International Journal of Geo-Information

MDPI

Article

Multi-Instance Zero-Watermarking Algorithm for Vector Geographic Data

Qifei Zhou [1], Lin Yan [2,3], Zihao Wang [2,3], Na Ren [2,4,5,6,*] and Changqing Zhu [4,5,6]

[1] School of Geoscience and Technology, Zhengzhou University, Zhengzhou 450001, China; zhouqifei@zzu.edu.cn
[2] Hunan Engineering Research Center of Geographic Information Security and Application, Changsha 410017, China
[3] The Third Surveying and Mapping Institute of Hunan Province, Changsha 410018, China
[4] Key Laboratory of Virtual Geographic Environment (Nanjing Normal University), Ministry of Education, Nanjing 210023, China; 09322@njnu.edu.cn
[5] State Key Laboratory Cultivation Base of Geographical Environment Evolution (Jiangsu Province), Nanjing 210023, China
[6] Jiangsu Center for Collaborative Innovation in Geographical Information Resource Development and Application, Nanjing 210023, China
* Correspondence: 09359@njnu.edu.cn

Abstract: To address the variability and complexity of attack types, this paper proposes a multi-instance zero-watermarking algorithm that goes beyond the conventional one-to-one watermarking approach. Inspired by the class-instance paradigm in object-oriented programming, this algorithm constructs multiple zero watermarks from a single vector geographic dataset to enhance resilience against diverse attacks. Normalization is applied to eliminate dimensional and deformation inconsistencies, ensuring robustness against non-uniform scaling attacks. Feature triangle construction and angle selection are further utilized to provide resistance to interpolation and compression attacks. Moreover, angular features confer robustness against translation, uniform scaling, and rotation attacks. Experimental results demonstrate the superior robustness of the proposed algorithm, with normalized correlation values consistently maintaining 1.00 across various attack scenarios. Compared with existing methods, the algorithm exhibits superior comprehensive robustness, effectively safeguarding the copyright of vector geographic data.

Keywords: zero watermarking; robustness; vector geographic data; multi-instance zero watermarking; copyright protection

Academic Editors: Wolfgang Kainz, Jun Feng and Changqing Luo

Received: 04 December 2024
Revised: 20 January 2025
Accepted: 27 January 2025
Published: 30 January 2025

Citation: Zhou, Q.; Yan, L.; Wang, Z.; Ren, N.; Zhu, C. Multi-Instance Zero-Watermarking Algorithm for Vector Geographic Data. *ISPRS Int. J. Geo-Inf.* **2025**, *14*, 54. https://doi.org/10.3390/ijgi14020054

Copyright: © 2025 by the authors. Published by MDPI on behalf of the International Society for Photogrammetry and Remote Sensing. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

Vector geographic data are an essential resource in GIS-related industries and have found widespread applications in fields such as land surveying, military simulations, and urban planning. However, the conflict between data sharing and protection has become increasingly difficult to reconcile. Frequent copyright infringement incidents highlight the urgent need for effective protection measures [1–3]. Digital watermarking is a key technology for copyright protection. It establishes a strong relationship between digital data and watermark information, safeguarding the copyright of vector geographic data [4–6]. At the same time, attackers employ various methods to attack watermarks, making the improvement of watermark robustness a major research focus.

Current watermarking algorithms for vector geographic data can be broadly classified into embedded watermarking and constructive watermarking [7]. Embedded watermarking modifies the coordinates (including features derived from coordinates), the attribute, or the storage order to embed watermark information directly into the vector geographic data. For example, Peng et al. [8] used the Douglas–Peucker compression algorithm [9] to construct the feature vertex distance ratio (FVDR), embedding the watermark by adjusting the FVDR. Other approaches include embedding invisible characters into the attribute values of road data [10] or changing the storage order of vertices within polylines to embed the watermark information [11]. While embedded watermarking increases the relationship between the data and the watermark, it inevitably alters the data. Methods that embed watermarks into coordinates affect data precision, making them unsuitable for high data accuracy scenarios. Methods embedding watermarks into attributes increase file size and storage demands, potentially exposing the watermark to attackers. Furthermore, embedding watermarks into storage order may disrupt the semantic or functional meaning of the data, such as river flow directions or pipeline pathways. Therefore, the robustness of embedded watermarking is achieved at the cost of data integrity, limiting its application.

