On the Topology of Walkable Environments

Benjamin Burton[1], Arne Hillebrand[2], Maarten Löffler[2], Schleimer, Saul[3], Dylan Thurston[4], Stephan Tillmann[5], and Wouter van Toll[2]

1 **University of Queensland, Australia**
2 **Dept. of Inform. and Computing Sciences, Utrecht University, the Netherlands**
3 **University of Warwick - Coventry, Great Britain**
4 **Indiana University - Bloomington, United States of America**
5 **University of Sydney, Australia**

Abstract

Motivated by motion planning applications, we study 2-dimensional surfaces embedded in 3-dimensional space with the property that their vertical projection is an immersion. We provide bounds on the complexity of a triangulation of such a surface, given that the projection of the boundary is a polygon with m segments. We then show how these bounds lead to efficient algorithm to compute such a triangulation. Finally, we relate our result to concrete motion planning setting and review related open questions.

1 Introduction

Simulations and computer games are often occupied by virtual characters that need to move autonomously through a virtual environment. This asks for efficient data structures and algorithms for path planning and crowd simulation. An environment such as a multi-storey building is three-dimensional, but the characters are restricted to surfaces on which they can walk. Therefore, while these surfaces are embedded in $\mathbb{R}^3$, they are (in some ways) locally similar to $\mathbb{R}^2$. They form an interesting class of environments that we call *walkable environments* (WEs).

For the purpose of path planning, it is important to automatically obtain an efficient representation of a walkable environment [3, 4, 9, 10]. In particular, it is desirable to subdivide a walkable environment into surfaces that can each be projected onto $\mathbb{R}^2$ without overlap. We refer to such a decomposition as a *multi-layered environment* (MLE). An individual surface within an MLE is called a *layer*, and the 'cuts' made for the decomposition are called *connections*. The main advantage of an MLE is that each separate layer can be treated as a 2D component, which allows the extension of 2D data structures [9].

Technically, a connection is a curve along the WE between two boundary vertices of the WE. For the application's data structures and algorithms, it is also important that each connection is a straight line segment when projected onto $\mathbb{R}^2$. Furthermore, it is often desirable to obtain an MLE with a small number of connections because this number influences the complexity of various algorithms. To understand these demands better, we first need to study the topological properties of a walkable environment.

Contributions In this paper, we study the topology of walkable environments, and we present an algorithm that triangulates a WE. Suppose a walkable environment $\mathcal{W}$ is given as a triangulated surface with n vertices and m *boundary* vertices. (We will give a more precise definition in Section 2.) We prove that any triangulation of $\mathcal{W}$ has $O(m)$ diagonals, and we present an algorithm that computes a triangulation of $\mathcal{W}$ in $O(n + m \log m)$ time, in such a way that each diagonal is a line segment when projected onto $\mathbb{R}^2$.

At this point, an alert reader may wonder why we should wish to triangulate a surface that is already triangulated. The difference is that the input triangulation has linear

34th European Workshop on Computational Geometry, Berlin, Germany, March 21–23, 2018.
This is an extended abstract of a presentation given at EuroCG'18. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

complexity in n, while the output triangulation only has linear complexity in m, which can be much smaller. For example, an environment containing hilly terrains has many interior vertices, which will not be present in our output triangulation. The output triangulation does not have straight edges in $\mathbb{R}^3$, but the projections of its edges are straight segments in $\mathbb{R}^2$. This type of triangulation is interesting for path planning applications because these applications often use projected distances [10], and the diagonals of this triangulation may be good candidates for connections in an MLE.

Related work There are several applications and algorithms that are already using some form of a multi-layered environment. Van Toll et al. [9] use an MLE to create a multi-layered navigation mesh based on the medial axis, which allows for fast path planning queries. Rodriguez and Amato [8] convert a multi-layered environment to a roadmap representation, which they then use to find a strategy for efficiently clearing a building. However, both of these works do not describe how to *obtain* such an MLE from an arbitrary 3D environment.

A popular way to convert a 3D environment to a walkable environment is to approximate the environment by 3D grid cells (*voxels*), in which each voxel is marked as walkable or non-walkable. Voxelization typically uses the graphics card. Several methods use this concept as the first step in a pipeline for computing a navigation mesh [1, 6, 7].

Converting a walkable environment to a *multi-layered environment*, preferably with a small number of cuts, is a separate problem. Hillebrand et al. [3, 4] model this as a graph problem. They prove that obtaining an MLE with a minimum number of cuts is NP-hard, but that good MLEs can often be obtained using heuristics. However, these cuts are not always suitable as connections since they are restricted to edges of the input triangulation.

