Simplification of Indoor Space Footprints

Joon-Seok Kim  
Department of Geography and Geoinformation Science  
George Mason University  
Fairfax, VA 22030  
jkim258@gmu.edu

Carola Wenk  
Department of Computer Science  
Tulane University  
New Orleans, LA 70118  
cwenk@tulane.edu

Abstract

Simplification is one of the fundamental operations used in geoinformation science (GIS) to reduce size or representation complexity of geometric objects. Although different simplification methods can be applied depending on one’s purpose, a simplification that many applications employ is designed to preserve their spatial properties after simplification. This article addresses one of the 2D simplification methods, especially working well on human-made structures such as 2D footprints of buildings and indoor spaces. The method simplifies polygons in an iterative manner. The simplification is segment-wise and takes account of intrusion, extrusion, offset, and corner portions of 2D structures preserving its dominant frame.

Keywords  
simplification · building · indoor space · footprint

1 Introduction

Simplification is one of the methods used for generating data at different levels of detail (LoDs) from precise data. The more compact size of simplified data is desirable in a variety of data processing tasks including data transmission. Let P be a polygon, P' be a simplified polygon, and D(P, P') be the distance between P and P'. The computational geometry community distinguishes between two variants of curve simplification: While the min–# problem is to find P' with the minimum number of vertices such that D(P, P') ≤ ε, the min–ε problem is to find P' of at most k vertices such that D(P, P') is minimized. Depending on the constraints on the location of vertices of P', the problem can be categorized into (1) vertex-restricted, (2) curve-restricted, and (3) non-restricted simplification, see [1]. The Ramer-Douglas-Peucker (RDP) algorithm [2, 3] is widely used in practice and provides a vertex-restricted approximation to simplify 2D polylines and polygons. However, it does not provide any quality guarantees and it may not preserve essential shape of the entire footprint because it does not reflect a specific form (see Figure 1b). Although Figures 1b and 1c are similar in terms of the number of segments, the figures demonstrate the RDP cannot preserve spatial features such as intrusion, extrusion, offset, and corners. In geographic information science (GIS), simplification is considered a type of generalization process [4]. These processes or operations consider not only metric constraints but also topological, semantic, and Gestalt constraints. In particular, Gestalt constraints are used to preserve the characteristics of spatial features such as a room.

This article presents a method that progressively simplifies 2D polygons by preserving their spatial properties in an iterative manner. The method was originally introduced by Kim and Li [5] to simplify 3D indoor spaces (e.g., rooms and hallways) that can be represented with the prism model [6, 7], which is an alternative 3D data model. As shown in Figure 1c the simplification can be applied to footprints of complex buildings such as shopping malls or subway

*This research was supported by National Science Foundation grant CCF 1637576.
Figure 2 depicts the flow of the simplification algorithm that considers the rules, and Algorithm 1 shows the corresponding pseudo-code. In order to simplify shorter segments first in accordance with Rule-1, all segments of \( P \) are

\[ s_i = (p_i, p_{i+1}) \]  

for all \( 0 \leq i < n \). The sequence of vertices of \( P \) is circular such that \( p_i = p_{(i \mod n)} \) for \( i \in \mathbb{Z} \). Let \( s_i \) be the length of \( s_i \), and let \( \hat{s}_i \) be the internal angle between two consecutive segments \( s_i \) and \( s_{i+1} \), i.e., \( \hat{s}_i = \angle p_ip_{i+1}p_{i+2} \). In the following, \( Q \) denotes a priority queue containing line segments \( s_i \) sorted by their length \( s_i \). Given a distance threshold \( \tau \), an angle threshold \( \varepsilon \), a collinearity threshold \( \delta \), and a joining distance threshold \( \gamma \), the simplification is designed to preserve the overall shape of a 2D polygon according to the following rules:

- **Rule-1**: The shorter the line segment, the smaller the effect on the overall shape of the geometry.
- **Rule-2**: Only segments longer than the tolerance \( \tau \) are considered to reflect the overall shape of the polygon.
- **Rule-3**: Consistent simplifications should be performed for feature types such as intrusion/extrusion, offset, and corner.
Algorithm 1: Simplification

```plaintext
input : P // Polygon to simplify
input : τ // Tolerance distance
input : ε // Angle threshold used to determine if consecutive segments are collinear
input : δ // Distance threshold used to determine whether to join neighboring segments

1. Q ← 0; // Initialize a priority queue
2. foreach s ∈ P do // Insert all segments of polygon P into the queue Q
3.  Q ← Q ∪ s;
4. while |Q| ≥ 0 and |P| ≥ 3 do // Dequeue next segment
5.  s_i ← Q.dequeue();
6.  α ← |Δs_i|; // Angle difference between the angles entering and leaving s_i
7.  if π − δ < s_i < π + δ then
8.     RemoveMiddlePoint(s_i); // Merge two approximately collinear consecutive segments
9.  else if π ≤ s_i ≤ ε then
10.     SegmentRegression(s_i);
11.   else if π − α ≤ ε then
12.     TranslateSegment(s_i);
13. else
14.     q ← Intersect(s_i−1, s_i+1); // Intersection of two lines obtained by extending s_i−1 and s_i+1
15.     if dist(s_i, q) ≤ γ then
16.         JoinSegment(s_i, q);
17.     else if s_i−1 < s_i+1 then
18.         RemoveMiddlePoint(s_i−1);
19.     else
20.         RemoveMiddlePoint(s_i);
21. return P; // Return the simplified polygon
```

sorted by length and inserted into a priority queue Q (lines 1-3). Simplification is repeatedly carried out on the next segment in the queue until the queue is empty (line 4). The shortest line segment is dequeued from the queue as the current segment s_i (line 5). To merge collinear segments, it is checked whether π − δ < s_i < π + δ, where δ > 0 is a small angle threshold that is used to evaluate collinearity (line 7). If so, the middle point between s_i and s_i+1 is removed so that a new line segment between the two end points is created (line 8; see also Algorithm 2). If the length of current segment is less than or equal to τ, simplification is performed (lines 9-21). In consideration of Rule-3, one of four methods is chosen depending on the angle difference α = |Δs_i| ≤ π (lines 10-21). Figures 3-5 illustrate this process.

- If 0 ≤ α ≤ ε, regress two segments s_i−1 and s_i+1, considering their lengths and tangents as shown in Figures 3(c), (d) and (e). The detailed process is outlined in Algorithm 3.
- If π − α ≤ ε, this is considered an intrusion/extrusion, and the current segment is translated in order to remove the intrusion/extrusion as shown in Figure 4 (see Algorithm 4).
- Otherwise, it is checked whether the two segments can be joined by extending s_i−1 and s_i+1 until s_i−1 and s_i+1 intersect (see Figure 6). Let q be the intersection point, dist a distance function, and γ the distance threshold.
  - If dist(s_i, q) ≤ γ as shown in Figures 6(a)-(d), then remove s_i and join s_i−1 and s_i+1 (see Algorithm 5 and Figure 6(e)-(h)).
  - Otherwise, do not join the segments because the extended part of the segments is too long as shown in Figures 7(a)-(e). Instead, remove the middle point between s_i and s_i−1 (or s_i+1) (see Figure 7(f)-(k)).

Pseudocode for the functions invoked in Algorithm 1 is described as Algorithms 2-5. Assume that P and Q defined in Algorithm 1 can be accessed from Algorithms 2-5. Python implementation and examples can be found at the git repository (https://github.com/joonseok-kim/simplification).
Algorithm 2: RemoveMiddlePoint

\[ Q \leftarrow Q \setminus s_{k+1}; \]  // Remove \( s_{k+1} \) from the queue \( Q \)
\[ s' \leftarrow (p_k, p_{k+2}); \]  // Create a new segment \( (p_k, p_{k+2}) \) by merging \( s_k \) and \( s_{k+1} \)
\[ P \leftarrow P \setminus (s_k \cup s_{k+1}); \]  // Remove existing \( s_k \) and \( s_{k+1} \) from \( P \)
\[ Q \leftarrow Q \cup s'; \]  // Insert the new segment into the queue \( Q \)

