Best Knots

- Construct a vector given this minimization:

$$D[i] = \min_{1 \le k < i} (D[k] + e_{ki})$$

- Perform this in a table

error	1	2	3	4
1	0	15	20	30
2	x	0	10	20
3	x	x	0	4
4	x	x	x	0

	1	2	3	4
D	0			
Previous	None			

4. Best Knots

- Construct a vector given this minimization:

$$D[i] = \min_{1 \le k < i} (D[k] + e_{ki})$$

- Perform this in a table

error	1	2	3	4
1	0	15	20	30
2	x	0	10	20
3	x	x	0	4
4	x	x	x	0

	1	2	3	4
D	0	15		
Previous	None	1		

4. Best Knots

- Construct a vector given this minimization:

$$D[i] = \min_{1 \le k < i} (D[k] + e_{ki})$$

- Perform this in a table

error	1	2	3	4
1	0	15	20	30
2	x	0	10	20
3	x	x	0	4
4	x	x	x	0

	1	2	3	4
D	0	15	20	
Previous	None	1	1	

4. Best Knots

- Construct a vector given this minimization:

$$D[i] = \min_{1 \le k < i} (D[k] + e_{ki})$$

- Perform this in a table

error	1	2	3	4
1	0	15	20	30
2	x	0	10	20
3	x	x	0	4
4	x	x	x	0

	1	2	3	4
D	0	15	20	24
Previous	None	1	1	3

4. Best Knots

- Construct a vector given this minimization:

$$D[i] = \min_{1 \le k < i} (D[k] + e_{ki})$$

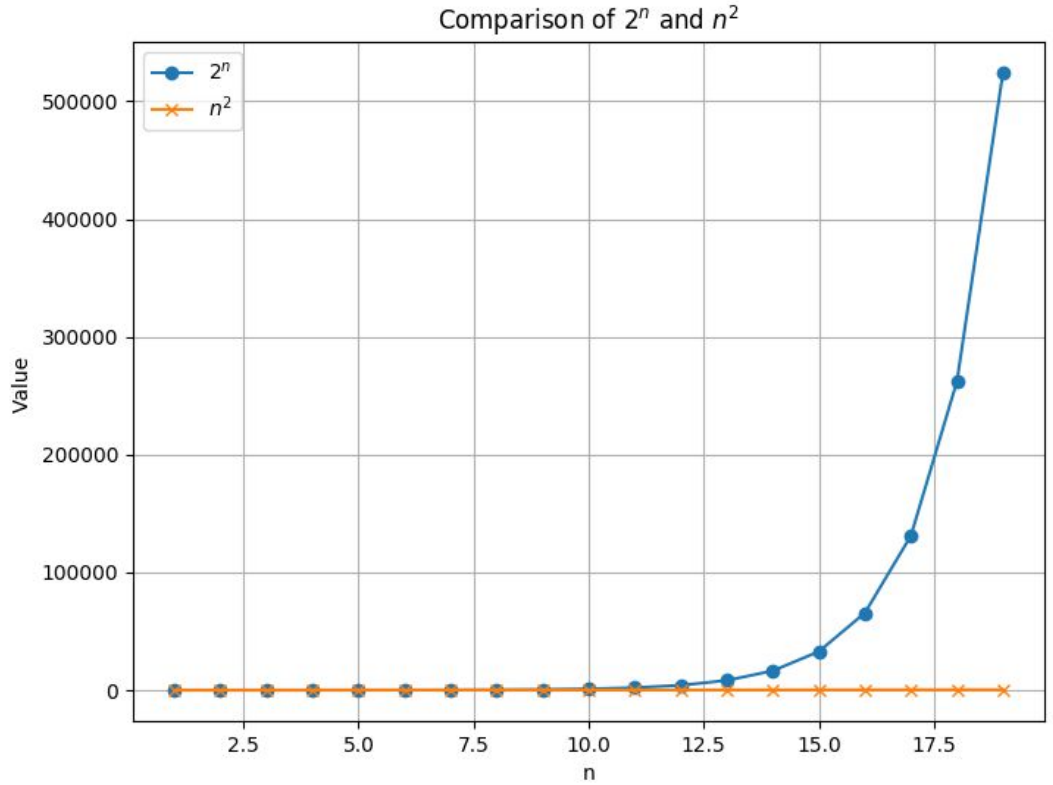
- Perform this in a table
- Perform back-tracking
  - Find path
  - $1 \rightarrow 3 \rightarrow 4$