The second type is constructive watermarking, often referred to as zero watermarking [12–14]. Zero watermarking is distinguished by its non-intrusive nature, as it extracts features from the original data without introducing any modifications. The constructed watermark based on the features is then registered with third-party intellectual property rights (IPR) agencies for copyright protection. This type of method typically partitions data into sub-blocks using regular or irregular segmentation techniques, such as grids [15], rings [16], quadtrees [17], Delaunay triangulation [18], or object-based divisions [19–23]. Features such as geometric properties, vertex counts, or storage orders are quantified into watermark bits or indices, which are then combined to construct the zero watermark. For instance, Zhang et al. [15] employed a grid-based approach, partitioning the data into 64 × 64 blocks and subsequently quantifying the vertex count within each block to construct the watermark. Peng et al. [19] adopted an alternative strategy, treating individual polylines as the fundamental operational units. This approach, analogous to the embedded watermarking method described in the literature [8], utilized the FVDR as the basis for quantification, converting these features into watermark indices and bits. The constructed watermark was then refined through the majority voting mechanism. Similarly, Wang et al. [23] leveraged both the vertex count and the storage order features. Specifically, the vertex counts of arbitrary polyline or polygon pairs were quantified into watermark indices, while the storage order was used to generate watermark bits. Notably, such methods preserve the integrity of the original data by avoiding any modifications, thereby addressing the inherent limitations associated with embedded watermarking.

However, existing zero-watermarking methods, while advantageous in preserving data integrity, are constrained by their reliance on constructing a single watermark derived from the overall characteristics of the data. This scheme inherently limits their adaptability in resisting diverse and complex attack types. For instance, FVDR-based methods demonstrate robust performance against simplification attacks but remain vulnerable to non-uniform scaling. In contrast, vertex count-based methods excel in resisting non-uniform scaling and other geometric transformations but struggle under simplification and attacks that significantly modify vertex statistics. Consequently, these methods exhibit robustness in isolated scenarios but fail to achieve comprehensive resistance across multiple attack types.

In summary, embedded watermarking methods enhance robustness by embedding watermarks directly into the data, tightly coupling the watermark with the data. However, this approach inevitably compromises the original data's usability and precision,

limiting its applicability in scenarios demanding high data accuracy. Constructive watermarking, on the other hand, constructs watermarks from the data without modifying the data, effectively addressing the precision degradation issue of embedded methods. Nonetheless, existing constructive watermarking approaches still face challenges in achieving comprehensive robustness against diverse attack types. Therefore, developing algorithms capable of simultaneously resisting multiple attack types remains a critical research problem in the field of digital watermarking for vector geographic data.

To address these issues, this paper proposes a multi-instance zero-watermarking method for vector geographic data, inspired by the class-instance paradigm in object-oriented programming. By applying different preprocessing techniques, the proposed method constructs multiple zero watermarks from a single dataset, enabling it to adapt to the variability and complexity of attack types. Unlike existing approaches that segment the dataset to construct separate watermarks, the proposed algorithm constructs each zero watermark using the full set of data features, differentiated only by the preprocessing applied. The remainder of this paper is structured as follows: Section 2 introduces the methodology and its details, Section 3 describes the experimental setup, Section 4 presents the experimental results and analysis, Section 5 discusses the broader implications of the findings, and Section 6 concludes the study.

