Topology-Preserving Watermarking of Vector Data

Stefan Huber* Martin Held* Roland Kwitt† Peter Meerwald*

Abstract

The embedding of a digital watermark in vector data results in a perturbation of the vertices which needs to be constrained in order to maintain geometric properties of the data. In this paper we investigate the problem of computing so-called perturbation regions in which the vertices of a planar straight-line graph may be dislocated while still preserving the topology of the input. We propose two different algorithms to solve this problem and discuss how they can form the geometric part of a watermarking framework.

1 Introduction

A standard way for watermarking digital data is to embed imperceptible, yet detectable, information into a digital signal. Since the watermark is only known to the embedder, copyright holders can use this technology to mark their content in order to prove ownership by being able to detect the embedded signal.

Most watermarking research has been directed towards techniques applicable to raster data and audio content. However, complex vector data in computer-aided design as well as maps and infrastructure data in geographic information systems, for instance, constitute equally valuable digital assets. When watermarking vector data statistical features are imposed by dislocating vertices. Hence, novel geometric requirements are introduced, while still demanding imperceptibility of the watermark and robustness against different kinds of attacks. For example, one needs to guarantee that watermarking does not introduce overlaps among rivers and streets in a map.

Surprisingly, copyright protection of vector data with distortion constraints has not received much attention. Doncel et al. [2] consider multiple polygonal chains sharing common vertices, such as the border of neighboring countries, and discuss how to preserve connectivity after watermarking. In [6, 7], methods are proposed that avoid perturbing certain vertices to preserve the visual appearance of the original data.

In [4] we presented a watermarking framework for planar straight-line graphs which consists of a geometric part, a watermarking part and a final correction step, cf. Fig. 1. In this paper we focus on the preservation of the input topology, i.e., on ensuring that

[T1] the numbers of vertices and edges
[T2] all containment relations
[T3] all incidence orders at vertices

remain unchanged, and that

[T4] no intersections are introduced.

We discuss two algorithms for computing a so-called *maximum perturbation region* (MPR) for every vertex such that T1–T4 can be guaranteed: If all vertices stay within their MPRs after watermarking then the input topology is guaranteed to be preserved. Once the MPRs are known, we embed the watermark into the input data. Since this process need not respect T2–T4, in a final correction step we use the MPRs to correct the output of the watermarking step in order to guarantee T1–T4. While the basic idea for preserving the input topology is simple, the computational challenge is to efficiently compute MPRs that are large enough such that the correction step does not destroy the watermark embedding. Our approach is general enough to work with any watermarking method that does not insert or remove vertices or edges of the input.

2 Maximum perturbation regions

Consider a planar straight-line graph $G = (V, E)$ with vertex set $V := \{v_1, \ldots, v_n\}$ and edge set E. The process of embedding a watermark in G can be interpreted as a transformation of the vertex set V to a *perturbed* vertex set $V' = \{v'_1, \ldots, v'_n\}$. We denote the graph given by the perturbed vertex set by $G' = (V', E')$ and by E' the corresponding edge set of G'. We seek *maximum perturbation regions* (MPRs) as regions $R_1, \ldots, R_n$, with $v_i \in R_i \subset \mathbb{R}^2$, such that the following property holds: If $v'_i \in R_i$ for all $1 \leq i \leq n$ then T1–T4 hold for G'. Thus, $R_1, \ldots, R_n$ are regions in which it is safe to perform perturbations without violating the input topology.

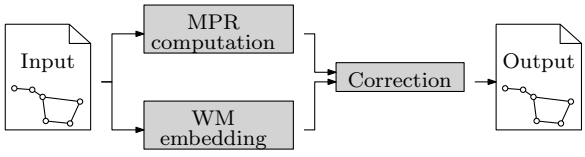

Figure 1: Our watermarking framework.

*Universität Salzburg, FB Computerwissenschaften, A-5020 Salzburg, {shuber,held,pmeerw}@cosy.sbg.ac.at. Work was supported by Austrian FWF Grant L367-N15.