error	1	2	3	4
1	0	15	20	30
2	x	0	10	20
3	x	x	0	4
4	x	x	x	0

	1	2	3	4
D	0	15	20	24
Previous	None	1	1	3

4. Best Knots

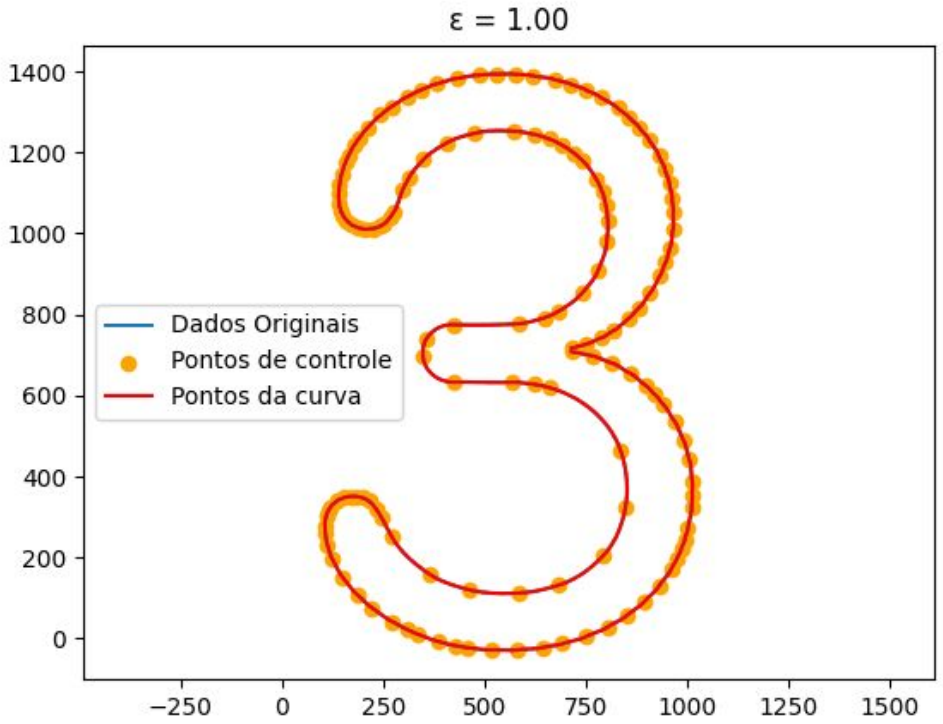
- Instead of exponential
  - We have quadratic