Algorithm 3: SegmentRegression

\[ Q \leftarrow Q \setminus (s_{k-1} \cup s_{k+1}); \]  // Segment
\[ r \leftarrow s_{k-1}/(s_{k-1} - s_{k+1}); \]  // Ratio to be used as a weight.
\[ p \leftarrow s_k \cdot \text{PointAlong}(r); \]  // A point \( p \) on \( s_k \) such that \( p \) is at distance \( r \) from \( p_k \).
\[ \theta \leftarrow \tan(s_{k-1} \cdot r + s_{k+1} \cdot (1 - r)); \]  // The slope of a regression line for \( s_{k-1} \) and \( s_{k+1} \).
\[ q_1 \leftarrow \text{Projection}(s_{k-2}, p, \theta); \]  // Intersection of \( s_{k-2} \) with the line through \( p \) with slope \( \theta \).
\[ q_2 \leftarrow \text{Projection}(s_{k+2}, p, \theta); \]  // Intersection of \( s_{k+2} \) with the line through \( p \) with slope \( \theta \).
\[ s_k \leftarrow (q_1, q_2); \]  // A regression line segment for \( s_{k-1} \) and \( s_{k+1} \).
\[ s_{k-1} \leftarrow (p_{k-1}, q_1); \]  // Update \( s_{k-1} \)
\[ s_{k+1} \leftarrow (q_2, p_{k+2}); \]  // Update \( s_{k+1} \)
\[ Q \leftarrow Q \cup s_{k-1} \cup s_k \cup s_{k+1}; \]  // Add the new three segments into the queue

Algorithm 4: TranslateSegment

\[ Q \leftarrow Q \setminus (s_{k-1} \cup s_{k+1}); \]  // Segment
\[ \text{if } s_{k-1} < s_{k+1} \text{ then} \]
\[ p' \leftarrow p_{k+1} - s_{k-1}; \]  // Remove \( s_{k-1} \) and \( s_{k+1} \) from the queue \( Q \)
\[ s_k \leftarrow (p_{k-1}, p'); \]  // Translate vertex \( p_{k+1} \) by vector \( s_{k-1} \)
\[ s_{k+1} \leftarrow (p', p_{k+2}); \]  // Update \( s_k \)
\[ P \leftarrow P \setminus s_{k-1}; \]  // Update \( s_{k+1} \)
\[ Q \leftarrow Q \cup (s_k \cup s_{k+1}); \]  // Remove existing \( s_{k-1} \)
\[ \text{else if } s_{k-1} > s_{k+1} \text{ then} \]
\[ p' \leftarrow p_k - s_{k+1}; \]  // Translate vertex \( p_k \) by vector \( s_{k+1} \)
\[ s_{k-1} \leftarrow (p_{k-1}, p'); \]  // Update \( s_{k-1} \)
\[ s_k \leftarrow (p', p_{k+2}); \]  // Update \( s_k \)
\[ P \leftarrow P \setminus s_{k+1}; \]  // Remove existing \( s_{k+1} \)
\[ Q \leftarrow Q \cup (s_{k-1} \cup s_k); \]  // Add \( s_{k-1} \) and \( s_k \) into \( Q \)
\[ \text{else} \]
\[ s_k \leftarrow (p_{k-1}, p_{k+2}); \]  // Update \( s_k \)
\[ P \leftarrow P \setminus (s_{k-1} \cup s_{k+1}); \]  // Remove existing \( s_{k-1} \) and \( s_{k+1} \)
\[ Q \leftarrow Q \cup s_k; \]  // Add \( s_k \) into \( Q \)

