# Tracing Compressed Curves in Triangulated Surfaces*

Jeff Erickson† Amir Nayyeri
University of Illinois, Urbana-Champaign
{jeffe,nayyeri2}@illinois.edu

Submitted to SOCG 2012 — December 2, 2011

**Abstract**

A simple path or cycle in a triangulated surface is *normal* if it intersects any triangle in a finite set of arcs, each crossing from one edge of the triangle to another. We describe an algorithm to "trace" a normal curve in $O(\min\{X, n^2 \log X\})$ time, where $n$ is the complexity of the surface triangulation and $X$ is the number of times the curve crosses edges of the triangulation. In particular, our algorithm runs in polynomial time even when the number of crossings is exponential in $n$. Our tracing algorithm computes a new cellular decomposition of the surface with complexity $O(n)$; the traced curve appears as a simple path or cycle in the 1-skeleton of the new decomposition.

We apply our abstract tracing strategy to two different classes of normal curves: abstract curves represented by *normal coordinates*, which record the number of intersections with each edge of the surface triangulation, and simple *geodesics*, represented by a starting point and direction in the local coordinate system of some triangle. Our normal-coordinate algorithms are competitive with and conceptually simpler than earlier algorithms by Schaefer, Sedgwick, and Štefankovič [COCOON 2002, CCCG 2008] and by Agol, Hass, and Thurston [*Trans. AMS* 2005].

*Un poète doit laisser des traces de son passage, non des preuves.*
*Seules les traces font rêver.*

— René Char, *La Parole en Archipel* (1962)

*A typical simple closed curve on a surface is complicated,*
*from the point of view of someone tracing out the curve.*

— William P. Thurston, "On the geometry and dynamics of diffeomorphisms of surfaces" (1988)

*This work was partially supported by NSF grant CCF 09-15519.
†Portions of this work were done while this author was visiting IST Austria.

## 1 Introduction

Curves on abstract surfaces are usually represented by describing the interaction between the curve and a decomposition of the surface into elementary pieces. For example, given a triangulation of the surface, any sufficiently well-behaved[1] curve can be described by listing the sequence of edges of the triangulation that the curve crosses, in order along the curve. This *intersection sequence* identifies the curve up to a continuous deformation that avoids the vertices. We call a subpath of a curve between two consecutive edge crossings an *elementary segment*.

For *simple* curves, however, there are several more compact representations. For example, given a triangulation of the surface, any sufficiently well-behaved simple curve can be described by listing the number of elementary segments connecting each pair of edges in each triangle. These numbers are called the *normal coordinates* of the curve [15, 19]. The normal coordinates identify a unique simple curve (again up to continuous deformation), because there is only one way to fill each triangle with the correct number of elementary segments without intersection. The normal coordinate representation is remarkably compact; only $O(n \log(X/n))$ bits are needed to list the normal coordinates of a curve with $X$ crossings on a triangulated surface with complexity $n$. Several algorithms in two- and three-dimensional topology owe their efficiency to the compactness of the normal-coordinate representation [1, 16, 34, 35, 37, 39, 40].

Schaefer *et al.* [34, 37, 40] consider several algorithmic questions about normal curves, such as computing the number of components of a curve, deciding whether two given curves are isotopic, and computing algebraic and geometric intersection numbers of pairs of curves. Classical algorithms for these problems require explicit intersection sequences as input. By connecting normal coordinates with grammar-based text compression [21, 22, 24, 33] and word equations [9, 30, 31, 32], Schaefer *et al.* developed algorithms whose running times are polynomial in the bit complexity of the normal coordinates. These algorithms rely on a complex algorithm of Plandowski and Rytter [30] to compute compressed solutions of word equations. We are unaware of any precise time analysis, but as Plandowski and Rytter's algorithm uses a nested sequence of quadratic- and cubic-time reductions, its running time is quite high. Štefankovič [40] described simpler linear-time algorithms for some of these problems, by reducing them to an algorithm of Robson and Deikert [31, 32] to solve word equations with a certain special structure. Some of the problems considered by Schaefer *et al.* can also be solved in polynomial time using the orbit-counting algorithm of Agol *et al.* [1], which was designed for normal *surfaces* in triangulated 3-manifolds.

Other compact representations include weighted train tracks [4, 5, 13, 14, 29], Dehn-Thurston coordinates (with respect to a fixed pants decomposition of the surface) [7, 13, 14, 28, 42], and compressed intersection sequences [37, 40].

### Our Results

We propose an alternate strategy to efficiently compute with curves on surfaces. Instead of using complex compression techniques to avoid unpacking the intersection sequence of the input curve, our algorithms *modify the underlying cellular decomposition* of the surface so that the curve has a small *explicit* description with respect to the new decomposition. Specifically, given the normal coordinates of a curve $\gamma$ on a triangulated surface with $n$ edges, we compute a new cellular decomposition of the surface with complexity $O(n)$, called a *street complex*, such that $\gamma$ is a simple path or cycle in the 1-skeleton. After reviewing some background terminology, we formally define the street complex in Section 2; see Figure 1 for an example.

At a high level, our algorithm simply *traces* the curve, continuously updating the street complex to reflect the portion of the curve traced so far. A naïve implementation of our tracing strategy runs in $O(X)$

[1] We offer more formal definitions in Section 2.

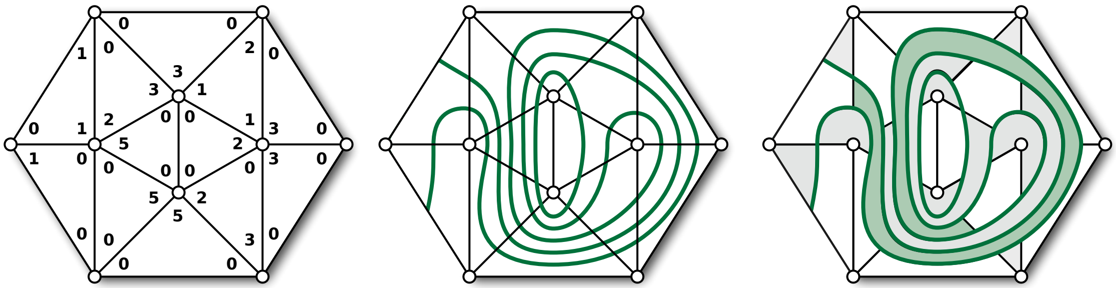


**Figure 1.** (a) Corner coordinates in a triangulated disk. (b) The corresponding normal curve. (c) The street complex of the curve. Unshaded faces are junctions; shaded faces are streets; one street is shaded darker (green) for emphasis.

time, where $X$ is the total number of edge crossings; each time the curve enters a triangle by crossing an edge, we can easily determine in $O(1)$ time which of the other two edges of the triangle to cross next. The main result of this paper is a tracing algorithm that runs in $O(n^2 \log X)$ time, an exponential improvement over the naïve algorithm for any fixed surface triangulation.

Our new algorithm relies on two simple ideas. First, we observe that for typical curves, most of the decisions made by the brute-force tracing algorithm are redundant. If a curve enters a triangle $\Delta$ between two older elementary segments that leave $\Delta$ through the same edge, the new elementary segment must also leave $\Delta$ through that edge. The street complex lets us skip these redundant decisions. Second, even with redundant decisions filtered out, the naïve algorithm may repeat the same series of crossings many times in a row when the input curve contains a *spiral* [27,36,38]. Our algorithm detects spirals as they occur, quickly determines the depth of the spiral (the number of repetitions), and then skips ahead to the first crossing after the spiral. We describe our generic tracing algorithm in Section 3 and analyze its running time in Section 4.

The street complex allows us to answer several fundamental topological questions about simple curves using elementary algorithms. For example, to determine whether a curve represented by normal coordinates is connected, we can trace one component of the curve, and then check whether the number of edge crossings we encountered is equal to the sum of the normal coordinates. To determine whether a connected normal curve is contractible, we can trace the curve and then apply a $O(n)$-time depth-first search in the dual of the resulting street complex [12].

In Section 5, we show that our tracing algorithm can be adapted to solve many of the problems considered by Schaefer *et al.* in $O(n^2 \log X)$ time. Our algorithms are significantly faster and simpler than the corresponding word-equation algorithms of Schaefer *et al.* [34] and the orbit-counting algorithm of Agol *et al.* [1]. However, for a few of the problems we consider, our algorithms are slower by a factor of $n$ than those described by Štefankovič [40].

In Section 6, we sketch and extension of our tracing algorithm to simple *geodesic* paths on piecewise-linear surfaces. Here, the input surface is presented as a set of $n$ triangles, each with its own local Euclidean coordinate system, with some pairs of equal-length edges identified; the geodesic path is specified by a starting point and a direction, in the local coordinate system of one of the triangles. In particular, we do *not* assume that the input surface is embedded (or embeddable) in any Euclidean space. As an example application, we sketch an algorithm to find the first point of self-intersection on a geodesic path in $O(n^2 \log X)$ time.

Due to space limitations, we defer many details to the appendix, especially in Section 6.

## 2 Background

We begin by recalling standard definitions from combinatorial topology; for further background, see Edelsbrunner and Harer [10] or Stillwell [41].

### 2.1 Surfaces, Curves, and Isotopy

A ***surface*** (more formally, a *2-manifold with boundary*) is a Hausdorff space in which every point has an open neighborhood homeomorphic to either the plane $\mathbb{R}^2$ or the closed halfplane $\{(x, y) \mid x \geq 0\}$. We consider only compact, connected, *orientable* surfaces in this paper. The set of points in a surface $\Sigma$ with halfplane neighborhoods is the ***boundary*** of the surface, denoted $\partial\Sigma$; the boundary is homeomorphic to a finite set of disjoint circles.