5. Results

5. Results

- Manage to create a visualization from the curve



5. Results

- Managed to reconstruct the curve

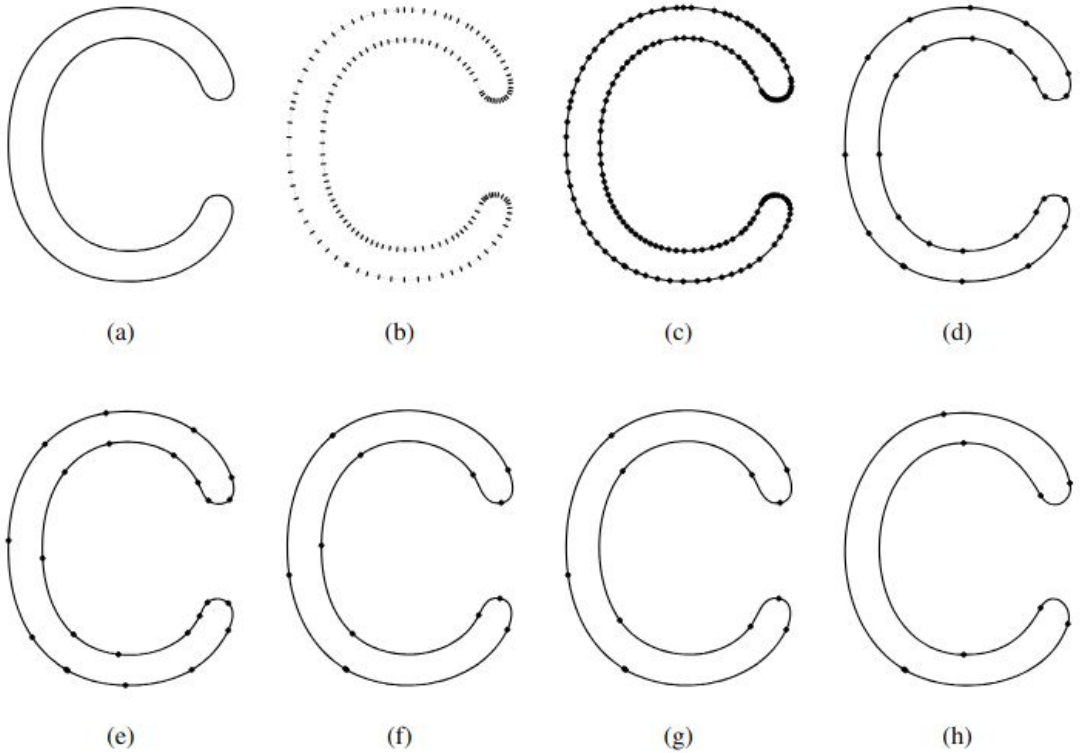


Figure 1: Results using sliding window for tangent computation (5 points before and after each center point, $\varepsilon = 1$). (a) Original data. (b) Potential knots and corners. (c)–(h) Processed curves: (c) $\tau = 0$, (d) $\tau = 4$, (e) $\tau = 20$, (f) $\tau = 320$, (g) $\tau = 3200$, (h) $\tau = 8200$.

5. Results

- Managed to reconstruct the curve

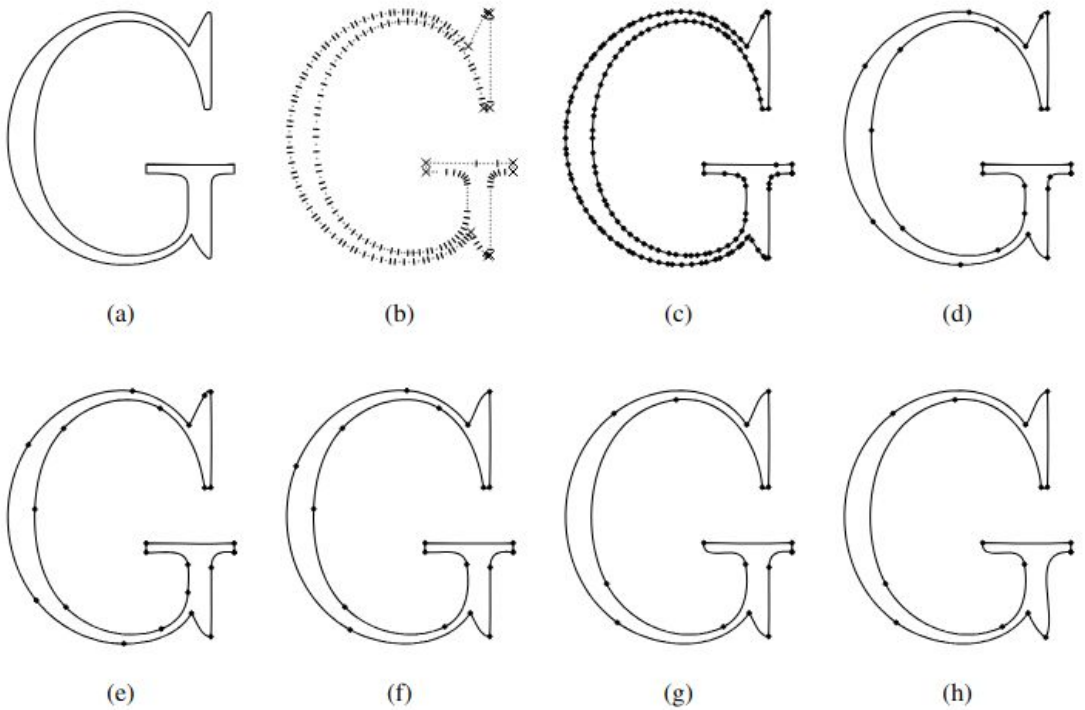


Figure 2: Results using sliding window for tangent computation (5 points before and after each center point, $\varepsilon = 0.25$). (a) Original data. (b) Potential knots and corners. (c)–(h) Processed curves: (c) $\tau = 0$, (d) $\tau = 6$, (e) $\tau = 13$, (f) $\tau = 20$, (g) $\tau = 100$, (h) $\tau = 130$.