Outline To obtain our results, we first introduce some subtly different concepts in Section 2 that capture the special properties of the surfaces we encounter. In Section 3, we analyse the possible values of the genus of these surfaces. Then, in Section 4, we show how an adaptation of the polygon decomposition algorithm by Lee and Preparata [5] can be used to triangulate our surfaces. Finally, in Section 5, we discuss how these results relate to the problem of finding a good MLE, and we pose the main open question in this area.

2 Definitions

Sloths and walkable environments. We define a *sloth* to be a compact surface Σ continuously embedded in $\mathbb{R}^3$ so that the vertical projection of Σ to $\mathbb{R}^2$ is an immersion with a polygonal boundary consisting of m line segments (or, equivalently, m vertices). That is, the vertical direction is transverse to Σ. We say that a sloth is *realistic* if the turning angle (projected to $\mathbb{R}^2$) around any boundary vertex is at most 360 degrees (aka 2π).

We define a *walkable environment* (WE) as a geometric representation of a realistic sloth by a set of connected triangles in $\mathbb{R}^3$. By definition, a WE has the following properties:

- The angle between the normal of each triangle and the normal of the ground plane is less than 90 degrees.
- The minimal vertical distance between any two triangles in the WE is non-zero, with the exception of shared edges and vertices.

Let n be the total number of vertices in the WE. Note that n can be much larger than the number of *boundary* vertices m. Thus, a WE is a particular type of sloth represented by triangles. Some parts of this paper apply to sloths in general; other parts apply specifically to realistic sloths or to WEs because they rely on extra properties.

Triangulations of sloths. We define a *topological triangulation* of a sloth Σ to be a subdivision of Σ into topological triangles whose vertices coincide with the vertices of Σ; that is, we add arbitrary arcs on the surface that connect pairs of vertices that were not connected before, and the arcs do not intersect each other except possibly at endpoints.

We define a *geometric triangulation* of a sloth Σ to be a topological triangulation with the additional restriction that each edge, when projected to $\mathbb{R}^2$, is a straight line segment.

3 Bounds on the genus of sloths

In this section, we provide bounds on the potential genus of sloths, expressed in the complexity of their boundaries. Specifically, for a sloth Σ with m boundary vertices, we are interested in possible values of $\text{genus}(\Sigma)$ as a linear function of m.

Gauss–Bonnet Recall that the *Euler characteristic* of a compact, orientable surface Σ is:

$$\chi(\Sigma) = 2 - 2\text{genus}(\Sigma) - \#\partial\Sigma,$$

where $\#\partial\Sigma$ is the number of connected components in the boundary of Σ. The following result relates the topology of the surface to its geometry. We use the special case when the curvature vanishes on the interior and the curvature of the boundary is zero except at the polygonal corners.

▶ **Theorem 3.1** (Gauss–Bonnet, flat version)**.** *Let Σ be a surface with a locally Euclidean metric and polygonal boundary, with corners at c_i with interior angle θ_i. Then*

$$\chi(\Sigma) = \frac{1}{2\pi}\sum_i (\pi - \theta_i).$$

Here, $\pi - \theta_i$ should be thought of as the bending angle at c_i: zero if there is no actual corner, positive if the corner is convex as on the boundary of a convex polygon in the plane, and negative if the corner is concave.

Application to sloths We use Theorem 3.1 to prove an upper bound of the genus of a sloth.

▶ **Theorem 3.2.** *Let Σ be a surface with boundary and let $f\colon \Sigma \to \mathbb{R}^2$ be an immersion on the interior of Σ so that $f(\partial\Sigma)$ is a polygonal path with m line segments. Then* $\text{genus}(\Sigma) \le m(m+1)/4$. *Furthermore, there are examples coming from embeddings in $\mathbb{R}^3$ with* $\text{genus}(\Sigma) = (m/8 - 1)^2$.

Proof. The examples achieving quadratic genus growth are "parking garages" $P_{k,l}$, as shown in Figure 1:

- take k parallel rectangular sheets;
- cut out l slits from each sheet (stacked on top of each other); and
- rejoin across the slits, shifting down one level as you go.

We can apply Theorem 3.1 (the Gauss–Bonnet theorem) to the metric on Σ coming from the map to $\mathbb{R}^2$. Here, $\pi - \theta_i$ should be thought of as the bending angle at c_i: zero if there is no actual corner, positive if the corner is convex as on the boundary of a convex polygon in the plane, and negative if the corner is concave. Some of the corners in $P_{k,l}$ are very concave, with a total internal angle of approximately $2k\pi$. The result of this computation is that $\text{genus}(P_{k,l}) = (k-1)(l-1)$. Furthermore, $P_{k,l}$ can be realized with a polygonal boundary with $4k + 4l$ corners.

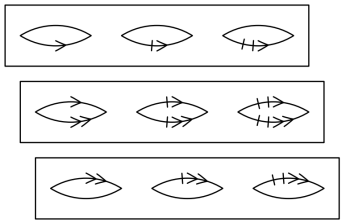

Figure 1 A parking garage, shown for $k = 3$ and $l = 3$. Edges with corresponding marking are glued; this can be achieved by stacking the sheets in 3 dimensions and attaching connecting ramps.