A ***simple cycle*** in a surface $\Sigma$ is (the image of) a continuous injective map $\gamma\colon S^1 \to \Sigma$. A ***simple path*** is (the image of) a continuous injective map $\pi\colon [0,1] \to \Sigma$; a ***simple arc*** $\alpha$ is a simple path whose endpoints $\alpha(0)$ and $\alpha(1)$ lie on the boundary $\partial\Sigma$. An arc is ***properly embedded*** if it intersects $\partial\Sigma$ only at its endpoints; similarly, a cycle is properly embedded if it avoids $\partial\Sigma$ entirely. A ***properly embedded curve*** is a finite collection of disjoint, properly embedded arcs and cycles.

A ***(proper) isotopy*** between two cycles $\gamma$ and $\gamma'$ is a continuous map $h\colon [0,1] \times S^1 \to \Sigma$ such that $h(0,\cdot) = \gamma$ and $h(1,\cdot) = \gamma'$, and $h(t,\cdot)$ is a properly embedded cycle for all $t \in [0,1]$. Similarly, a (proper) isotopy between two arcs $\alpha$ and $\alpha'$ is a continuous map $h\colon [0,1] \times [0,1] \to \Sigma$ such that $h(0,\cdot) = \alpha$ and $h(1,\cdot) = \alpha'$, and $h(t,\cdot)$ is a properly embedded arc for all $t \in [0,1]$. The definition of isotopy easily extends to properly embedded curves with multiple components. Two curves are ***isotopic***, or in the same ***isotopy class***, if there is an isotopy between them. A simple cycle or arc is ***contractible*** if it is isotopic to a point. The ***genus*** of a surface is the maximum number of disjoint simple cycles that can be removed without disconnecting the surface.

### 2.2 Triangulations, Normal Curves, and Coordinates

An ***embedding*** of a graph $G$ on a surface $\Sigma$ is a function mapping the vertices of $G$ to distinct points in $\Sigma$ and the edges of $G$ to paths in $\Sigma$ that are disjoint except at common endpoints. The ***faces*** of the embedding are maximal subsets of $\Sigma$ that are disjoint from the image of the graph. An embedding is ***cellular*** if every face is homeomorphic to an open disk; in particular, $\partial\Sigma$ must be the image of a set of disjoint cycles in $G$. A ***triangulation*** of $\Sigma$ is a cellularly embedded graph in which a walk around the boundary of any face has length three. Equivalently, a triangulation expresses $\Sigma$ as a set of disjoint triangles with certain pairs of edges identified; the ***1-skeleton*** of the resulting cell complex is the induced graph of vertices and edges.

We assume that our input surfaces are presented as triangulations, either as a set of triangles and gluing rules, or as an abstract graph with a rotation system [25]. We do *not* assume that triangulations are simplicial complexes. That is, triangulations may contain parallel edges and loops; two triangles may share more than a single vertex or a single edge; and the same triangle may be incident to a vertex or edge more than once.

Let $T$ be a triangulation of a surface $\Sigma$. A properly embedded curve $\gamma$ in $\Sigma$ is ***normal*** with respect to $T$ if (1) $\gamma$ avoids the vertices of $T$; (2) every intersection between $\gamma$ and an edge of $T$ is transverse; and (3) the intersection of $\gamma$ with any triangular face of $T$ is a finite set of disjoint ***elementary segments***: simple paths whose endpoints lie on distinct sides of the triangle. A ***normal isotopy*** between two normal curves is a proper isotopy $h$ such that $h(t,\cdot)$ is a normal curve for all $t$. Two curves are ***normal isotopic***, or in the same ***normal isotopy class***, if there is a normal isotopy between them.

**Lemma 2.1 (Kneser [19]).** *Let $\gamma$ be a normal curve on a triangulated surface with $n$ triangles, genus $g$, and $b$ boundary cycles. The components of $\gamma$ fall into at most $O(g+b)$ isotopy classes and at most $O(n)$ normal isotopy classes.*

A normal cycle is ***trivial*** if it bounds a disk in $\Sigma$ containing a single vertex of $T$. We call a normal curve ***reduced*** if no component of $\gamma$ is trivial and no two components of $\gamma$ are normal isotopic. Lemma 2.1 immediately implies that any reduced normal curve has at most $O(n)$ components.

Any normal curve can be identified, up to normal isotopy, by two different vectors of $O(n)$ non-negative integers, where $n$ is the number of faces of $T$. There are three types of elementary segments within any face $\Delta$, each separating one corner of $\Delta$ from the other two; the ***corner coordinates*** of $\gamma$ list the number of elementary segments of each type in each face of $T$. The ***edge coordinates*** of $\gamma$ list the number of times $\gamma$ intersects each edge of $T$. We collectively refer to the corner and edge coordinates of a curve as its ***normal coordinates***.[2] Given either normal coordinate representation, it is easy to compute the other representation in $O(n)$ time. The ***total crossing number*** of a normal curve is the sum of its edge coordinates.

### 2.3 Piecewise-Linear Surfaces and Geodesics

A ***piecewise-linear*** surface is a 2-manifold, possibly with boundary, constructed from a finite number of closed *Euclidean* polygons by identifying pairs of equal-length edges. The interiors of the constituent polygons are called *faces* of the surface; without loss of generality, we assume that all faces are triangles. The *vertices* and *edges* of the surface are the equivalence classes of vertices and edges of the polygons, respectively. The simplest example of a piecewise-linear surface is the boundary of a convex polyhedron in $\mathbb{R}^3$. However, most piecewise-linear surface cannot be embedded in *any* Euclidean space so that every face is flat; consider, for example, the *flat torus* obtained by identifying opposite sides of the unit square. Our algorithms assume only that the surface is orientable and that each face has its own local coordinate system; affine transformations between the local coordinate systems of neighboring faces can be derived from the gluing rules.

A ***geodesic*** is a path that is *locally* as short as possible; for any point $x$ in a geodesic $\gamma$, a sufficiently small neighborhood of $\gamma$ around $x$ is a shortest path. If $\gamma$ is a geodesic in a piecewise-linear surface $\Sigma$, any subpath of $\gamma$ that lies entirely within a face of $\Sigma$ is a straight line segment. Similarly, a subpath of $\gamma$ that crosses an edge of $\Sigma$ from one face $A$ to another face $B$ is a line segment in the polygon obtained by *unfolding* $A$ and $B$ into a common planar coordinate system [8, 23]. A geodesic is ***simple*** if it does not self-intersect.

### 2.4 Ports, Blocks, Junctions, and Streets

The intersections between any normal curve $\gamma$ and the edges of any triangulation $T$ partition $\gamma$ into ***elementary segments*** and partition the edges of $T$ into segments called ***ports***. The ***overlay graph*** $\boldsymbol{T \| \gamma}$ is the cellularly embedded graph whose edges are these elementary segments and ports. Every vertex of $T \| \gamma$ is either a vertex of $T$ or an intersection point of $\gamma$ and some edge of $T$. Every face of $T \| \gamma$ is a subset of some face $\Delta$ of $T$. We call each face a ***junction*** if it is incident to all three sides of its containing face $\Delta$, and a ***block*** if it is incident to only two sides of $\Delta$; these are the only two possibilities. Each face of $T$ contains exactly one junction. Each block is bounded either by two elementary segments and two ports, or by one elementary segment and two ports that share a vertex of $T$.

We call a port ***redundant*** if it separates two blocks; because each face of $T$ contains exactly one junction, each edge of $T$ contains at most two non-redundant ports. Removing all the redundant ports from the overlay graph $T \| \gamma$ merges contiguous sets of blocks into ***streets***. Each street is either a single disk with exactly two non-redundant ports on its boundary (called the ***ends*** of the street), a disk bounded by a trivial component of $\gamma$, or an annulus bounded by two normal isotopic components of $\gamma$. In particular, if $\gamma$ is reduced, all streets are of the first type. For any reduced normal curve $\gamma$, The ***street complex*** $\boldsymbol{S(T, \gamma)}$ is the complex of streets and junctions in the overlay $T \| \gamma$. See Figure 1.

By construction, the components of any reduced normal curve $\gamma$ appear as disjoint paths and cycles in the 1-skeleton of the street complex. Although the complexity of the overlay graph $T \| \gamma$ can be

[2] Schaefer *et al.* [34, 37, 40] refer to the edge coordinates as "normal coordinates", but the standard coordinate system for normal surfaces [15] is a generalization of corner coordinates.

arbitrarily large, the street complex $S(T,\gamma)$ is never more than a constant factor more complex than the original triangulation.

**Lemma 2.2.** *Let $T$ be a surface triangulation with $n$ triangles. For any* ***reduced*** *normal curve $\gamma$ in $\Sigma$, the street complex $S(T,\gamma)$ has complexity $O(n)$.*

Our restriction to reduced curves is necessary for two reasons. First, the number of components in a non-reduced curve can be arbitrarily larger than $n$. Second, in the street complex of any non-reduced curve $\gamma$, some faces are not open disks but open annuli, bounded either by two components of $\gamma$ or by one component of $\gamma$ and a vertex of $T$. Fortunately, as we argue in Section 5, it is easy to avoid tracing trivial components or more than one component in the same normal isotopy class.

The ***crossing length*** of a street is the number of constituent blocks plus one, or equivalently, the minimum number of edge intersections of a curve traversing the street from end to end. To simplify our analysis, we regard any port between two junctions, as well as any boundary port incident to a junction, as a street with crossing length 1. The sum of the crossing lengths of the streets in any street complex $S(T,\gamma)$ is the total crossing number of $\gamma$ plus the number of edges in $T$.