2. Proposed Approach

The proposed multi-instance zero-watermarking algorithm targets vector geographic data, specifically line data, as the watermarking carrier. Inspired by the class-instance paradigm in object-oriented programming, the algorithm treats the data as a "class" from which multiple "instances" (zero watermarks) are derived. By employing preprocessing techniques such as normalization, feature triangle construction, and angle selection, the algorithm generates multiple zero watermarks to resist various attack types. Normalization eliminates geometric deformations, enhancing robustness against non-uniform scaling attacks. Feature triangle construction and angle selection, inspired by the Douglas-Peucker algorithm, provide resilience against interpolation and simplification attacks. Furthermore, angle-based features are derived by quantizing the selected angles in the feature triangle. Previous studies [24–26] have demonstrated that these features exhibit robustness against translation, uniform scaling, and rotation. Their incorporation into the proposed method enhances its resistance to these geometric attacks. To address non-uniform scaling, interpolation, and simplification attacks, two zero watermarks are constructed. The main framework of the proposed algorithm is shown in Figure 1.

As illustrated in Figure 1, the algorithm begins by extracting polylines from the line data. The overall process can be divided into three main parts: collinearity checks, the construction of the first zero watermark (for resisting interpolation and simplification attacks), and the construction of the second zero watermark (for resisting non-uniform scaling attacks). In the first part, for each polyline, collinearity checks are performed on its vertices. If all vertices of a polyline are collinear, it is skipped, and the next polyline is processed. Only non-collinear polylines are further proceeded to subsequent stages.

The second and third parts involve the generation of two zero watermarks. The primary difference between these two processes lies in the application of normalization. Specifically, normalization is applied in the construction of the second zero watermark to enhance robustness against non-uniform scaling attacks. However, normalization is avoided during the construction of the first zero watermark for robustness against interpolation and simplification attacks. This is because simplification may alter the bounding coordinates and lead to unintended deformations during normalization. Beyond normalization, the subsequent steps for generating both zero watermarks are identical, as summarized below: (1) Feature triangle construction: The polyline is reduced to its most essential form

by retaining only three vertices, which form the feature triangle; (2) Angle selection: Two angles are selected from the three angles in the constructed feature triangle; (3) Watermark construction: The selected angles are quantified into watermark indices and bits, which are aggregated using the majority voting mechanism to construct two zero watermarks.