For the upper bounds on genus, we again apply Theorem 3.1 and give an upper bound on the interior angles θ_i. To do this, we first bound the total multiplicity in any region, the degree by which it is covered by Σ. The multiplicity at a point $x \in \mathbb{R}^2$ can be computed by sending a ray out to infinity in either direction from x, and so is at most $m/2$. The angle θ_i at a corner c_i is bounded by 2π times the multiplicity in any adjoining region. This yields the stated upper bound on genus. ◀

The last observation leads to the following theorem:

▶ **Theorem 3.3.** *For a realistic sloth Σ with m boundary vertices,* genus(Σ) *is $O(m)$.*

4 Geometric triangulations of sloths

We are interested in geometric triangulations of walkable environments because these can be useful representations for the purpose of path planning and crowd simulation. In this section, we first discuss triangulations of sloths in general, and then we present an algorithm for triangulating a WE.

▶ **Observation 4.1.** *If a sloth has m vertices, b boundary components, and genus g, then every (topological or geometric) triangulation has $t = 4g + b + m - 4$ triangles.*

▶ **Lemma 4.2.** *Every sloth has at least one geometric triangulation.*

Proof. We show that whenever Σ is not yet triangulated, we can always find a diagonal that splits Σ into two valid sloths, which we can then recursively triangulate again.

Consider a convex vertex v. A *surface ray* is a curve in $\mathbb{R}^3$ that lies completely inside Σ, but whose vertical projection to $\mathbb{R}^2$ is a straight line segment. We shoot a surface ray r from v over the sloth in an arbitrary direction; note that, once we fix the direction, the ray is unique except potentially when it passes through a vertex of Σ. If r does pass through a vertex of Σ, we are happy. If r does not pass through a vertex of Σ, we can continuously rotate r about v until it does; note we can rotate r in two directions until it coincides with either of the incident boundary edges of Σ at v. We distinguish two cases.

1. The ray r passes through a vertex x of Σ, and x is not a neighbour of v on the boundary of Σ. By construction, there is a unobstructed path on Σ from v to x which projects to a line segment. We add edge vx to our triangulation, and recursively triangulate the one or two smaller sloths.
2. When sweeping in both directions, the ray r hit no vertices other than u and w, the neighbours of v on the boundary of Σ. Consider the edge uw, which projects to a straight segment. If uw crosses any existing boundary edge e of Σ, there must be either

be a vertex x contained in the triangle uvw (on Σ), or the edge e crosses (passes over or under) uv or uw. In the first case, we should have found x when sweeping r. In the second case, there must be another edge e' on Σ that e crosses, in order to move over or under uv or uw. We now argue similarly about the endpoinds of e': either, there is an endpoint inside uvw, or e' also crosses uv or uw. Eventually, we run out of edges. Contradiction. We add edge uw to our triangulation, and recursively triangulate the smaller sloth. ◀

Combined with Theorem 3.3, we can conclude the following:

- Every sloth with m vertices has a geometric triangulation with $O(m^2)$ triangles.
- Every sloth with m vertices, whose vertices all have a constant maximum turning angle, has a geometric triangulation with $O(m)$ triangles. By definition, this also holds for every *realistic* sloth, and for every walkable environment with m boundary vertices.
- All triangulations of a given sloth have the same number of triangles and diagonals.

We now describe an algorithm that computes a geometric triangulation of a WE with n vertices and m boundary vertices in $O(n + m \log m)$ time. It applies only to WEs because it relies on the maximum turning angle of realistic sloths around the boundary vertices.

▶ **Lemma 4.3.** *A geometric triangulation of a walkable environment $\mathcal{W}$ with n vertices and m boundary vertices can be computed in $O(n + m \log m)$ time.*

Proof sketch. The idea is to split $\mathcal{W}$ into y-monotone pieces via a 2D plane sweep over all boundary points sorted by y-coordinate, similar to the algorithm by Lee and Preparata [5] for splitting a 2D polygon (with or without holes) into y-monotone pieces. However, several complications arise due to the nature of WEs.

First, we need to obtain a boundary representation of $\mathcal{W}$ such that we can move from any boundary vertex to any adjacent boundary vertex in constant time. This can be done in $O(n)$ time if $\mathcal{W}$ is given as a DCEL.

Using this boundary representation as input, we sweep a plane H parallel to the x- and z-axes, starting at $y = -\infty$, and we maintain a set of curves on H where it intersects $\mathcal{W}$. We may encounter four types of events, reminiscent of the Reeb graph: start events (where a new curve appears), split events (where a curve splits into two curves), merge events (where two curves merge into a single curve), and end events (where a curve disappears).