5. Results

- Managed to reconstruct the curve

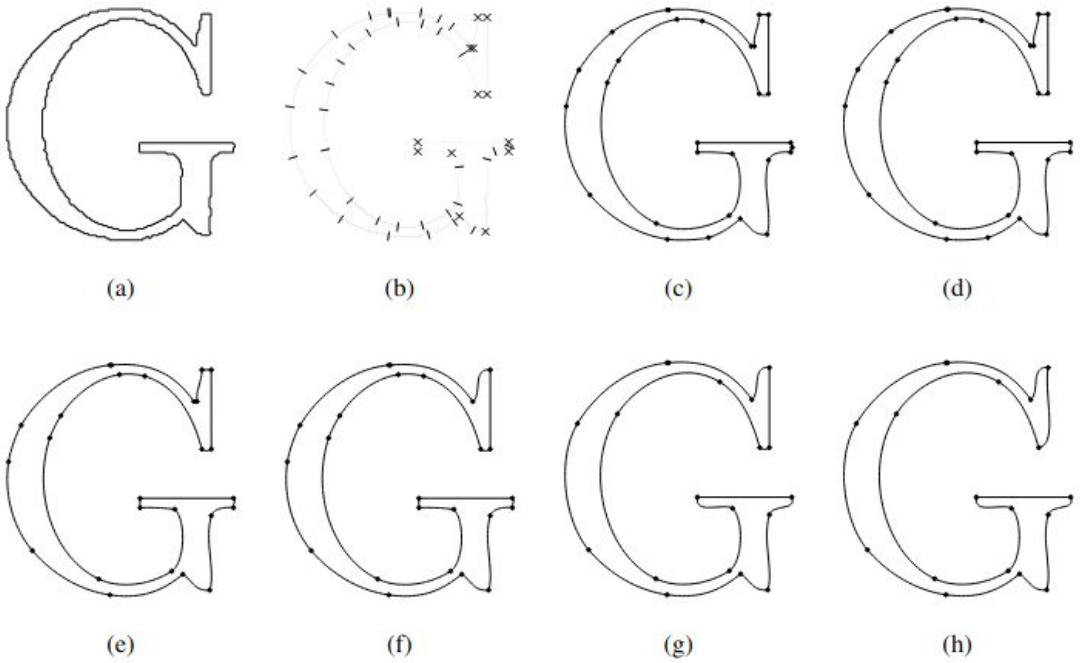


Figure 3: Results using sliding window for tangent computation (25 points before and after each center point, $\varepsilon = 1.75$). (a) Original data. (b) Potential knots and corners. (c)–(h) Processed curves: (c) $\tau = 0$, (d) $\tau = 0.5$, (e) $\tau = 5$, (f) $\tau = 7$, (g) $\tau = 76$, (h) $\tau = 100$.