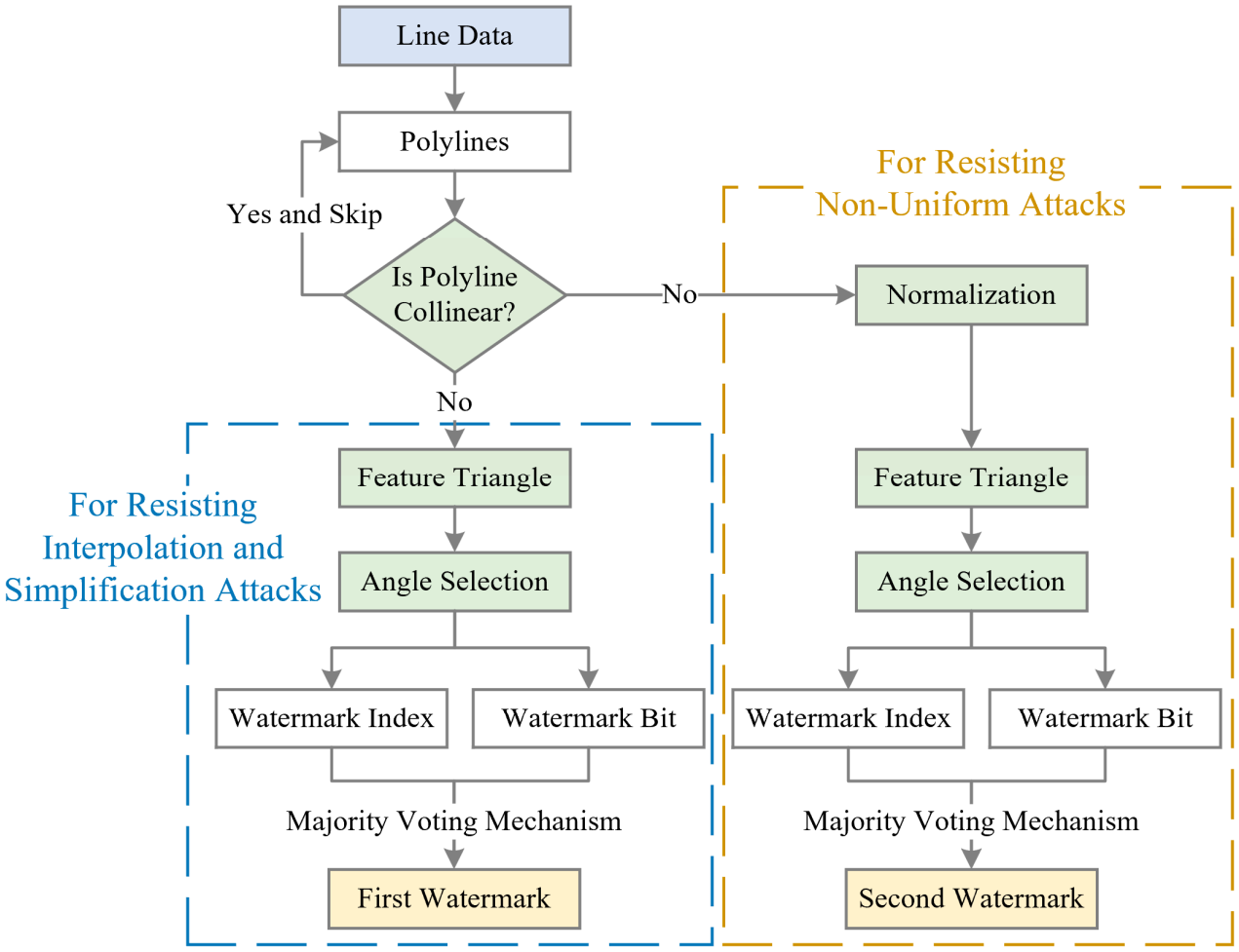


Figure 1. Main framework of the proposed method.

During copyright verification, the generated zero watermarks are evaluated for consistency. The steps of collinearity check, normalization, feature triangle construction, angle selection, zero-watermark construction, and watermark evaluation are detailed in the following subsections.

2.1. Collinearity Check

To ensure meaningful geometric features for watermark construction, the algorithm excludes polylines with collinear vertices. This process involves determining whether three vertices in a polyline are collinear based on their geometric properties. The collinearity check is divided into two cases: (1) For polylines with only two vertices, collinearity is inherent, and such polylines are skipped. (2) For polylines with more than two vertices, three vertices are iteratively selected from the vertex set to evaluate collinearity. This process is repeated C_n^3 times, where n is the total number of vertices in a polyline.

The collinearity of three vertices (x_1, y_1), (x_2, y_2), and (x_3, y_3) is determined using the area method, defined as follows:

$$\text{Area} = \frac{1}{2}|x_1(y_2 - y_3) + x_2(y_3 - y_1) + x_3(y_1 - y_2)| \quad (1)$$

If the calculated area is less than a predefined threshold (1×10^{-6}), the vertices are considered collinear. The threshold is introduced to address the limitations of floating-point arithmetic, where numerical precision makes exact equality comparisons infeasible. Theoretically, collinearity requires the area formed by three vertices to be zero, but in practice, floating-point calculations may produce small non-zero values for collinear points. Thus, the threshold serves as a tolerance value to ensure robust and accurate detection of collinearity. Any polyline that does not contain at least one set of non-collinear vertices is excluded from subsequent steps. This ensures that all polylines used in watermark construction contribute to meaningful geometric features, improving the reliability of the resulting zero watermarks.

2.2. Normalization

Normalization is employed as a preprocessing step to eliminate deformation caused by non-uniform scaling. This ensures that the geometric features extracted from the data remain consistent, regardless of the scaling factors applied to the original coordinates. The process involves normalizing the x and y coordinates of the vertices using min–max normalization, as defined by the following equations:

$$\begin{cases} x_{\text{norm}} = \dfrac{x - \min(x)}{\max(x) - \min(x)} \\ y_{\text{norm}} = \dfrac{y - \min(y)}{\max(y) - \min(y)} \end{cases} \tag{2}$$

Here, x_{norm} and y_{norm} represent the normalized coordinates, while $\min(x)$, $\max(x)$, $\min(y)$, and $\max(y)$ denote the minimum and maximum values of the x and y coordinates within the polyline, respectively.