Any normal curve $\gamma'$ that is disjoint from $\gamma$ subdivides each port in $S(T,\gamma)$ into smaller ports, each street in $S(T,\gamma)$ into narrower "blocks", and each junction in $S(T,\gamma)$ into blocks and exactly one smaller junction. Removing all redundant ports from this refinement gives us the refined street complex $S(T,\gamma\cup\gamma')$. Conversely, the intersection of $\gamma'$ with any street or junction in $S(T,\gamma)$ is a set of elementary arcs. There are three types of elementary arcs within any junction, each connecting two of the junction's three ports. The ***junction coordinates*** of $\gamma'$ list the number of elementary arcs of each type in each junction of $S(T,\gamma)$. Similarly, the ***street coordinates*** of $\gamma'$ list the number of such arcs within each street of $S(T,\gamma)$. The normal coordinates of a curve $\gamma$ are just the junction and street coordinates of $\gamma$ in the trivial street complex $S(T,\varnothing)$.

Our tracing strategy must handle normal curves that are partially drawn on the surface; we slightly extend our definitions to include such curves. A ***normal path*** is any simple path whose endpoints lie in the interior of edges of $T$ and that can be extended to a normal curve on $\Sigma$. A ***fork*** is the union of two ports that share one of the endpoints of $\pi$; for most purposes, we can think of a fork as a degenerate junction.

## 3 Tracing Connected Normal Curves

In this section, we describe our algorithm to trace *connected* normal curves. Given a triangulation $T$ of an orientable surface $\Sigma$ and the corner and edge coordinates of a connected normal curve $\gamma$, our tracing algorithm computes the street complex $S(T,\gamma)$. We extend our algorithm to arbitrary *reduced* curves in Section 4, and we consider arbitrary normal curves in Section 5.

Our algorithm maintains a normal subpath $\pi$ of $\gamma$ that is growing at one end, along with the street complex $S(T,\pi)$ and the junction and street coordinates of the complementary path $\gamma\setminus\pi$. If $\gamma$ is an arc, we trace it from one endpoint to the other. If $\gamma$ is a cycle, we trace it starting at some intersection point with an edge of $T$. Initially, $\pi$ is an arbitrary crossing point, which splits some edge into a fork.

### 3.1 Steps

In each ***step*** of our algorithm, we extend the path $\pi$ through one junction or fork, and then through one street, updating both the street complex and the junction and street coordinates. We call the streets that contain the moving endpoint of $\pi$ the left and right ***active*** streets.

Suppose $\pi$ is about to enter a junction. We call the streets adjacent to the junction but not the endpoint of $\pi$ the *left exit* and the *right exit*. Suppose the local junction coordinates are $a$, $b$, and $c$, and the active street coordinates are $l$ and $r$, as shown in Figure 2. These coordinates satisfy the equation $l+r+1=a+c$, so either $l<a$ or $r<c$. If $l<a$, we extend $\pi$ through the junction and through its

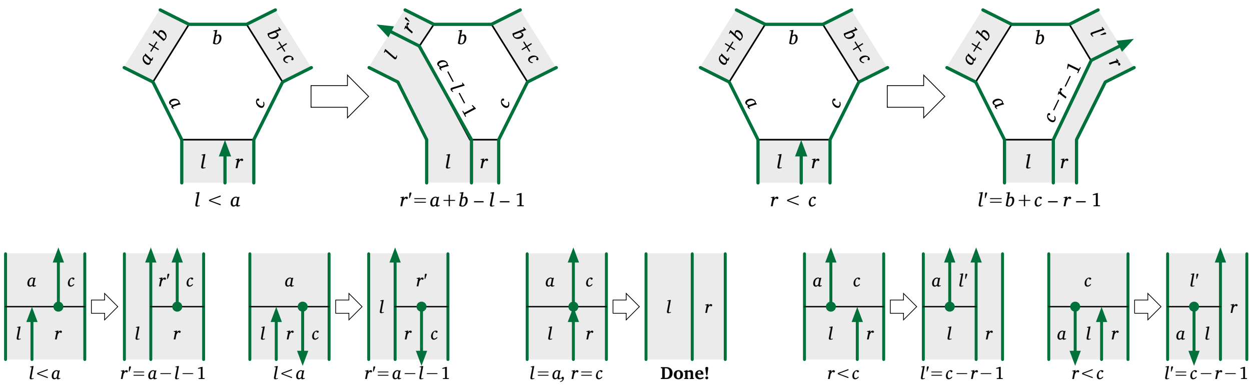


**Figure 2.** Tracing a curve through a junction or a fork.

left exit into the next junction; the left active street grows to the end of the left exit, and the left exit becomes the new right active street. We call this case a ***left turn***; the symmetric case $r < c$ is called a ***right turn***. In either case, we update the street and junction coordinates as shown in the top of Figure 2.

A similar complex case analysis applies if $\pi$ is about to cross a fork; see the bottom of Figure 2. In all cases, each step makes one active street longer, replaces the other active street, and makes the new active street narrower. All necessary operations for a single step—comparing and updating the junction and street coordinates and updating the street complex—can be performed in $O(1)$ time. The tracing algorithm ends when $\pi$ hits either the boundary of $\Sigma$ or the starting point of the trace.

### 3.2 Phases and Spirals

Unfortunately, executing each step by brute force is not necessarily efficient. To improve the brute-force algorithm, we more coarsely partition the tracing process into ***phases***. Each phase is a maximal sequence of either left turns or right turns. Every step in a phase consisting of left turns extends the same left active street; similarly, every step in a right phase extends the same right active street. In either case, each phase extends a single ***active street***.

During each phase, we maintain a sequence of *directed* streets and junctions traversed in that phase. If the growing path $\pi$ ever enters a street for the second time, in the same direction, during the same phase, then $\pi$ has entered a ***spiral***. The reentered street must be the first street traversed during the current phase; for the remainder of the phase, $\pi$ repeatedly traverses the same sequence of directed streets and junctions. The ***length*** of a spiral is the total number of streets it traverses, counted with multiplicity, and the ***depth*** of the spiral is the number of times we repeat the entire sequence of directed streets and junctions. If the spiral has length $\ell$ and traverses $m$ distinct directed streets, then the depth of the spiral is $\lceil \ell/m \rceil - 1$.

Instead of tracing the spiral step by step, we compute the depth of the spiral directly in $O(m)$ time as follows. Let $J_0, J_1, \ldots, J_{m-1}$ be the junction coordinates modified during the first iteration of the spiral. Let $w$ denote the ***width*** of the active street, defined as the corresponding street coordinate plus 1. The depth of the spiral is $d = \min_i \lfloor J_i/w \rfloor$, and the spiral ends at the first junction whose coordinate $J_i$ is smaller than $dw$. Once we compute $d$, we can update the street complex $S(T, \pi)$ and the appropriate street and junction coordinates in $O(m)$ time.

The crude upper bound $m = O(n)$ immediately implies that each phase of our tracing algorithm can be executed in $O(n)$ time. We analyze the number of phases, as a function of the total crossing number of the traced curve, in Section 4.

### 3.3 History

For some applications of our tracing algorithm, it is useful to maintain the ***history*** of the street complex, which records the evolution of each street during the algorithm's execution. For each phase, the history records which street is active for the entire phase, the sequence of $m$ distinct directed streets traversed during the phase, and the length $\ell$ of the phase. The resulting history is equivalent to a context-free grammar whose terminals are the edges of $T$, whose variables are the directed streets extended in each phase of the tracing algorithm, and whose productions have the form

$$S_a \to S_b\,(S_{i_0}\,S_{i_1}\cdots S_{i_{m-1}})^d\,S_{i_0}\,S_{i_1}\cdots S_{i_{(\ell-1)\bmod m}} \quad\text{or}\quad \overline{S}_a \to \overline{S}_{i_{(\ell-1)\bmod m}}\cdots\overline{S}_{i_1}\,\overline{S}_{i_0}(\overline{S}_{i_{m-1}}\cdots\overline{S}_{i_1}\,\overline{S}_{i_0})^d\,\overline{S}_b.$$

The language of each non-terminal $S_i$ is a single string, recording the intersection sequence of the street at the end of some phase. In the example above, $S_a$ is the intersection sequence of the active street just *after* the phase ends; $S_b$ is the intersection sequence of the active street just *before* the phase begins; $i_0, i_1, \ldots, i_{m-1}$ are the indices of the directed streets traversed during the first iteration of the spiral; and $\overline{S}_j$ denotes the reversal of $S_j$. We may have $S_{i_j} = \overline{S}_{i_k}$ for some indices $j$ and $k$, but otherwise the indices $i_j$ are distinct.

This context-free grammar can be transformed into Chomsky normal form by replacing each production in the form above with $O(m + \log d)$ productions of the form $A \to B\,C$. (Chomsky normal form grammars whose non-terminals generate exactly one string are often called *straight-line programs*.) Thus, our history data structure is essentially equivalent to the *compressed intersection sequence* constructed by Schaefer *et al.* [37, 40]. We analyze the complexity of our history data structure and the resulting compressed intersection sequence in the next section.

## 4 Analysis

We now bound the running time of our tracing algorithm. To simplify notation, we let $n$ denote the number of *edges* of the surface triangulation rather than the number of faces. Our time bounds are functions of both $n$ and $X$, the total crossing number of the traced curve. We also assume that $X = \Omega(n^2)$, since otherwise our analysis yields a time bound slower than the trivial bound $O(n + X)$.

In Section 4.1, we bound the time required to trace a *connected* normal curve. We extend our analysis to curves with multiple components in Section 4.2 and to the complexity of compressed intersection sequences in Section 4.3.