5. Results

- Managed to reconstruct the curve

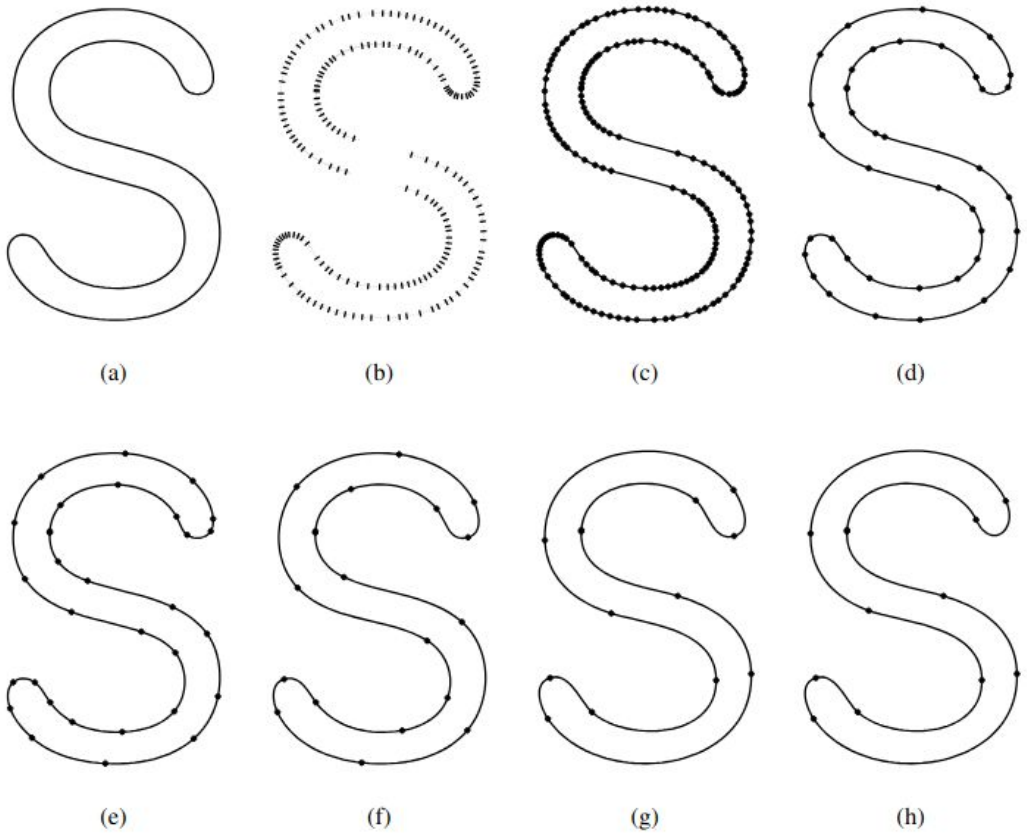


Figure 4: Results using sliding window for tangent computation (5 points before and after each center point, $\varepsilon = 1$). (a) Original data. (b) Potential knots and corners. (c)–(h) Processed curves: (c) $\tau = 0$, (d) $\tau = 1$, (e) $\tau = 10$, (f) $\tau = 160$, (g) $\tau = 2800$, (h) $\tau = 8500$.

5. Results

- Managed to reconstruct the curve

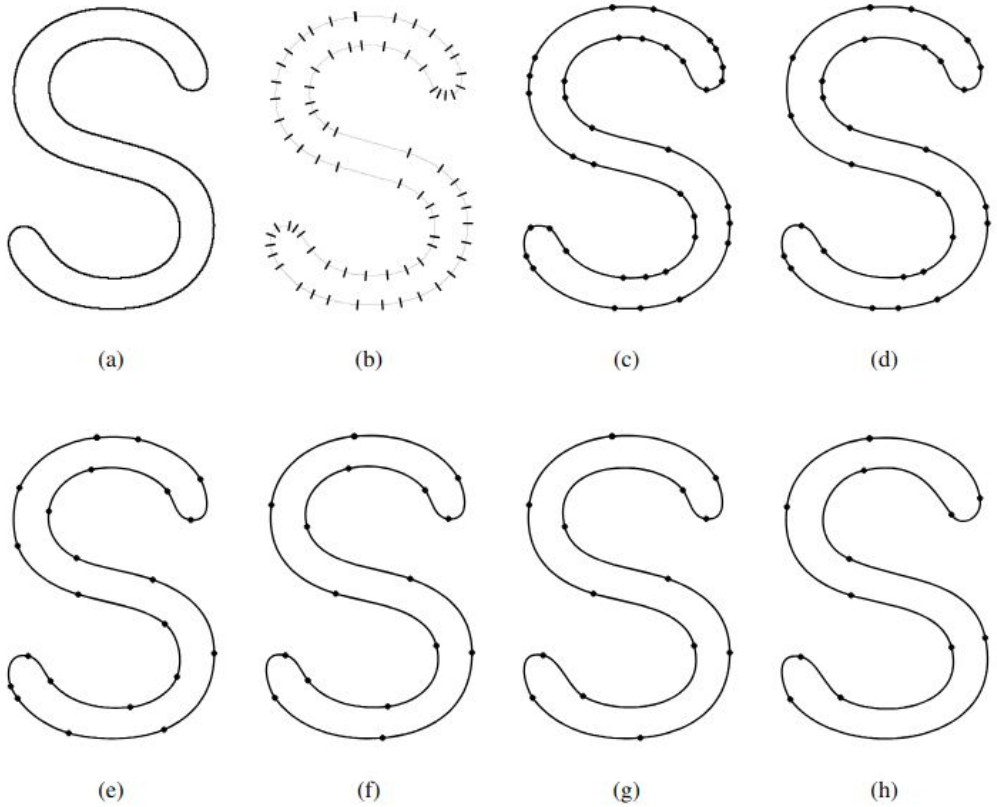
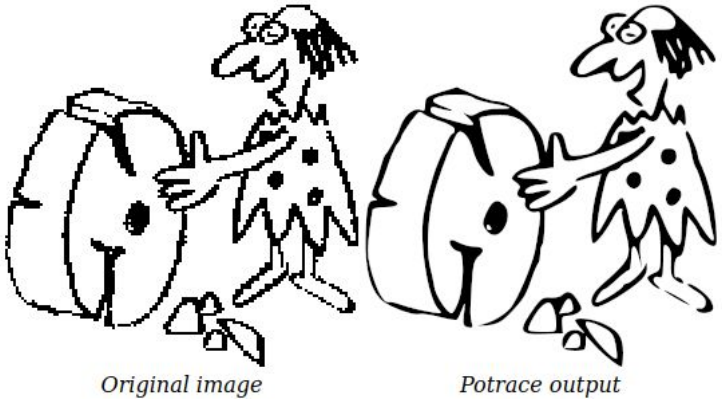


Figure 5: Results using sliding window for tangent computation (15 points before and after each center point, $\varepsilon = 1.5$). (a) Original data. (b) Potential knots and corners. (c)–(h) Processed curves: (c) $\tau = 0$, (d) $\tau = 10$, (e) $\tau = 50$, (f) $\tau = 210$, (g) $\tau = 450$, (h) $\tau = 2500$.