The stability of the normalized coordinates under non-uniform scaling has been validated theoretically. Zhang et al. [27,28] demonstrated that the normalized x and y coordinates remain invariant when the original coordinates are scaled by factors Sx and Sy. Specifically, after applying scaling, the normalized coordinates are recalculated as follows:

$$\begin{cases} x_{\text{norm}} = \dfrac{\text{Sx} \cdot x - \min(\text{Sx} \cdot x)}{\max(\text{Sx} \cdot x) - \min(\text{Sx} \cdot x)} = \dfrac{x - \min(x)}{\max(x) - \min(x)} \\ y_{\text{norm}} = \dfrac{\text{Sy} \cdot y - min(\text{Sy} \cdot y)}{\max(\text{Sy} \cdot y) - \min(\text{Sy} \cdot y)} = \dfrac{y - \min(y)}{\max(y) - \min(y)} \end{cases} \tag{3}$$

This invariance ensures that the geometric features extracted from the normalized coordinates remain robust against non-uniform scaling, a critical requirement for practical watermark construction.

Normalization is applied selectively. It is used to stabilize the geometry shape when addressing non-uniform scaling attacks. However, it is omitted when addressing interpolation and simplification attacks, as these attacks may alter the bounding coordinates, potentially introducing unintended deformations during normalization. This selective normalization provides the potential overall robustness to different attack scenarios.

2.3. Feature Triangle Construction

Feature triangle construction is a core step in the proposed algorithm, designed to extract robust geometric features from polylines. The method begins by connecting the first vertex A and the last vertex B of the polyline to form a line segment AB. Then, the algorithm iteratively examines the remaining vertices to identify point C, which is farthest from AB. The distance is calculated as the perpendicular distance from each vertex to the line segment. The three vertices, A, B, and C, form the feature triangle, encapsulating

critical geometric properties of the polyline. Figure 2 provides a demonstration of feature triangle construction.

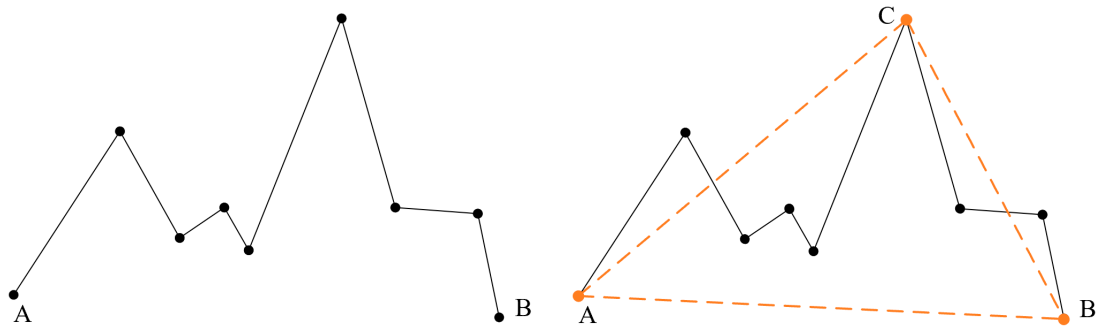


Figure 2. Demonstration of feature triangle construction: (**a**) a polyline with nine vertices and (**b**) the feature triangle ABC.

As illustrated in Figure 2, a polyline consisting of nine vertices is shown in Figure 2a, where A and B represent the first and last points, respectively. The vertex C, identified as the farthest point from AB, completes the feature triangle, as depicted in Figure 2b. This construction method corresponds to the initial step of the Douglas–Peucker compression algorithm. However, unlike the Douglas–Peucker algorithm, which recursively divides the polyline to find additional feature points, the proposed method selects only the farthest point. This is not a simplification of the process but a deliberate choice to improve robustness against compression attacks. As noted in the literature [13], when fewer feature points are retained, the watermarking scheme exhibits greater resistance to compression, as excessive simplification would be required to disrupt the constructed triangle.