†Kitware Inc., NC, USA, {roland.kwitt@kitware.com}

2.1 Computing MPRs using Voronoi diagrams

For the Voronoi-based computation of MPRs we consider the vertices of V and the straight-line segments of E as the set S of input sites. The Voronoi diagram $\mathcal{VD}(S)$ of S decomposes the plane into Voronoi cells $\mathcal{VC}(s, S)$, with $s \in S$. Our MPR algorithm relies on the following observation: If the perturbed counterpart $e' \in E'$ of an edge $e \in E$ does not intersect the Voronoi cells of edges non-adjacent to e, then G' is planar as the Voronoi cells of different sites do not overlap.

We denote by $|e|$ the length of the line segment e and define for any $r > 0$ and any line segment e the set $B(e, r) \subset \mathbb{R}^2$ as the rectangle with width $2r$ and length $|e|$ that has e as its center-line. Next, we denote by $D_v(r)$ the open disk with center v and radius r. Our MPR-algorithm consists of two phases:

Phase 1: For each vertex $v_i \in V$, we consider the incident edges $e^i_1, \ldots, e^i_{d_i}$ and we denote by $\hat{e}^i_j$ the half of e^i_j that is incident to v_i. Then we compute for each vertex $v_i \in V$ the largest $t_i \leq \min_{1 \leq j \leq d_i} |\hat{e}^i_j|$ such that

$$D_{v_i}(t_i) \cup \bigcup_{j=1}^{d_i} B(\hat{e}^i_j, t_i) \subseteq \mathcal{VC}(v_i, S) \cup \bigcup_{j=1}^{d_i} \mathcal{VC}(e^i_j, S). \quad (1)$$

Let $T(v_i) := D_{v_i}(t_i) \cup \bigcup_{j=1}^{d_i} B(\hat{e}^i_j, t_i)$, cf. Fig. 2. In order to compute $t_1, \ldots, t_n$ we first cut the Voronoi cell $\mathcal{VC}(e, S)$ of every edge $e \in E$, with $e = v_i v_j$, into two halves along the bisector of v_i and v_j and insert the two points of intersection as Voronoi nodes. Every parabolic Voronoi edge is also split by a node at its apex. Then we traverse all halved Voronoi cells and compute the (non-zero) distances of the nodes of the halved Voronoi cells to their input sites.

Lemma 1 *The interiors of $T(v_i)$ and $T(v_j)$ do not overlap for different v_i and v_j.*

Phase 2: For every vertex v_i we consider its adjacent vertices $v^i_1, \ldots, v^i_{d_i}$ and compute the value

$$r_i := \min\{t_{v_i}, t_{v^i_1}, \ldots, t_{v^i_{d_i}}\}. \quad (2)$$

Then we define the maximum perturbation region R_i of the vertex v_i as

$$R_i := D_{v_i}(r_i). \quad (3)$$

In Fig. 3, we illustrate all MPRs as dashed circles. For an edge $e \in E$ we call the area that is bounded by the MPRs of the incident vertices v_i, v_j of e and their bitangents the *hose* H_e around e. In fact, H_e represents the valid area in which the perturbed edge $v'_i v'_j$ may lie. All hoses are shown as shaded areas in Fig. 3. In order to avoid hoses that touch we can simply multiply all radii r_i by $1-\epsilon$, for a small positive constant $\epsilon < 1$.

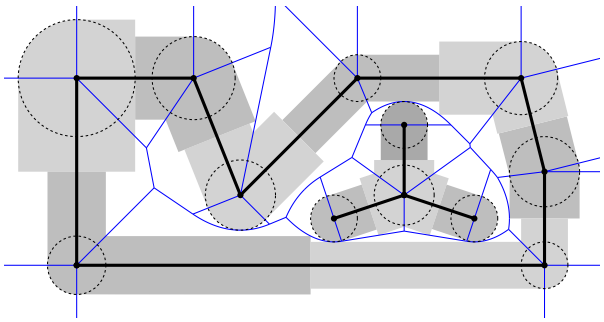

Figure 2: Phase 1: The regions $T(v_i)$ for vertices v_i are shaded in gray levels. The circles $D_{v_i}(t_i)$ are shown dashed.