### 4.1 Abstract Tracing

In each phase of our tracing algorithm, the crossing length of the active street increases by the sum of the crossing lengths of the other traversed streets, counted with appropriate multiplicity. The algorithm AbstractTrace, shown on the left of Figure 3, abstractly models this growth.

AbstractTrace maintains an array $x[1\,..\,n]$ of positive integers, corresponding to the crossing lengths of the streets maintained in our tracing algorithm, along with the index $a$ of the current active street. Each iteration of the outer loop of AbstractTrace models a phase of our tracing algorithm. The inner loops update the crossing length $x[a]$ of the active street as the curve traverses a spiral of length $\ell$ and depth $d$, containing $m$ distinct streets whose indices are in the vector $(i_0, i_1, \ldots, i_{m-1})$. The last street traversed in the current phase becomes the active street for the next phase. For purposes of analysis, we assume that the parameters $\ell$, $m$, and $(i_0, i_1, \ldots, i_{m-1})$, as well as the termination condition for the outer loop, are determined *adversarially* rather than being determined by the topology of the curve.

To prove an upper bound on the number of phases of AbstractTrace, we can assume conservatively that $m = 1$ in every phase; equivalently, we can ignore the contribution to the active street's crossing length from all but the last street in every spiral. Thus, we consider the simpler algorithm SimpleTrace shown on the right of Figure 3. The new variable $\delta$ is the number of times the last street in the spiral is

```
AbstractTrace(n):
  for j ← 1 to n
      x[j] ← 1
  a ← 1
  while not done
      choose an integer m ∈ [n]
      choose an integer ℓ ≥ m
      choose a vector (i_0, i_1, ..., i_{m−1}) ∈ [n]^m
      d ← ⌈ℓ/m⌉ − 1
      for j ← 0 to m − 1
          x[a] ← x[a] + d · x[i_j]
      for j ← 0 to (ℓ − 1) mod m
          x[a] ← x[a] + x[i_j]
      a ← i_{(ℓ−1) mod m}
```

```
SimpleTrace(n):
  for j ← 1 to n
      x[j] ← 1
  Δ ← 0
  a ← 1
  while not done
      choose an index i ∈ [n]
      choose an integer δ ≥ 1
      x[a] ← x[a] + δ · x[i]
      Δ ← Δ + lg(δ + 1)
      a ← i
```

**Figure 3.** Our abstract tracing algorithm and a simplified version for analysis.

traversed; specifically, $\delta = d$ if $\ell/m$ is an integer and $\delta = d + 1$ otherwise. The other new variable $\Delta$ is used only in the analysis.

**Lemma 4.1.** *At the end of each iteration of* SimpleTrace*, we have* $\Delta \leq 2\sum_{i=1}^{n} \lg x[i]$.

**Proof:** Consider the potential function $\Phi := 2\sum_{i=1}^{n} \lg x[i] - \lg x[a]$. Initially we have $\Phi = 0$. There are two cases to consider, depending on whether $x[a]$ is smaller or larger than $d \cdot x[i]$ at the start of each iteration of the loop.

- If $x[a] \leq \delta \cdot x[i]$, then the assignment $x[a] \leftarrow x[a] + \delta \cdot x[i]$ increases $\Phi$ by at least $\lg(\delta + 1)$, and the assignment $h \leftarrow i$ does not decrease $\Phi$.
- If $x[a] \geq \delta \cdot x[i]$, then the assignment $x[a] \leftarrow x[a] + \delta \cdot x[i]$ does not decrease $\Phi$, and the assignment $h \leftarrow i$ increases $\Phi$ by at least $\lg(\delta + 1)$.

In both cases, $\Phi$ increases by at least $\lg(\delta + 1)$ in each iteration. It immediately follows by induction that $\Delta \leq \Phi \leq 2\sum_{i=1}^{n} \lg x[i]$ at the end of every iteration. □

**Lemma 4.2.** AbstractTrace$(n)$ *runs for at most* $2L = O(n \log X)$ *phases, where* $L$ *is the final value of* $\sum_{i=1}^{n} \lg x[i]$ *and* $X$ *is the final value of* $\sum_{i=1}^{n} x[i]$.

**Proof:** To maximize the number of phases, we assume that $\ell = 1$ in every phase. This assumption allows us to simplify the execution to an instance of SimpleTrace where $\delta = 1$ in every phase, and therefore $\Delta$ is simply the number of phases. Lemma 4.1 implies that the algorithm terminates after at most $2L$ phases. The parameter $L$ is maximized as a function of $n$ and $X$ when $x[i] = X/n$ for all $i$. (Our assumption that $X = \Omega(n^2)$ implies that $\log(X/n) = \Theta(\log X)$.) □

**Corollary 4.3.** AbstractTrace$(n)$ *runs in* $O(nL) = O(n^2 \log X)$ *time, where* $L$ *is the final value of* $\sum_{i=1}^{n} \lg x[i]$ *and* $X$ *is the final value of* $\sum_{i=1}^{n} x[i]$.

**Theorem 4.4.** *Let* $\Sigma$ *be a surface composed of* $n$ *triangles, and let* $\gamma$ *be a* ***connected*** normal *curve in* $\Sigma$ *with total crossing number* $X$*. Given the normal coordinates of* $\gamma$*, we can trace* $\gamma$ *in* $O(n^2 \log X)$ *time.*

The bounds reported in Lemma 4.2 and Corollary 4.3 are tight in the worst case, at least when $X$ and $n$ are sufficiently large; see Appendix B.2 for details. However, we optimistically conjecture that the upper bound in Theorem 4.4 is *not* tight.

### 4.2 Tracing Disconnected Curves

Now consider the more general case where $\gamma$ is a *reduced* curve, possibly with more than one component. (For the application we describe in Section 5, this is the most general case we need to consider.) Our tracing algorithm requires little modification to handle these curves; we simply trace the components one at a time, in arbitrary order. Each component refines the street complex defined by the previous components. Lemma 2.2 immediately implies that the resulting algorithm runs in $O(n^3 \log X)$ time, but this time bound can be improved with more careful analysis; see Appendix B for details.

**Theorem 4.5.** *Let $\Sigma$ be a surface composed of $n$ triangles, and let $\gamma$ be a **reduced** normal curve in $\Sigma$ with total crossing number $X$. Given the normal coordinates of $\gamma$, we can trace every component of $\gamma$ in $O(n^2 \log X)$ time.*

### 4.3 Compression: Logarithmic Spiral Cost

Recall that our history data structure can be transformed into a straight-line program encoding the intersection sequence of a connected curve $\gamma$. For each phase of the tracing algorithm, the straight-line program contains $O(m + \log d)$ productions, where $m$ is the number of distinct streets traversed in that phase and $d$ is the depth of that phase's spiral. Lemma 4.1 implies the following upper bound on the length of this straight-line program.

**Theorem 4.6.** *Let $\Sigma$ be a triangulated surface with $n$ triangles, and let $\gamma$ be a **reduced** normal curve in $\Sigma$ with total crossing number $X$. Given the normal coordinates of $\gamma$, we can compute a straight-line program of length $O(n^2 \log X)$ that encodes the intersection sequence of $\gamma$ in $O(n^2 \log X)$ time.*

After tracing a disconnected normal curve $\gamma$, our history data structure can be transformed into a straight-line program such that the intersection sequence of any component of $\gamma$ is the language of some non-terminal. The proof of Theorem 4.5, combined with Lemma 4.1, implies that the total length of this straight-line program is also $O(n^2 \log X)$.

## 5 Counting Isotopy Classes

Our tracing algorithm implies efficient solutions for several problems involving normal curves represented by their normal coordinates. As an example, we describe an algorithm to compute the number of distinct isotopy classes of components of a given normal curve, as well as the number of components in each isotopy class. The input to our algorithm is a surface triangulation $T$ with $n$ triangles, and the edge and corner coordinates a normal curve $\gamma$ with total crossing length $X$. Our algorithm runs in $O(n^2 \log X)$ time, significantly improving the large-polynomial-time solution of Schaefer *et al.* [34]. Due to space limitations, our description omits many low-level details; algorithms for several related problems will be described in the full paper.

We start by counting and deleting the trivial components of $\gamma$. The number of trivial components that separate any vertex $v$ from the other vertices is just the minimum of the corner coordinates incident to $v$. Thus, we can easily count all trivial components and delete them from $\gamma$, by reducing the appropriate normal coordinates, in $O(n)$ time. We separately record the number of trivial cycles and the number of trivial arcs with endpoints on each boundary cycle.

Next, for each non-trivial *normal* isotopy class, we trace one component of $\gamma$ in that class, and then count and remove all other components in that class, as follows. Suppose we have already traced components $\gamma_1, \ldots, \gamma_{i-1}$. Let $\hat{\gamma}_{<i}$ denote the reduced normal curve $\gamma_1 \cup \cdots \cup \gamma_{i-1}$, and let $\gamma_{\geq i}$ denote the union of all components of $\gamma$ *not* normal-isotopic to any component of $\hat{\gamma}_{<i}$. In particular, we have $\hat{\gamma}_{<1} = \varnothing$ and $\gamma_{\geq 1} = \gamma$. By assumption, we have computed the street complex $S(T, \hat{\gamma}_{<i})$ as well as the street and junction coordinates of $\gamma_{\geq i}$. Let $x$ be the leftmost intersection point between $\gamma_{\geq i}$ and

some non-redundant port $p$ in $S(T, \hat{\gamma}_{<i})$, and let $\gamma_i$ denote the component of $\gamma_{\geq i}$ that contains $x$. We trace $\gamma_i$ through $S(T, \hat{\gamma}_{<i})$ to produce the street complex $S(T, \hat{\gamma}_{<(i+1)})$, along with the street and junction coordinates of $\gamma_{\geq i} \setminus \gamma_i$. The number of other components of $\gamma$ that are normal isotopic to $\gamma_i$ is the minimum of the junction coordinates just to the right of $\gamma_i$ in the new street complex $S(T, \hat{\gamma}_{<(i+1)})$. Thus, we can easily count these components and reduce the appropriate street and junction coordinates in $O(n)$ time, thereby computing the street and junction coordinates of $\gamma_{\geq(i+1)}$.