2.4. Angle Selection and Zero-Watermark Construction

From the feature triangle, two angles, $\angle ACB$ and $\angle CAB$, are selected and denoted as α and β, respectively. These angles are calculated using the cosine rule as follows:

$$\begin{cases} \alpha = \arccos\left(\dfrac{AB^2 + BC^2 - AC^2}{2 \cdot AB \cdot BC}\right) \\ \beta = \arccos\left(\dfrac{AB^2 + AC^2 - BC^2}{2 \cdot AB \cdot AC}\right) \end{cases} \tag{4}$$

where AB, BC, and AC are the side lengths of the feature triangle, calculated from the coordinates of the vertices A, B, and C. The angles α and β are measured in radians, ensuring their range lies within $(0, \pi)$. As ensured by the collinearity check in Section 2.1, the feature triangle is always valid with non-collinear vertices.

Next, α is used for quantifying the watermark index. The quantization process involves retaining all digits from the start of α up to and including the q-th digit after the decimal point:

$$\alpha' = \mathrm{round}(\alpha \cdot 10^q) \tag{5}$$

where the round( ) function rounds the value to the nearest integer. The resulting α' is then passed as a seed to a uniformly distributed pseudorandom integer generator, denoted as rand. The watermark index, denoted as WI, is computed as follows:

$$\mathrm{WI} = \mathrm{rand}(\alpha', 1, \mathrm{N}) \tag{6}$$

where N is the watermark length. The last two arguments of the rand( ) function represent the minimum and maximum values of the discrete uniform distribution, defining the

range of the watermark index as [1, N]. Similarly, the angle β is processed to generate the watermark bit. After extracting the integer β' in the same manner as α', the watermark bit, denoted as WB, is calculated as follows:

$$\mathrm{WB} = \mathrm{rand}(\beta', 0, 1) \tag{7}$$

Here, the watermark bit takes a value of either 0 or 1.

Since a line dataset typically contains multiple polylines, multiple watermark indices and bits are generated. These are aggregated using a majority voting mechanism, as described in the literature [8,21]. Specifically, an array Stat, equal in length to the watermark, is initialized with all elements set to 0. For each watermark index and its corresponding watermark bit, the following update rules are applied:

$$\mathrm{Stat}_{\mathrm{WI}_i} = \begin{cases} \mathrm{Stat}_{\mathrm{WI}_i} - 1, & \text{if } \mathrm{WB}_i = 0 \\ \mathrm{Stat}_{\mathrm{WI}_i} + 1, & \text{if } \mathrm{WB}_i = 1 \end{cases} \tag{8}$$

where WI_i refers to the specific index in the watermark corresponding to *i*-th polyline, and WB_i represents the corresponding bit. After processing all indices and bits, the binary watermark W is finalized based on the values in Stat:

$$\mathrm{W}_{\mathrm{WI}_i} = \begin{cases} 0, & \text{if } \mathrm{Stat}_{\mathrm{WI}_i} \le 0 \\ 1, & \text{if } \mathrm{Stat}_{\mathrm{WI}_i} > 0 \end{cases} \tag{9}$$

The resulting W is a binary sequence composed of 0 s and 1 s, completing the construction of the zero watermark.

2.5. Watermark Evaluation

The proposed algorithm constructs two zero watermarks from the original dataset, which are registered with an authoritative third-party IPR agency. In the event of a copyright dispute, two zero watermarks are also constructed from the suspicious dataset for comparison. The normalized correlation (NC) is introduced as a quantitative metric to evaluate the similarity between the watermarks, calculated using the following formula:

$$\mathrm{NC} = \frac{\sum_{i=1}^{\mathrm{N}} \mathrm{W}_i \, \mathrm{W}_i'}{\sqrt{\sum_{i=1}^{\mathrm{N}} \mathrm{W}_i^2} \sqrt{\sum_{i=1}^{\mathrm{N}} \mathrm{W}_i'^2}} \tag{10}$$

where W and W′ represent the binary sequences of the registered and reconstructed watermarks, respectively. The NC value lies within the range of (0, 1], where a value closer to 1 indicates greater similarity.