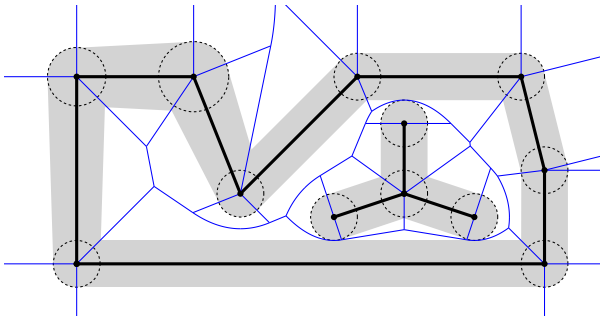

Figure 3: Phase 2: The MPRs are shown dashed. The union $\bigcup_{e \in E} H(e)$ of all hoses is shaded.

Lemma 2 *A hose $H(e)$ of an edge $e = v_i v_j$ is contained in $T(v_i) \cup T(v_j)$.*

Corollary 3 *$H(e_i)$ and $H(e_j)$ do not overlap if e_i and e_j have no common vertex.*

Theorem 4 *If the perturbation v'_i of the vertex $v_i \in V$ is constrained to R_i as defined in (3), for $1 \leq i \leq n$, then T1–T4 are guaranteed for G'.*

Proof. Since no vertices or edges are added or removed, T1 is met. The hoses of one connected component of G are separated from all other hoses by the Voronoi diagram, thus enforcing T2. Similarly, due to the Voronoi diagram, the cyclic order of the hoses of edges incident upon a vertex is identical to the order of those edges and, therefore, T3 is guaranteed. Finally, Cor. 3 ensures T4. □

The Voronoi-based approach assigns the MPRs in a *fair* manner in the following sense: If two hoses touch each other then they touch at the apex of a parabolic Voronoi edge. Hence, both hoses are equally wide.

This approach does not necessarily determine the largest possible MPRs, though: Phase 2 is easy to implement but rather conservative. (For example, the MPR of the vertex in the upper left-hand corner of Fig. 3 could be chosen larger.) In general, we could increase individual MPRs in an additional third phase

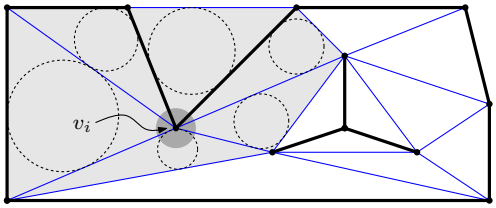

Figure 4: The triangulation-based approach: R_i is defined by triangles incident to v_i.

as long as Lemma 2 remains valid. We did not study this strategy in more detail, though.

We can compute all $t_1, \ldots, t_n$ in $O(n)$ time as the Voronoi diagram $\mathcal{VD}(S)$ is of linear size. The computation of the Voronoi diagram itself takes $O(n \log n)$, which leads to a total runtime of $O(n \log n)$ for the Voronoi-based MPR-algorithm. We implemented this approach in C++, based on the Voronoi package VRONI [3] to compute the Voronoi diagram. Computing all MPRs on a dataset with 60 000 vertices takes about a second on a mid-range personal computer.

The Voronoi-based approach also permits generalisations to input elements more general than straight-line segments. For example, one could consider circular arcs, which can also be processed by VRONI.

2.2 Computing MPRs using triangulations

Our second approach to compute MPRs is based on triangulations. The algorithm starts with a constrained triangulation T of the convex hull of G, with E forming the line segments of the constraints. Hence, G can be interpreted as a subgraph of T, see Fig. 4.

Let us denote by T' the graph that results from T by replacing V by V'. The key observation in the triangulation-based approach is the following: If dislocating the vertices violates T2–T4 then at least one triangle in the triangulation has changed its orientation. Hence, the objective is to restrict the perturbations of the vertices to MPRs such that no orientation of a triangle changes.