Theorem 4.5 implies that the total time spent tracing all components $\gamma_i$ is $O(n^2 \log X)$. Lemma 2.1 implies that there are at most $O(n)$ normal-isotopy classes of components in $\gamma$, so the total time spent counting and removing parallel components of $\gamma$ is only $O(n^2)$.

Finally, let $\hat{\gamma}$ be the union of the traced components $\gamma_i$. Each traced component $\gamma_i$ is a simple arc or cycle in the 1-skeleton of the final street complex $S(T, \hat{\gamma})$. Thus, we can compute the Euler characteristic of every component of $\Sigma \setminus \hat{\gamma}$ in $O(n)$ time using a depth-first search in the dual graph of $S(T, \hat{\gamma})$ [12]; in particular, we can identify which components of $\Sigma \setminus \hat{\gamma}$ are disks ($\chi = 1$) and annuli ($\chi = 0$). Then it is straightforward to cluster the components of $\hat{\gamma}$ into isotopy classes in $O(n)$ time. We can also compute the number of components of $\gamma$ in each isotopy class in $O(n)$ time by adding the sizes of the appropriate normal-isotopy classes.

**Theorem 5.1.** *Let $\Sigma$ be a triangulated surface with $n$ triangles, and let $\gamma$ be a normal curve in $\Sigma$ with total crossing length $X$, represented by its normal coordinates. We can compute the number of isotopy classes of components of $\gamma$ and the number of components in each isotopy class in $O(n^2 \log X)$ time.*

The algorithm of Schaefer *et al.* [34] to identify isotopy classes actually computes the normal coordinates of one component in each isotopy class. If we require the same output representation, we can recover the normal coordinates of any component of $\hat{\gamma}$ (or the union of any subset of components) in $O(n^2 \log X)$ time, essentially by running the tracing algorithm backward; see Appendix C for details. Lemma 2.1 implies that the overall time to compute the normal coordinates of every component of $\hat{\gamma}$ is $O((g + b)n^2 \log X)$, which is still significantly faster than Schaefer *et al.*

## 6 Tracing Geodesics

Finally, we extend our tracing algorithm to simple geodesic paths on piecewise-linear triangulated surfaces. The input to our algorithm is a piecewise-linear surface $\Sigma$ (specified by triangles and gluing rules), along with a starting point $p$ and a direction vector $v$ in the local coordinate system of some face of $\Sigma$ that contains $p$. As an example application, we sketch an algorithm to trace the geodesic path $\gamma$ that starts at $p$ in direction $v$ up to its first point of self-intersection.

To simplify our exposition, we assume that both the surface $\Sigma$ and the direction vector $v$ are *generic*; thus, the geodesic $\gamma$ does not intersect any vertex of $\Sigma$ but does eventually intersect itself.[3] e emphasize that even for generic inputs, the total crossing number of $\gamma$ is not bounded *a priori* by *any* function of $n$.

Any simple geodesic path $\gamma$ that starts and ends on edges of the triangulation is a *normal* path. Thus, the street complex of $\gamma$ is well-defined and has complexity $O(n)$ by Lemma 2.2. Moreover, each of the $O(n)$ faces of the street complex is isometric to a convex polygon with at most six sides; in particular, every street is either a triangle or a convex quadrilateral. (A street can have $\Omega(n)$ vertices of the street complex on its boundary, but at most four of those vertices .)

Although we do not know the normal coordinates of $\gamma$ in advance, we can still easily decide in constant time at each step of our tracing algorithm whether a geodesic $\gamma$ entering a junction leaves through its left exit, leaves through its right exit, or hits an earlier segment of $\gamma$. Geometrically, this

[3] In a piecewise-linear surface where the total angle around every vertex is an integer multiple of $\pi$, almost every geodesic path can be extended infinitely without self-intersection. Examples of such surfaces include the boundary of a regular tetrahedron, the flat torus defined by identifying opposite edges of any parallelogram, and the *eierlegende Vollmilchsau* [17].

decision is equivalent to a ray-shooting query in a convex polygon with at most six sides. Thus, we can easily adapt the brute force tracing algorithm to the geodesic setting. However, to achieve a running time of $O(n^2 \log X)$, we require a new algorithm to efficiently compute the depth of a geodesic spiral.

We reduce computing spiral depth to the following ***annular ray shooting*** problem, which may be of independent interest: Given a ray $\rho$ on a piecewise-linear annulus $A$, how many times does $\rho$ wrap around $A$ before hitting the boundary? More formally, suppose we are given a triangulated simple polygon $P$ in the plane, with two edges $e_0$ and $e_1$ of equal length, and a ray $\rho$ that starts on $e_0$ and points into $P$. The annulus $A$ is obtained from $P$ by identifying $e_0$ and $e_1$. Equivalently, $e_0$ and $e_1$ are *portals*; when the ray exits $P$ at any point on $e_1$, it immediately reenters $P$ through the corresponding point on $e_0$ at the same incidence angle [11, 26, 44, 45, 46]. Our goal is to determine the number of times $\rho$ crosses the portal(s) before hitting a non-portal edge of $P$.

The naïve solution to this requires $O(n + t^* \log n)$ time, where $t^*$ is the output of the query: Preprocess $P$ for ray-shooting queries in $O(n)$ time, and then perform $t^* + 1$ queries, each in $O(\log n)$ time [18]. Our algorithm improves this naïve bound exponentially, by combining the standard unbounded search strategy of Bentley and Yao [3] with careful $O(n)$-time pre- and post-processing to ensure that each step of the unbounded search requires only $O(1)$ time.

**Lemma 6.1.** *The annular ray-shooting problem can be solved in $O(n + \log t^*)$ time and space, where $n$ is the number of edges in $P$ and $t^*$ is the output value.*

**Theorem 6.2.** *Let $\Sigma$ be a triangulated piecewise-linear surface with $n$ triangles. Given the starting point and direction of a geodesic $\gamma$ in $\Sigma$, we can compute the first self-intersection point in $\gamma$ in $O(n^2 \log X)$ time, where $X$ is the number of edges $\gamma$ crosses before it self-intersects.*

## References


[1] Ian Agol, Joel Hass, and William P. Thurston. The computational complexity of knot genus and spanning area. *Trans. Amer. Math. Soc.* 358(9):3821–3850, 2006. ArXiv:math/0205057.

[2] David Avis, Teren Gum, and Godfried T. Toussaint. Visiblity between two edges of a simple polygon. *Vis. Comput.* 2:342–357, 1986.

[3] Jon Louis Bentley and Andrew Chi-Chih Yao. An almost optimal algorithm for unbounded searching. *Inform. Proc. Lett.* 5:82–87, 1976.

[4] Joan S. Birman and Caroline Series. Geodesics with bounded intersection number on surfaces are sparsely distributed. *Topology* 24(2):217–225, 1985.

[5] Joan S. Birman and Caroline Series. Algebraic linearity for an automorphism of a surface group. *J. Pure Appl. Algebra* 52:227–275, 1988.

[6] Bernard Chazelle. A theorem on polygon cutting with applications. *Proc. 23rd Ann. IEEE Symp. Found. Comput. Sci.*, 339–349, 1982.

[7] Max Dehn. Die Gruppe der Abbildungsklassen (Das arithmetische Feld auf Flächen). *Acta Mathematica* 69:135–206, 1938.

[8] Erik D. Demaine and Joseph O'Rourke. *Geometric Folding Algorithms: Linkages, Origami, Polyhedra*. Cambridge Univ. Press, 2007.

[9] Volker Diekert and Manfred Kufleitner. A remark about quadratic trace equations. *Proc. 6th Int. Conf. Devel. Language Theory*, 59–66, 2003. Lecture Notes Comput. Sci. 2450, Springer-Verlag.

[10] Herbert Edelsbrunner and John Harer. *Computational Topology, An Introduction*. American Mathematical Society, 2010.

[11] Harlan Ellison. The city on the edge of forever. *Star Trek*, season 1, episode 28, April 6, 1967.

[12] Jeff Erickson and Sariel Har-Peled. Optimally cutting a surface into a disk. *Discrete Comput. Geom.* 31(1):37–59, 2004.

[13] Albert Fathi, François Laudenbach, and Valentin Poénaru. *Travaux de Thurston sur les surfaces*. Astérisque 66-67. Soc. Math. de France, 1979. Séminaire Orsay. English translation in [14].

[14] Albert Fathi, François Laudenbach, and Valentin Poénaru. *Thurston's Work on Surfaces*. Mathematical Notes. Princeton Univ. Press, 2011. Translated by Djun Kim and Dan Margalit. English translation of [13].

[15] Wolfgang Haken. Theorie der Normalflächen: Ein Isotopiekriterium für den Kreisknoten. *Acta Mathematica* 105:245–375, 1961.

[16] Joel Hass, Jeffrey C. Lagarias, and Nicholas Pippenger. The computational complexity of knot and link problems. *J. ACM* 46(2):185–211, 1999.

[17] Frank Herrlich and Gabriela Schmithüsen. An extraordinary origami curve. *Math. Nachr.* 281(2):219–237, 2008.

[18] John Hershberger and Subhash Suri. A pedestrian approach to ray shooting: Shoot a ray, take a walk. *J. Algorithms* 18(3):403–431, 1995.