To determine whether the watermarks match, a predefined NC threshold is introduced. The two watermarks are considered identical if the NC value exceeds the threshold. The suspicious dataset's two zero watermarks are compared against the two registered zero watermarks, resulting in four NC values theoretically. In practice, a fixed one-to-one mapping is enforced, whereby the first constructed watermark is aligned with the first registered watermark, and the second constructed watermark is aligned with the second registered watermark. This mapping reduces the comparison to two NC values, denoted as NC1 and NC2, which quantify the similarity between each pair of corresponding watermarks. The final detection result is determined by selecting the maximum value between NC1 and NC2. This approach ensures that even if one instance of the watermark is affected by an attack, the other instance can still provide a reliable match. The redundancy introduced by the multi-instance scheme enhances the robustness of the algorithm, allowing it to handle diverse attack scenarios. Therefore, this evaluation process ensures that the algorithm effectively quantifies the similarity between watermarks, providing a robust copyright validation and infringement detection mechanism.

3. Experiments

3.1. Datasets

The experiments utilize a vector geographic dataset representing waterways, as shown in Figure 3. This dataset comprises line data from regions located at the intersection of Hunan, Jiangxi, and Guangdong provinces in China. The coordinate reference system is CGCS2000, with a 3-degree Gauss–Kruger projection and a central meridian at 114°E. The dataset contains 1540 polylines, with an average of 56 vertices per polyline. The maximum number of vertices in a single polyline is 550, while the minimum is 2, and the median is 45, making it suitable for evaluating the proposed algorithm's robustness and efficiency.

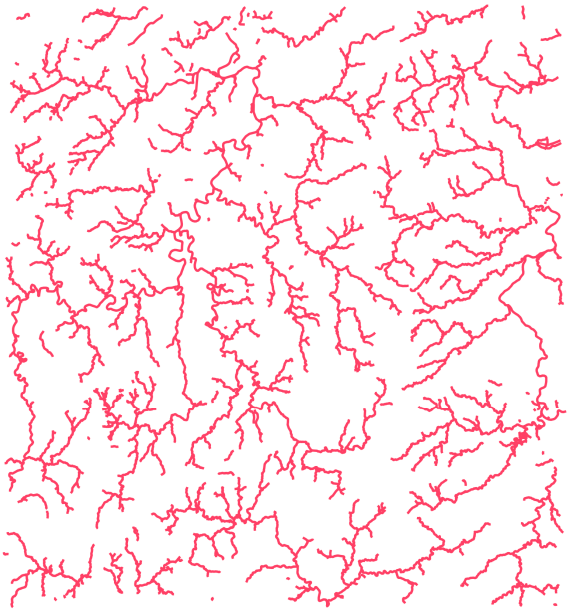


Figure 3. Waterways dataset used in the experiments.

3.2. Parameter Settings and Comparison Algorithms

The proposed algorithm is configured with key parameters to balance robustness and computational efficiency. The angle quantization precision q is set to 4, meaning the quantization process retains all digits from the start of the number up to and including the fourth digit after the decimal point. The watermark length N is chosen as 64, providing a suitable trade-off between performance and computational cost. An NC threshold of 0.75 is adopted, referencing values commonly used in prior literature [7,21,29,30]. These settings are empirically determined to provide reliable performance under various attack scenarios.

For comparison algorithms, two zero-watermarking methods are selected and denoted as the method Peng [19] and the method Wang [23]. As mentioned in Section 1, partitioning data into sub-blocks is a common strategy in zero-watermarking algorithms. Among these sub-blocking methods, the object-based division is particularly stable because data modifications are typically performed at the object level. Peng and Wang adopted the object-based division method, consistent with the approach in the proposed algorithm. The method Peng extracts the FVDR from polylines, while the method Wang uses vertex count and storage order features. Both of them are configured with the same watermark length as the proposed method to ensure consistency. Additionally, the method Peng employs the Douglas–Peucker algorithm for feature point extraction, with a relative threshold of 0.3. All algorithms are evaluated using the NC metric to provide a fair basis for comparison.