Algorithm 5: JoinSegment

\[ Q \leftarrow Q \setminus (s_{k-1} \cup s_{k+1}); \]  // Segment
\[ \text{input: } s_k \]
\[ \text{input: } q \]  // Intersection point
\[ s_{k-1} \leftarrow (p_{k-1}, q); \]  // Remove \( s_{k-1} \) and \( s_{k+1} \) from the queue \( Q \)
\[ s_{k+1} \leftarrow (q, p_{k+2}); \]  // Update \( s_{k-1} \)
\[ P \leftarrow P \setminus s_k; \]  // Update \( s_{k+1} \)
\[ Q \leftarrow Q \cup (s_{k-1} \cup s_{k+1}); \]  // Remove existing \( s_k \)
\[ Q \leftarrow Q \cup s_{k-1} \cup s_k \cup s_{k+1}; \]  // Add \( s_{k-1} \) and \( s_{k+1} \) into \( Q \)
Figure 3: Removing points and merging into regression segment

Figure 4: Translating current segment

Figure 5: Removing of points and joining segments

Figure 6: Joining segments ($\gamma = \overline{s}$)
3 Experiments

IndoorGML data for the Lotte World Mall (LWM), see Figure 8, one of the most complex shopping malls in Seoul, is used to conduct a comparative experiment on the performance of the introduced simplification. IndoorGML is one of OGC standards to provide a standard framework of semantic, topological, and geometric models for indoor spatial information [8]. The dataset for the experiment is compatible with CityGML LoD4 [9] so that data can be visualized via any CityGML viewer. Figures 9 and 10 show the visual and quantitative results of simplification of one large and complex corridor for varying $\tau$, given $\epsilon = \pi/36$, $\delta = \pi/180$, and $\gamma = \pi/7$. As $\tau$ increases, gradual simplification of shorter segments is observed, in particular at extremities, dominant features such as intrusions, extrusions, offsets, and corners are preserved.

While Figure 10 focuses on simplification results for one polygon, Figure 11 shows the comparison between the original LWM data, the indoor simplification, and RDP for all spaces on a floor. Note that a needle-shaped polygon will vanish after simplification if the length of its width is less than $\tau$.

References

[1] M. van de Kerkhof, M. Löffler I. Kostitsyna, M. Mirzanezhad, and C. Wenk. Global curve simplification. In 27th Annual European Symposium on Algorithms (ESA 2019), page 67:1–67:14, 2019.

IndoorGML data (core module) for Lotte World Mall (IndoorGML 1.0.3), http://www.indoorgml.net/resources/
Figure 10: Results of simplification for varying $\tau$

Figure 11: Comparison between the original data, the indoor simplification ($\tau=2$), and RDP (tolerance=1.2)

[2] David H Douglas and Thomas K Peucker. Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. *Cartographica: The International Journal for Geographic Information and Geovisualization*, 10:112–122, 1973.

[3] Urs Ramer. An iterative procedure for the polygonal approximation of plane curves. *Computer graphics and image processing*, 1(3):244–256, 1972.

[4] Robert Weibel and Geoffrey Dutton. Generalising spatial data and dealing with multiple representations. *Geographical information systems 1*, 1:125–155, 1999.

[5] Joon-Seok Kim and Ki-Joune Li. Simplification of geometric objects in an indoor space. *ISPRS journal of photogrammetry and remote sensing*, 147:146–162, 2019.

[6] Joon-Seok Kim, Tae-Hun Lee, and Ki-Joune Li. Prism geometry: Simple and efficient 3-d spatial model. In *The Proceedings of the 3rd International Workshop on 3D Geo-information*, pages 139–145, Seoul, South Korea, 2008.

[7] Wm Randolph Franklin and Harry R. Lewis. 3-D graphic display of discrete spatial data by prism maps. In *Proc. SIGGRAPH’78*, volume 12(3), pages 70–75, August 1978.

[8] Hae-Kyong Kang and Ki-Joune Li. A standard indoor spatial data model—ogcindoorgml and implementation approaches. *ISPRS International Journal of Geo-Information*, 6(4):116, 2017.

[9] Joon-Seok Kim, Sung-Jae Yoo, and Ki-Joune Li. Integrating indoorgml and citygml for indoor space. In *International Symposium on Web and Wireless Geographical Information Systems*, pages 184–196. Springer, 2014.