Consider a triangle Δ of T and denote by $I(\Delta)$ the radius of its incircle. If the vertices of Δ are dislocated by a distance of less than $I(\Delta)$, then the orientation of Δ remains the same. Hence, we compute the MPR R_i of the vertex $v_i \in V$ as

$$R_i := D_{v_i}(r_i), \tag{4}$$

where $r_i := \min_{1 \le j \le D_i} I(\Delta^i_j)$, with $\Delta^i_1, \ldots, \Delta^i_{D_i}$ denoting all triangles incident to v_i. Since for each triangle Δ in T it holds that the MPRs of its vertices are disks with radii at most $I(\Delta)$, we get that all triangles in T preserve their orientations.

Theorem 5 *If the perturbation v'_i of the vertex $v_i \in V$ is constrained to R_i as defined in (4), for $1 \le i \le n$, then T1–T4 are guaranteed for G'.*

Proof. Consider the Voronoi diagram of T: all Voronoi nodes coincide with the centers of the incircles of the triangles of T, and every Voronoi cell of an interior edge $e = v_iv_j$ is given by two triangles formed by v_i, v_j and one of the incircle centers of the two triangles of T that have e as an edge. Hence, the triangulation-based approach can be interpreted as a slightly modified variant of the Voronoi-based approach applied to the edges of T, allowing us to confer the correctness result of Thm. 4. □

We note that this algorithm works with any constrained triangulation T of G. However, in order to permit perturbations to robustly embed the watermark, we seek MPRs that are as large as possible. Hence, the question arises which triangulations of G have large incircles and subsequently large MPRs. Obviously, among all triangles with fixed circumcircle, the equilateral triangle has the largest incircle. This observation motivated us to employ the (constrained) Delaunay triangulation for our actual implementation of the above algorithm.

Guaranteed-quality triangulations that use Steiner vertices in order to guarantee a lower bound for the smallest angle in a triangulation are known, see, e.g., [8]. However, maximizing the smallest incircle and maximizing the smallest inner angle of a triangle are not the same objectives. For example, a skinny triangle may be fine for us if it is just large enough and, in further consequence, contains still a large incircle.

To the best of our knowledge, maximizing the smallest incircle did not attract attention so far. Note that successively adding Steiner vertices may increase the smallest angle, but, at some point, certainly does not enlarge the smallest incircle any further.

We implemented the following heuristic approach in order to enlarge the incircles: After computing the triangulation of G, we determine a list of triangles Δ, where $I(\Delta)$ is by a factor of $f > 1$ larger when compared to its three neighboring triangles and insert Steiner points at the centers of their incircles. In practice, we observed that using $f = 1.5$ increases the average incircle radius by a few percent. However, applying this refinement step multiple times did not lead to a further improvement in our experiments.

The constrained Delaunay triangulation can be computed in $O(n \log n)$ time. Computing $r_1, \ldots, r_n$ can be done in $O(n)$ time as a triangulation contains $O(n)$ triangles. Hence, we end up with a total time complexity of $O(n \log n)$. For our C++ implementation of the triangulation-based algorithm we used the GNU Triangulated Surface (GTS) library [5], which implements a semi-dynamic algorithm. Computing all MPRs on a dataset with 60 000 vertices and running one refinement step with $f = 1.5$ takes about a second on a mid-range computer.

The triangulation-based method can also be ex-

tended to $\mathbb{R}^3$, if we want to embed a watermark on a polyhedron P with n vertices. Chazelle and Palios [**?**] showed that there exists a tetrahedralization of P that employs up to $O(n + r^2)$ Steiner vertices, where r denotes the number of reflex edges of P, and that it can be computed in $O(nr + r^2 \log r)$ time. The radius of the MPR of a vertex v of P is then defined as the minimum of the radii among the inscribed spheres of the tetrahedra incident to v.

3 Application to watermarking

We conclude with an outline of a watermarking scheme that was proposed in the context of vector data, cf. [2, 4]. The common basis of many watermarking approaches is to transform the input in a way such that conventional strategies can be applied. For this purpose, we regard all vertices V of G as points in the complex plane $\mathbb{C}$. To foster embedding a long, and consequently more robust, watermark sequence we only consider chains of G of a certain length as our input data and then perform a Discete Fourier Transform (DFT). This gives us a frequency-domain representation of the chains in the form of a set of Fourier coefficients. A scaled bipolar {+1,-1} random sequence (generated using the secret seed K belonging to the copyright holder) is then added to the DFT magnitudes and the inverse DFT is computed to finally reclaim the points in $\mathbb{C}$ and, consequently, the perturbed vertex set V'.