3.3. Attack Design

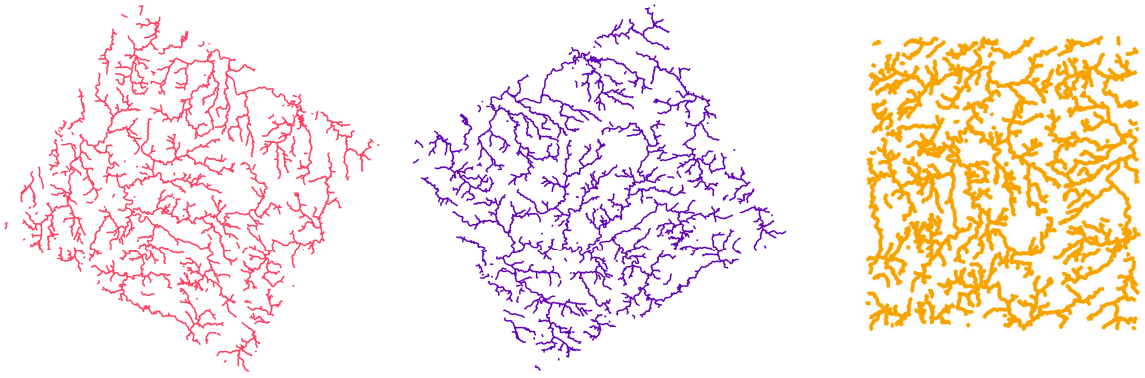
To comprehensively evaluate the robustness of the proposed algorithm, six types of attacks commonly encountered in vector geographic data processing are applied: rotation, uniform scaling, non-uniform scaling, translation, interpolation, and simplification. Each attack is configured with parameters referenced from the literature [21] to simulate real-world data distortions. Table 1 summarizes the configurations of these attacks.

Table 1. Attack configurations.

Attack Type	Parameter	Value
Rotation	Rotation angle θ	60° to 360° (in 60° increments)
Uniform scaling	Scaling factors (Sx = Sy)	0.4, 0.6, 0.8, 1, 2, 3, 4
Non-uniform scaling	Scaling factors (Sx ≠ Sy)	Sx = 0.4, 0.6, 0.8, 1, 2, 3, 4; Sy = 1
Translation	Translation distance *t*	10 m to 60 m (in 10 m increments)
Interpolation	Tolerance	10 m to 60 m (in 10 m increments)
Simplification	Tolerance	10 m to 60 m (in 10 m increments)

In Table 1, most attacks, such as rotation, scaling, and translation, are standard transformations applied to the entire dataset without altering the density of vertices. However, interpolation and simplification deserve special attention, as they directly impact the dataset's vertex density. New vertices are inserted into segments exceeding the specified tolerance for interpolation, increasing the data's density. In contrast, simplification reduces vertex count by removing coordinates while preserving the overall shape, with larger tolerances resulting in more aggressive vertex removal. These complementary attack types can provide valuable insights into the robustness of watermarking algorithms.

Figure 4 provides a visual demonstration of the attack effects on the dataset. As shown in Figure 4a–c,f, rotation, uniform scaling, and translation attacks do not alter the shape of the data. Conversely, non-uniform scaling attacks lead to visible distortions in the data shape, as illustrated in Figure 4d,e. For instance, when Sx = 0.4, the data appear significantly narrower in the horizontal direction than when Sx = 0.8. To facilitate the visualization of the attack effects, Figures 4g,h overlay the attacked data onto the original data. Interpolation attacks increase the number of vertices in the data, as shown in Figure 4g, where the interpolation tolerance of 10 m adds new points to segments shorter than the specified distance. In contrast, simplification attacks reduce the number of vertices by removing coordinates, resulting in sharper geometries, as depicted in Figure 4h.



(**a**) (**b**) (**c**)