Since the projection in the z-direction of the boundary of $\mathcal{W}$ is polygonal, and the sweep plane is parallel to the z-axis, all events occur at vertices of $\mathcal{W}$. Furthermore, because $\mathcal{W}$ is a realistic sloth, all events are indeed related to at most two curves. (This would not be the case in the non-realistic parking garage from Figure 1, where a single vertex can induce many curves). Hence, we can detect and sort all events in $O(m \log m)$ time.

Now, we process the events maintaining the latest merge vertex, as in the classic algorithm in [5]. We provide a complete description of the algorithm in the full version of this paper.

After applying the algorithm, we have subdivided $\mathcal{W}$ into a set of y-monotone pieces. Observe that each y-monotone piece P_i is a simple polygon when projected to $\mathbb{R}^2$. Therefore, each P_i (with m_i boundary vertices) can be triangulated in $O(m_i)$ time, and the combined time for triangulating all parts is $O(m)$. The result is a triangulation of the WE.

Combined with the pre-processing time for obtaining a boundary representation, this proves the lemma. ◀

5 Layers and Connections

The results from the previous sections form a first step towards the practical decomposition of a walkable environment into layers for path planning purposes. Recall that we are interested in obtaining a *multi-layered environment* (MLE) with a minimum number of connections, where each connection is a straight line segment when projected onto $\mathbb{R}^2$.

Expressed in terms of sloths, we define a sloth to be *flat* if its projection to $\mathbb{R}^2$ is a polygon without self-intersections; that is, no two points on the sloth project to the same point in $\mathbb{R}^2$. The generalized version of our problem is to decompose a sloth Σ into flat sloths, by cutting Σ in such a way that the endpoints of each cut are boundary vertices of Σ, and each cut projects to a line segment in $\mathbb{R}^2$.

The triangulation algorithm from Section 4 first decomposes Σ into y-monotone pieces (and then into triangles). This subdivision into y-monotone pieces already induces a valid MLE. However, better subdivisions (with fewer connections) might exist.

Let $C^*(\Sigma)$ be the minimum number of connections required for subdividing a sloth Σ into layers. We conclude by stating the following open question:

▶ Question 1. Given a sloth Σ represented as a WE with n internal vertices and m boundary vertices, (how) can we subdivide it into a multi-layered environment with $C^*(\Sigma)$ connections, or with a number of connections that approximates $C^*(\Sigma)$?

Acknowledgements. This research was initiated at Dagstuhl Seminar 17072. We would like to thank all participants of the seminar for their fruitful discussions. M.L was partially supported by the Netherlands Organisation for Scientific Research (NWO) through project no. 614.001.504.

References

1 L. Deusdado, A.R. Fernandes, and O. Belo, *Path planning for complex 3D multilevel environments*. Proc. 24th Spring Conf. on Computer Graphics, pp. 187–194 (2008).

2 J. Guo, F. Hüffner, E. Kenar, R. Niedermeijer, and J. Uhlmann, *Complexity and exact algorithms for multicut*, Proc. Current Trends in Theory and Practice of Computer Science, SOFSEM 3831, pp. 137–147 (2006).

3 A. Hillebrand, M. van den Akker, R. Geraerts, and H. Hoogeveen, *Performing multicut on walkable environments*, Proc. 10th Int. Conf. on Combinatorial Optimization and Applications, pp. 311–325 (2016).

4 A. Hillebrand, M. van den Akker, R. Geraerts, and H. Hoogeveen, *Separating a walkable environment into layers*, Proc. 9th Int. Conf. on Motion in Games, pp. 101–106 (2016).

5 D.T. Lee and F.P. Preparata, *Location of a point in a planar subdivision and its applications*, SIAM Journal of Computing, 6:594–606 (1977).

6 R. Oliva and N. Pelechano, *NEOGEN: Near optimal generator of navigation meshes for 3D multi-layered environments*. Computers & Graphics 37(5), pp. 403–412 (2013).

7 J. Pettré, J.P. Laumond, and D. Thalmann, *A navigation graph for real-time crowd animation on multilayered and uneven terrain*, Proc. First Int. W. Cr. Sim., pp. 81–89 (2005).

8 S. Rodriguez and N.M. Amato, *Roadmap-based level clearing of buildings*, Lecture Notes in Computer Science, vol. 7060 LNCS, pp. 340–352 (2011).

9 W. van Toll, A. F. Cook IV, and R. Geraerts, *Navigation meshes for realistic multi-layered environments*, Proc. Int. Conf. on Intelligent Robots and Systems, pp. 3526–3532 (2011).

10 W. van Toll, R. Triesscheijn, M. Kallmann, R. Oliva, N. Pelechano, J. Pettré, and R. Geraerts, *A Comparative Study of Navigation Meshes*, Proc. 9th Int. Conf. on Motion in Games, pp. 91–100 (2016).