[19] Helmuth Kneser. Geschlossene Flächen in dreidimensionalen Mannigfaltigkeiten. *Jahresbericht Deutschen Math.-Verein.* 38:248–260, 1930.

[20] Der-Tsai Lee and Franco P. Preparata. Euclidean shortest paths in the presence of rectilinear barriers. *Networks* 14:393–410, 1984.

[21] Yury Lifshits. *Algorithms and Complexity Analysis for Processing Compressed Texts*. Ph.D. thesis, Steklov Inst. Math., May 2007. In Russian.

[22] Yury Lifshits. Processing compressed texts: A tractabiliity border. *Proc. 18th Ann. Symp. Combin. Pattern Matching*, 228–240, 2007. Lecture Notes Comput. Sci. 4850, Springer-Verlag.

[23] Lazar Aronovich Lyusternik. *Shortest Paths: Variational Problems*. Popular Lectures Math. 13. Pergamon Press, 1964. Translated and adapted from the Russian by P. Collins and Robert B. Brown.

[24] Masamichi Miyazaki, Ayumi Shinohara, and Masayuki Takeda. An improved pattern matching algorithm for strings in terms of straight-line programs. *J. Discrete Algorithms [Hermes]* 1(1):187–204, 2000.

[25] Bojan Mohar and Carsten Thomassen. *Graphs on Surfaces*. Johns Hopkins Univ. Press, 2001.

[26] Nuclear Monkey Software. Narbacular Drop. Video game, 2005.

[27] János Pach and Géza Tóth. Recognizing string graphs is decidable. *Discrete Comput. Geom.* 28(4):593–606, 2001.

[28] Robert C. Penner. The action of the mapping class group on curves in surfaces. *L'Enseignment Mathématique* 30:39–55, 1984.

[29] Robert C. Penner and John L. Harer. *Combinatorics of Train Tracks*. Annals of Math. Studies 125. Princeton Univ. Press, 1992.

[30] Wojciech Plandowski and Wojciech Rytter. Application of Lempel-Ziv encodings to the solution of word equations. *Proc. 25th Int. Conf. Automata Lang. Prog.*, 731–742, 1998. Lecture Notes Comput. Sci. 1443, Springer-Verlag.

[31] John M. Robson and Volker Diekert. On quadratic word equations. *Proc. 16th Ann. Conf. Theoretical Aspects Comput. Sci.*, 217–226, 1999. Lecture Notes Comput. Sci. 1563, Springer-Verlag.

[32] John M. Robson and Volker Diekert. Quadratic word equations. *Jewels are Forever, Contributions on Theoretical Computer Science in Honor of Arto Salomaa*, 314–326, 1999. Springer-Verlag.

[33] Wojciech Rytter. Application of Lempel-Ziv factorization to the approximation of grammar-based compression. *Theoret. Comput. Sci.* 302:211–222, 2003.

[34] Marcus Schaefer, Eric Sedgwick, and Daniel Štefankovič. Algorithms for normal curves and surfaces. *Proc. 8th Int. Conf. Comput. Combin.*, 370–380, 2002. Lecture Notes Comput. Sci. 2387, Springer-Verlag.

[35] Marcus Schaefer, Eric Sedgwick, and Daniel Štefankovič. Recognizing string graphs in NP. *J. Comput. System Sci.* 67(2):365–380, 2003.

[36] Marcus Schaefer, Eric Sedgwick, and Daniel Štefankovič. Spiraling and folding: The topological view. *Proc. 19th Ann. Canadian Conf. Comput. Geom.*, 73–76, 2007.

[37] Marcus Schaefer, Eric Sedgwick, and Daniel Štefankovič. Computing Dehn twists and geometric intersection numbers in polynomial time. *Proc. 20th Canadian Conf. Comput. Geom.*, 111–114, 2008. Full version: Tech. Rep. 05–009, Comput. Sci. Dept., DePaul Univ., April 2005, ⟨http://facweb.cs.depaul.edu/research/techreports/abstract05009.htm⟩.

[38] Marcus Schaefer, Eric Sedgwick, and Daniel Štefankovič. Spiraling and folding: The word view. *Algorithmica* 60(3):609–626, 2011.

[39] Saul Schleimer. Sphere recognition lies in NP. Preprint, July 2004. ArXiv:math/0407047.

[40] Daniel Štefankovič. *Algorithms for simple curves on surfaces, string graphs, and crossing numbers*. Ph.D. thesis, Dept. Comput. Sci., Univ. Chicago, June 2005.

[41] John Stillwell. *Classical Topology and Combinatorial Group Theory*, 2nd edition. Graduate Texts in Mathematics 72. Springer-Verlag, 1993.

[42] William P. Thurston. On the geometry and dynamics of diffeomorphisms of surfaces. *Bull. Amer. Math. Soc.* 19(2):417–431, 1988. Circulated as a preprint in 1976.

[43] Godfried T. Toussaint. Shortest path solves edge-to-edge visibility in a polygon. *Patt. Recog. Letters* 4(3):165–170, 1986.

[44] Valve Corporation. Portal. Video game, 2007.

[45] Valve Corporation. Portal 2. Video game, 2011.

[46] Andy Wachowski and Larry Wachowski. *Matrix Revolutions*. Warner Bros., 2003. Motion picture.

# Appendix

## A Omitted Proofs from Section 2

**Lemma 2.1 (Kneser [19]).** *Let $\gamma$ be a normal curve on a triangulated surface with $n$ triangles, genus $g$, and $b$ boundary cycles. The components of $\gamma$ fall into at most $O(g+b)$ isotopy classes and at most $O(n)$ normal isotopy classes.*

**Proof:** First we consider the isotopy classes of components of $\gamma$. We separately bound the noncontractible cycles, noncontractible arcs, and contractible components, so our analysis is not tight.

Let $\mathcal{C}$ be a maximal set of pairwise-disjoint noncontractible *cycles* in distinct isotopy classes. Cutting the surface along any cycle leaves the Euler characteristic of the surface unchanged. Each component of $\Sigma \setminus \mathcal{C}$ is either a pair of pants bounded by three cycles in $\mathcal{C}$ or an annulus bounded by a cycle in $\mathcal{C}$ and a boundary cycle of $\Sigma$. A pair of pants has Euler characteristic $-1$, and each annulus contains exactly one boundary cycle of $\Sigma$. Thus, $\Sigma \setminus \mathcal{C}$ consists of $-\chi(\Sigma) = 2g + b - 2$ pairs of pants and $b$ annuli. It follows that $|\mathcal{C}| = (3(2g+b-2)+b)/2 = 3g + 2b - 3$.

Finally, let $\mathcal{A}$ be a maximal set of pairwise-disjoint noncontractible arcs in distinct isotopy classes. Each component of $\Sigma \setminus \mathcal{A}$ is a disk bounded by exactly three arcs in $\mathcal{A}$ and three boundary arcs. Contracting each boundary cycle of $\Sigma$ to a point transforms $\mathcal{A}$ into a $b$-vertex triangulation of a surface of genus $g$. Thus, Euler's formula implies that $|\mathcal{A}| = 3g + 3b - 6$.

All contractible cycles in $\Sigma$ are isotopic. Two contractible arcs are isotopic if and only if their endpoints lie on the same boundary cycle of $\Sigma$; thus, contractible arcs fall into at most $b$ isotopy classes. We conclude that the components of $\gamma$ lie in at most $6g + 6b - 8$ distinct isotopy classes.

To bound the number of *normal*-isotopy classes, consider the surface $\Sigma^\circ$ obtained by removing a small disk around every vertex. Two curves in $\Sigma$ are normal isotopic in $\Sigma$ if and only if they are isotopic in $\Sigma^\circ$. Thus, our earlier analysis implies that the components of $\gamma$ fall into at most $6n_0 + 6g + 6b - 8$ normal isotopy classes, where $n_0$ is the number of vertices of $\Sigma$. We trivially have $n_0 \le 3n$, and Euler's formula implies that $g + b = O(n)$. □

**Lemma 2.2.** *Let $T$ be a surface triangulation with $n$ triangles. For any **reduced** normal curve $\gamma$ in $\Sigma$, the street complex $S(T,\gamma)$ has complexity $O(n)$.*

**Proof:** The triangulation $T$ trivially has at most $3n$ vertices and at most $3n$ edges. Each interior edge of $T$ contains at most two non-redundant ports, so $S(T,\gamma)$ has $O(n)$ interior vertices. Each boundary vertex of $S(T,\gamma)$ is either a boundary vertex of $T$ or an endpoint of one of the $O(n)$ components of $\gamma$, so $S(T,\gamma)$ has $O(n)$ boundary vertices. Each vertex of $S(T,\gamma)$ is either a vertex of $T$ or has degree at most 4, so $S(T,\gamma)$ has $O(n)$ edges. Each non-redundant port is an end of at most one street, so $S(T,\gamma)$ has $O(n)$ streets. Finally, $S(T,\gamma)$ has exactly $n$ junctions, one in each triangle of $T$. □

## B Tracing Analysis

### B.1 Disconnected Curves

**Theorem 4.5.** *Let $\Sigma$ be a surface composed of $n$ triangles, and let $\gamma$ be a reduced normal curve in $\Sigma$ with total crossing number $X$. Given the normal coordinates of $\gamma$, we can trace every component of $\gamma$ in $O(n^2 \log X)$ time.*

**Proof:** Consider the effect of ending one component and starting another on the vector of crossing lengths modeled by the array $x[1..n]$ in SimpleTrace. When we begin tracing a new cycle component, we split some street into three smaller streets by introducing a fork; one of the three new streets becomes the active street for the first phase of the new component. This update can be modeled in SimpleTrace by adding the following lines:

```
if starting a cycle:
    choose an index i ∈ [n]
    choose an integer y ∈ [x[i]]
    x[i] ← x[i] − y + 1
    x[n+1] ← y
    x[n+2] ← y
    n ← n + 2
    a ← n + 1
```

When we finish tracing a cycle component, we merge the four streets adjacent to the initial fork into two longer streets; see the bottom center of Figure 2. This update can be modeled in SimpleTrace by adding the following lines:

```
if ending a cycle:
    choose an index j ∈ [n]
    choose an index k ∈ [n]
    x[j] ← x[j] + x[n−1]
    x[k] ← x[k] + x[n]
    n ← n − 2
```

Similarly, when we begin tracing a new arc component, we split some street into two narrower streets; this update can be modeled in SimpleTrace by adding the following lines:

```
if starting an arc:
    choose an index i ∈ [n]
    x[n+1] ← x[i]
    n ← n + 1
    a ← n + 1
```

No additional changes are necessary when we end an arc component.