6. Conclusion

6. Conclusion

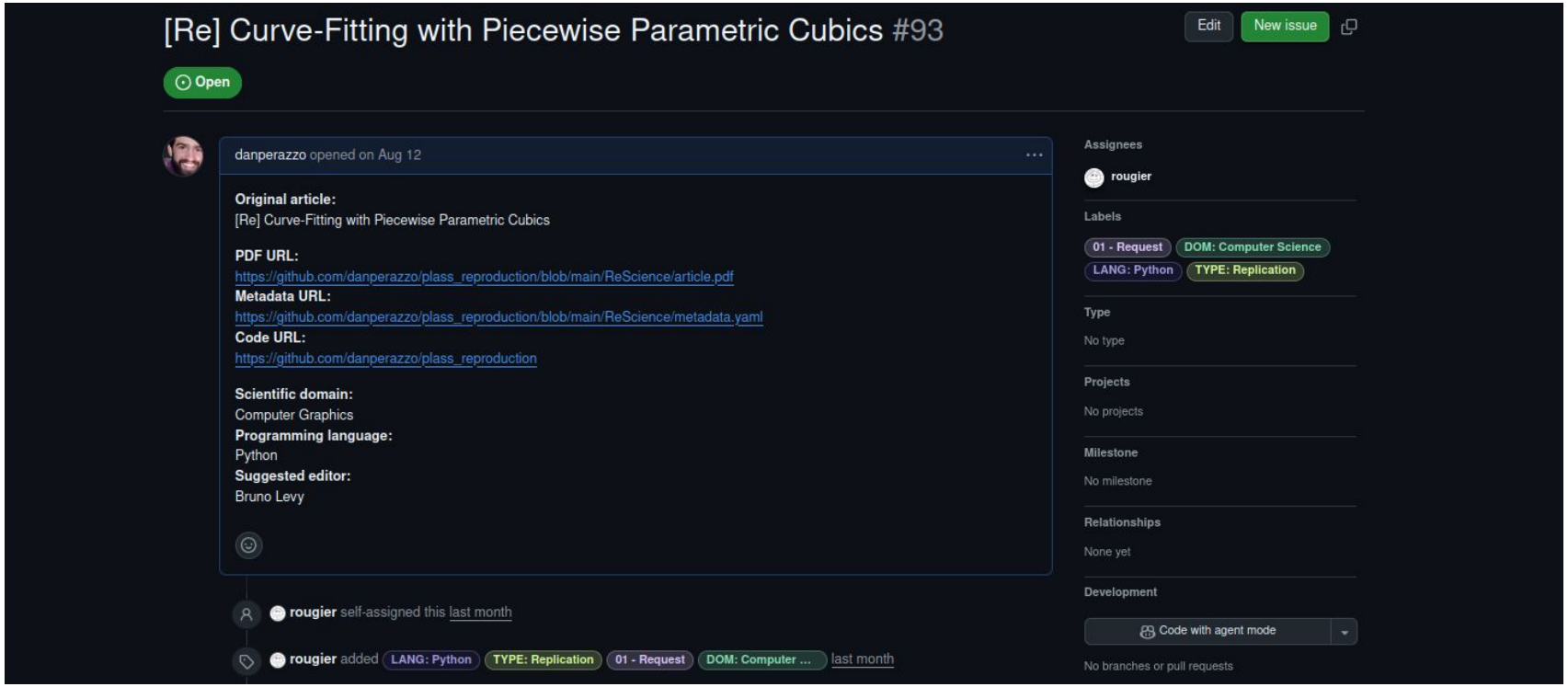
- Lots of current applications!
  - Adobe Photoshop
  - Adobe illustrator
  - Potrace



Original image *Potrace output*

6. Conclusion

- Started submission of replication study!



[Re]
Curve-fitting with piecewise parametric cubics

Daniel, Davi

75 anos impa
Instituto de
Matemática
Pura e Aplicada