If all vertices perturbed by the watermarking stay within their MPRs then nothing needs to be done. Otherwise, if a vertex is outside of its pre-computed MPR, we can choose between two correction methods which both aim at preserving most of the watermark signal: (i) project each vertex onto the boundary of its corresponding MPR disk, and (ii) only perform the correction for the vertices of edges that actually intersect. Method (i) runs in $O(n)$ time, while the conditional correction (ii) can be done in $O(kn)$ time, where $k \in O(n)$ denotes the number of edges of G which have at least one vertex not in its MPR. (Very recent results suggest that $O(n \log n + m)$ time is achievable for (ii), where m denotes the number of intersections among $E \cup E'$.)

This trade-off between time and strength of the watermark needs to be evaluated in the context of the desired application. Imposing the topology constraints T2–T4 on the watermarked data dampens the watermark. To get an impression of detection performance for both strategies, we embedded a watermark in a vector graphic of roughly 24000 vertices and only considered chains of length > 200. With a modest watermark embedding strength, this led to ≈ 1600 vertices subject to MPR correction. Using correction strategy (i), the probability of missing the watermark is $\approx 10^{-20}$, while strategy (ii) yields $\approx 10^{-60}$.

4 Discussion

Our experiments with both MPR computations allow the following conclusions: First, using a Voronoi tessellation yields larger MPR regions and consequently gives more freedom for the watermarking process. Further, it can be extended to more general input, such as circular arcs for instance. Using triangulations, on the other hand, leads to potentially smaller MPRs and less freedom for watermarking. However, in consideration of a potential 3-D extension, one might favor this approach due to the simpler implementation.

Our work also reveals two interesting problems for future research: How can we compute a triangulation such that the radii of the incircles are maximized? And, more generally, how can me make the watermarking respect other important geometric properties of vector data, such as right angles or parallelism?

References

[1] B. Chazelle and L. Palios. Triangulating a Non-Convex Polytope. In *Proc. 5th Annu. ACM Sympos. Comput. Geom.*, pages 393–400, Saarbrücken, Germany, Sept. 1989.

[2] V. Doncel, N. Nikolaidis, and I. Pitas. An Optimal Detector Structure for the Fourier Descriptor Domain Watermarking of 2D Vector Graphics. *IEEE Trans. Visualizat. Comput. Graph.*, 13(5):851–863, Sept. 2007.

[3] M. Held and S. Huber. Topology-Oriented Incremental Computation of Voronoi Diagrams of Circular Arcs and Straight-Line Segments. *Comput. Aided Design*, 41(5):327–338, May 2009.

[4] S. Huber, R. Kwitt, P. Meerwald, M. Held, and A. Uhl. Watermarking of 2D Vector Graphics with Distortion Constraint. In *IEEE Int. Conf. on Multimedia & Expo (ICME 2010)*, pages 480–485, Singapore, July 2010.

[5] S. Popinet. GTS – The GNU Triangulated Surface Library. Online available from `http://gts.sourceforge.net/`.

[6] Y.-C. Pu, W.-C. Du, and I.-C. Jou. Perceptually Transparent Polyline Watermarking Based on Normal Multi-Resolution Representation. *IEICE Trans. Information and Systems*, E89-D(12):2939–2949, Digital Press 2006.

[7] C. Y. Shao, H. L. Wang, X. M. Niu, and X. T. Wang. Shape-Preserving Algorithm for Watermarking 2-D Vector Map Data. In *Proc. 7th IEEE Workshop on Multimedia Signal Proc.*, pages 1–4, Shanghai, China, Oct. 2005.

[8] J. Shewchuk. Delaunay Refinement Algorithms for Triangular Mesh Generation. *Comput. Geom. Theory and Appl.*, 22(1-3):21–74, May 2002.