Altogether, ending one component and starting a new one decreases the potential function $\Phi$ by at most $O(\log X)$. An easy modification of the proof of Lemma 4.1 now implies that after each iteration of SimpleTrace, we have $\Delta \le 2\sum_{i=1}^{n} \lg x[i] + O(t \lg X)$, where $t$ is the number of components we have completely traced so far. Lemma 2.1 implies that any reduced normal curve has $O(n)$ components. We conclude that SimpleTrace executes at most $O(n \log X)$ phases; each phase requires $O(n)$ time. □

### B.2 Matching Lower Bounds

Here we show that the analysis of AbstractTrace in Lemmas 4.2 and Corollary 4.3 is tight.

Suppose the adversary chooses $i = (a \bmod n) + 1$ and $\delta = 1$ in every phase of SimpleTrace. An easy inductive argument implies that for any integer $r \ge 1$, at the end of $r \cdot (n-1)$ phases we have $x[i] \le 2^r$ for all $i$. Thus, SimpleTrace must perform at least $(n-1)\lg(X/n) = \Omega(n \log X)$ iterations before $\sum_i x[i] = X$. We conclude that AbstractTrace can execute $\Omega(n \log X)$ phases in the worst case.

Corollary 4.3 is also tight in the worst case, at least when $X$ and $n$ are sufficiently large. Suppose $n = 2k$ for some integer $k \ge 2$, and in every phase of AbstractTrace, the adversary chooses $\ell = m = k+1$ (and therefore $d = 0$) and $(i_0, i_1, \ldots, i_k) = (k+1, k+2, \ldots, 2k, (a \bmod k) - 1)$. In other words, the adversary mimics the strategy described in the previous paragraph in the lower half $x[1\,..\,k]$ of the crossing lengths array, but uses the upper half $x[k+1\,..\,2k]$ to add $k$ additional steps to each phase. The values in $x[k+1\,..\,2k]$ never change; at all times, $x[i] = 1$ for all $i > k$. Thus, the additional steps have very little impact on the growth of the sum $\sum_i x[i]$. A straightforward inductive argument implies that for any integer $r \geq 1$, at the end of $r \cdot (k-1)$ phases, we have $\sum_{i=1}^{k} x[i] < (2^r - 1)k^2 + k < 2^r k^2 - k$ and therefore $\sum_{i=1}^{n} x[i] < 2^r n^2/4$. We conclude that AbstractTrace must execute at least $(n-1) \times \lg(4X/n^2) = \Omega(n \log X)$ phases before $\sum_{i=1}^{n} x[i] = X$. Each phase requires $\Omega(n)$ time.

We leave open the possibility that our analysis is *not* tight for instances that actually arise from tracing normal curves on triangulated surfaces. We conjecture that Lemma 4.2 is still tight in this context, but that Corollary 4.3 is not.

## C Recovering Normal Coordinates

**Lemma C.1.** *Let $\Sigma$ be a surface composed of $n$ triangles, let $\gamma$ be a reduced normal curve in $\Sigma$ with total crossing number $X$, and let $\lambda$ be the union of any subset of components of $\gamma$. Given the street complex $S(T,\gamma)$* ***and its history****, we can compute the normal coordinates of $\lambda$ with respect to $T$ in $O(n^2 \log X)$ time.*

**Proof:** Intuitively, to compute the normal coordinates of $\lambda$, we simply undo the tracing algorithm, by walking backward through the tracing history, maintain the street coordinates of the already-untraced components. When the curve is completely untraced, the streets degenerate to edges, and the street coordinates are the required edge coordinates.

Specifically, we maintain an array $st[1\,..\,n]$ of street coordinates, all initially zero, and a bit $\phi$ that indicates whether we are currently untracing a component of $\lambda$. We consider the phases stored in the history in reverse order. To undo a phase, we update the street coordinates as follows:

```
d ← ⌈ℓ/m⌉ − 1
for j ← 0 to m − 1
    st[i_j] ← st[i_j] + d · (st[a] + φ)
for j ← 0 to (ℓ − 1) mod m
    st[i_j] ← st[i_j] + (st[a] + φ)
```

Here, $a$ is the index of the active street, $\ell$ is the length of the spiral, $m$ is the number of unique directed streets in the spiral, and $i_0, i_1, \ldots, i_{m-1}$ are the indices of the streets in the spiral. (Compare with the AbstractTracing algorithm in Figure 3.) Note that the street coordinates $st[i]$ do not actually change until we start untracing a component of $\lambda$.

Just as in AbstractTracing, we spend $O(n)$ time in each phase of the untracing algorithm; the theorem now follows from Theorem 4.5. □

Even without the complete tracing history, we can untrace any street complex $S(T,\gamma)$ if we also know the crossing lengths of every street. In fact, it is not necessary to follow the tracing algorithm backward; we can untrace the components of $\gamma$ in any order, starting each cycle component at any crossing. Just as we compute the depth of a forward spiral from the street and junction coordinates, we can compute the depth of any backward spiral from the crossing lengths of the $m$ streets that the spiral traverses, also in $O(m)$ time. Analysis similar to Theorem 4.5 implies that this more flexible untracing algorithm also runs in $O(n^2 \log X)$ time. Details will appear in the full paper.

## D Tracing Geodesics

### D.1 Annular Ray Shooting

Let $P$ be a triangulated simple polygon with $n$ vertices; let $e_0$ and $e_1$ be two edges of $p$ of equal length; and let $\rho$ be a ray starting on $e_0$ and pointing into the interior of $P$. Finally, let $A$ be the piecewise-linear annulus obtained from $P$ by identifying the edges $e_0$ and $e_1$.

Our annular ray-shooting algorithm considers, but does not actually construct, finite portions of the *universal cover* of $A$. Let $\tau: \mathbb{R}^2 \to \mathbb{R}^2$ denote the rigid motion (either translation or rotation) that maps $e_0$ onto $e_1$. For each integer $i > 0$, let $\tau^i = \tau \circ \tau^{i-1}$, and let $e_i = \tau(e_{i-1}) = \tau^i(e_0)$. Similarly, let $P_0 = P$, and for any integer $i > 0$, let $P_i = \tau(P_{i-1}) = \tau^i(P_0)$; by construction, the segment $e_i$ is an edge of both $P_{i-1}$ and $P_i$. Finally, for any non-negative integer $t$, let $P_{<t}$ denote the region obtained by gluing the polygons $P_0, P_1, \ldots, P_{t-1}$ along corresponding edges $e_j$. Although $P_{<t}$ is a topological disk, it is not in general a simple polygon, as different copies of $P$ may overlap.

We say that $\boldsymbol{\rho}$ ***traverses*** $\boldsymbol{P_{<t}}$ if and only if $\rho$ intersects all $e_j$, $0 \le j < t$, but no other edge of $P_{<t}$. The output $t^*$ of the annular ray-shooting problem is the maximum of all integers $t$ such that $\rho$ traverses $P_{<t}$. Our algorithm finds $t^*$ using a standard unbounded search strategy due to Bentley and Yao [3]. We emphasize that our algorithm does not actually construct $P_{<t^*}$ or any significant portion thereof, but instead computes on the fly only the vertices and edges required for the search.

To simplify our presentation, we assume that the edge $e_0$ is vertical, the polygon $P$ lies locally to the right of $e_0$, and that the transformation $\tau$ is either a translation or a counterclockwise rotation by angle $\theta > 0$. Thus, $P_{<k}$ grows to the right and possibly curves upward as the parameter $k$ increases. We also assume that $t^* \ge 1$, since it is easy to solve the ray shooting problem in $O(n)$ time otherwise. (Also, our geodesic tracing algorithm performs an annular ray-shooting query only after a complete traversal of the sequence of $O(n)$ streets and junctions that defines the annulus.)

The *funnel* of $P$ is the union of all shortest paths from points in $e_0$ to points in $e_1$. The funnel is bounded by the shortest paths in $P$ between corresponding endpoints of the two portals; we call these shortest paths $A$ ("above") and $B$ ("below"). Our assumption that $t^* \ge 1$ implies that $\rho$ reaches $e_1$ without intersecting either $A$ or $B$; thus, $A$ and $B$ are disjoint convex chains [2,43]. Because $P$ is already triangulated, it is straightforward to construct its funnel in $O(n)$ time [6,20,43]. From now on, we assume implicitly that $P$ is equal to its funnel.[4]

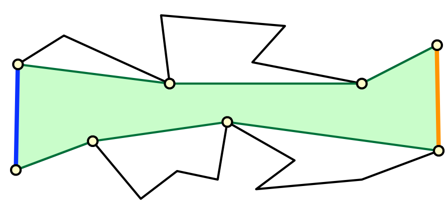

**Figure 4.** The funnel between the portals of $P$.

Let $a_0, a_1, \ldots, a_\alpha$ denote the vertices of $A$ in order from left to right, and let $b_0, b_1, \ldots, b_\beta$ denote the vertices of $B$ in order from left to right. Thus $e_0 = a_0 b_0$ and $e_1 = a_\alpha b_\beta$; we also have $\alpha + \beta \le n + 2$. For any indices $i$ and $j$, let $a_{i,j} = \tau^j(a_i)$ denote the vertex of $P_j$ corresponding to $a_i$, and let $b_{i,j} = \tau^j(b_i)$ denote the vertex of $P_j$ corresponding to $b_i$. In particular, we have $a_{0,j} = a_{\alpha,j-1}$ and $b_{0,j} = b_{\beta,j-1}$ for every positive integer $j$.

We define two families of segments that connect adjacent copies of $A$ and $B$. For any index $j$, let $c_j$ ("the $j$th ceiling") denote the lower common tangent of $A_{j-1}$ and $A_j$. By symmetry, there are indices $l$ and $r$ such that $c_j = a_{l,j-1} a_{r,j}$ for every index $j$. Similarly, let $f_j$ ("the $j$th floor") denote the upper common tangent of $B_{j-1}$ and $B_j$. By symmetry, there are indices $p$ and $q$ such that $f_j = b_{p,j-1} b_{q,j}$ for every index $j$. Because $\tau$ is either a translation or a counterclockwise rotation, we have $r \le l$ and $p \le q$; thus, segments $c_j$ and $c_{j+1}$ are interior-disjoint, but segments $f_j$ and $f_{j+1}$ intersect inside $P_j$. See Figure 5. In the degenerate case where $\tau$ is a translation, we have $l = r$ and $p = q$.

Our algorithm applies a standard unbounded search strategy of Bentley and Yao [3] to find the smallest value of $t$ that satisfies at least one of the following conditions:

$(A)$ Segment $c_t$ contains a point below $\rho$.

[4] This assumption actually implies that $P_{<k}$ is a simple polygon for any value of $k$ that occurs in the algorithm.

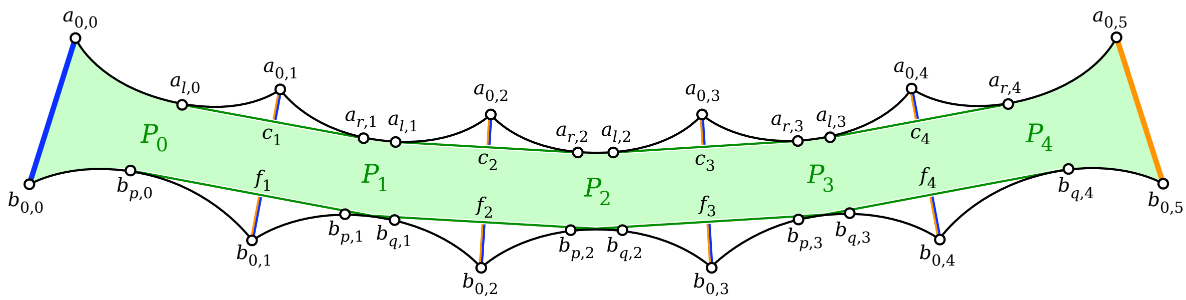


**Figure 5.** Floors and ceilings in the universal cover of the annulus.

$(A')$ The slope of segment $c_t$ is larger than the slope of $\rho$.

$(B)$ Segment $f_t$ contains a point above $\rho$.

If $t > t^* + 1$, then at least one of these three conditions must be true. On the other hand, if $t < t^*$, then conditions $(A)$ and $(B)$ do not hold, and condition $(A')$ holds only if $\rho$ does not intersect *any* upper chain $A_j$. In particular, condition $(A')$ is necessary to detect the situation where $\rho$ crosses some upper chain $A_t$ twice between the ceiling endpoints $a_{r,t}$ and $a_{l,t}$.

Starting with the estimate $t = 1$, our algorithm repeatedly doubles $t$ until at least one of these three conditions is satisfied, and then performs a binary search for the critical value of $t$. When the unbounded search ends, there are three cases to consider, depending on which conditions are satisfied.

$(A)$ Suppose $c_t$ contains a point below $\rho$ but $c_{t-1}$ does not. Then $\rho$ must hit either $A_{t-1}$ or $A_t$; that is, either $t^* = t - 1$ or $t^* = t$. We can distinguish between these two cases in $O(n)$ time by brute force.

$(A')$ Suppose $c_{t-1}$ and $c_t$ both lie above $\rho$, and the slope of $\rho$ lies between the slopes of $c_{t-1}$ and $c_t$. Then either $\rho$ hits $A_{t-1}$, or $\rho$ does not intersect any upper chain $A_j$. Again, we can distinguish between these two cases in $O(n)$ time by brute force. If $\rho$ hits $A_{t-1}$, we return $t^* = t - 1$. Otherwise, we perform a second unbounded search to find the first chain $B_{t^*}$ hit by $\rho$.

$(B)$ Finally, if $f_t$ contains a point above $\rho$ but $f_{t-1}$ does not, then $\rho$ must hit either $B_{t-1}$ or $B_t$. Again, we can distinguish between these two cases in $O(n)$ time by brute force.

If the critical value of $t$ satisfies more than one termination condition, we perform the relevant computation for all satisfied conditions and return the smallest value found.

An important subtlety in the algorithm is that computing the coordinates of $c_t$ or $f_t$ from scratch requires $\Theta(\log t)$ time. However, by computing an array of $O(\log t^*)$ transformations of the form $\tau^{2^i}$ during the doubling phase of the unbounded search, we can compute the coordinates of the appropriate segments $c_t$ or $f_t$ in $O(1)$ time in each iteration of the search.

To summarize: Our algorithm spends $O(n)$ time preprocessing the polygon $P$; performs an unbounded binary search to approximate $t^*$ up to a small additive constant, spending $O(1)$ time per iteration; and then spends $O(n)$ postprocessing time to find the precise value of $t^*$.

**Lemma 6.1.** *The annular ray-shooting problem in a piecewise-linear annulus with $n$ vertices can be solved in $O(n + \log t^*)$ time and space, where $t^*$ is the output value.*

### D.2 Self-Collisions and Directed Streets

The reduction from tracing geodesic spirals to annular ray-shooting is not quite as straightforward as it may appear at first glance. First, we actually require the following minor modification: Given a polygon $P$ with equal-length portal edges and a ray $\rho$, we need to determine how many times $\rho$ crosses the portal before hitting either a non-portal edge of $P$ *or some earlier point in* $\rho$. Equivalently, we need to compute the largest integer $t$ such that $\rho$ does not intersect $A_{<t}$ or $B_{<t}$ *and* does not cross the ray $\tau^{-1}(\rho)$ inside $P_{<t}$. To solve this modified problem, we add a fourth termination condition to the unbounded search:

$(C)$ The point $\rho \cap \tau^{-1}(\rho)$ lies inside the convex quadrilateral $\operatorname{conv}\{e_{t-1}, e_t\}$.

Adding this condition increase the running time of the unbounded search algorithm by only a small constant factor.

We must also consider the possibility that the spiral traverses the same street in both directions; in this case, the simple self-intersection test described in the previous paragraph may fail. To avoid this possibility, we partition each street into two ***lanes*** by drawing a ***median*** geodesic segment between vertices of the triangular faces at each end of the street, as shown in Figure 6. The median segments also partition the junctions into smaller convex polygons of constant complexity. Straightforward case analysis implies that the geodesic crosses the median of a street only at the very end of a phase. It follows that during any phase, the growing geodesic traverses each lane in only one direction. Specifically, in a right-turning phase, the growing geodesic always traverses streets so that its median lies to the left; more informally, the geodesic "drives on the right". Similarly, during left-turning phases, the growing geodesic "drives on the left".

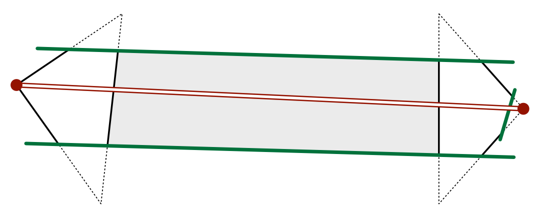

**Figure 6.** Splitting a geodesic street into two one-way lanes

Thus, to compute the depth of a spiral, we can unfold the $m$ relevant *lanes* and *junction fragments* into a polygon $P$ with complexity $O(m)$, and then invoke our modified annular ray-shooting algorithm.

**Lemma D.1.** *The depth $d$ of a geodesic spiral through $m$ distinct directed streets can be computed in $O(m + \log d)$ time.*

Corollary 4.3 now immediately implies that our geodesic tracing algorithm runs in $O(n^2 \log X)$ time, where $X$ is the number of times the geodesic crosses an edge of the original triangulation. The output of the tracing algorithm is the final street complex $S(T, \gamma)$, along with the location of the first self-intersection point $x$, in the local coordinates of the face of $S(T, \gamma)$ that contains $x$. If necessary, we can locate $x$ in the original triangulation $T$ in $O(n^2 \log X)$ additional time by running the tracing algorithm backward and maintaining the location of $x$ is the devolving street complex.

**Theorem 6.2.** *Let $\Sigma$ be a triangulated piecewise-linear surface with $n$ triangles. Given the starting point and direction of a geodesic $\gamma$ in $\Sigma$, we can compute the first self-intersection point in $\gamma$ in $O(n^2 \log X)$ time, where $X$ is the number of edges $\gamma$ crosses before it self-intersects.*