3-1-2016

Geodesic Universal Molecules

John C. Bowers
James Madison University

Ileana Streinu
Smith College, istreinu@smith.edu

Math.Comput.Sci. (2016) 10:115–141
DOI 10.1007/s11786-016-0253-5

Mathematics in Computer Science

Geodesic Universal Molecules

John C. Bowers · Ileana Streinu

Received: 18 June 2015 / Revised: 29 February 2016 / Accepted: 3 March 2016 / Published online: 29 March 2016

Abstract The first phase of TreeMaker, a well-known method for origami design, decomposes a planar polygon (the "paper") into regions. If some region is not convex, TreeMaker indicates it with an error message and stops. Otherwise, a second phases is invoked which computes a crease pattern called a "universal molecule". In this paper we introduce and study *geodesic universal molecules,* which also work with non-convex polygons and thus extend the applicability of the TreeMaker method. We characterize the family of disk-like surfaces, crease patterns and folded states produced by our generalized algorithm. They include non-convex polygons drawn on the surface of an intrinsically flat piecewise-linear surface which have self-overlap when laid open flat, as well as surfaces with negative curvature at a boundary vertex.

Keywords Algorithmic origami · Planar subdivision · Metric tree · Non-convex polygon

Mathematics Subject Classification 68U05 (Computer graphics; computational geometry)

1 Introduction

Lang's TreeMaker method [12] is a seminal work in the field of computational origami. Given a square sheet of paper and a metric tree T, it computes a crease pattern that can be folded into a tree-like 3D structure projecting onto T, as in Fig. 1. Precise definitions will be given in Sect. 2.

TreeMaker works in two phases. The first phase solves an optimization problem which subdivides the square paper into polygonal regions and the input tree into subtrees compatible (in a sense that will be made precise below) with them. When all of these polygons are convex, TreeMaker fills each of them with a crease pattern called a *universal molecule*. If the first phase produces some non-convex polygon, TreeMaker stops without producing an

We acknowledge support for this work through an NSF graduate fellowship (JB) and NSF Grant CCF-1319366 (IS).

J. C. Bowers
Department of Computer Science, James Madison University, Harrisonburg, VA 22801, USA
e-mail: bowersjc@jmu.edu

I. Streinu (✉)
Department of Computer Science, Smith College, Northampton, MA 01063, USA
e-mail: istreinu@smith.edu

output. *Due to the nature of the optimization problem solved, there is little hope of constraining the first phase to guarantee convexity.*

Our Results In this paper we generalize the universal molecule algorithm to work with non-convex polygons produced by the first phase, thus allowing TreeMaker to proceed to the end in all cases in which the first phase succeeds. *This solves a long standing open question* [10] *concerning Lang's algorithm.*
We rely on our streamlined presentation of Lang's algorithm from [5] and generalize the concepts and terminology introduced there. The main difference is that *geodesic distance* inside the (non-convex) input polygon, rather than Euclidean distance, enforces the relationship of a *geodesic Lang polygon* P_T with a metric tree T. As in [5], we aim at obtaining a *full characterization* of the shapes (geodesic Lang surfaces) produced by this generalized algorithm, as well as of the inputs (geodesic Lang polygons) on which it works.

Basic Concepts Let T be a metric, topologically embedded tree: it has positive weights attached to each arc, and has a defined ordering or *rotation* of the incident arcs at each internal node. See Fig. 2. A polygon P_T is said to be a *doubling polygon* for T if it is metrically and combinatorially equivalent to a right-hand-turn walk around that arcs of T starting from some leaf node. We say that P_T satisfies the *geodesic Lang property* if the geodesic distance (inside the polygon) between any two vertices of P_T is greater than or equal to the corresponding tree distance in T.

The *input* to the geodesic universal molecule algorithm described in this paper is a tree T and a *geodesic Lang polygon* P_T compatible with it. The *output* of the algorithm is a subdivision of the polygon into vertices, edges, and faces (a *crease pattern*) that is intrinsically equivalent to what we call a *geodesic Lang surface* S_T *constructed on* T. This concept, defined in Sect. 5, captures formally what it means, in Lang's approach, for a 3D folded origami shape to be *compatible with* and *project onto* a given metric tree (see Fig. 3).

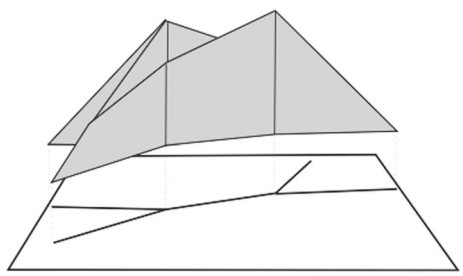

Fig. 1 A folded origami shape produced by Lang's algorithm, projecting onto a metric tree

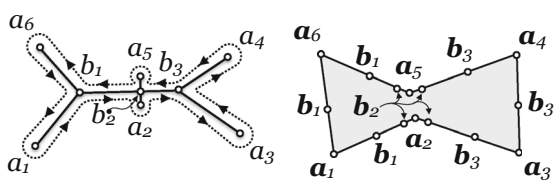

Fig. 2 A tree (*left*) and doubling-polygon (*right*)

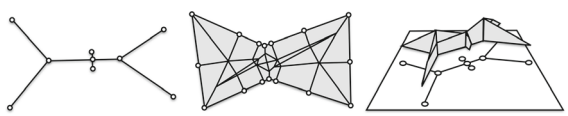

Fig. 3 A tree (*left*), geodesic Lang polygon subdivided by its geodesic universal molecule (*middle*), and an intrinsically equivalent Lang surface (*right*)

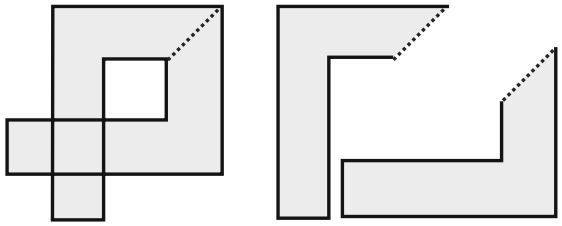

Fig. 4 An intrinsically flat polygon that self-overlaps when (extrinsically) flattened out in the plane (*left*). It is intrinsically a disk, obtained by glueing two simple planar polygons along an edge (*right*)

A *(geodesic) Lang surface* is defined inductively from two types of building block surfaces: extrusion disks and extrusion rings. These are defined with respect to an extrusion process that embeds a *kinetic polygon* into a plane that sweeps upwards; a kinetic polygon is a doubling polygon of a *kinetic tree*, namely a tree in which the leaf nodes are moving with specified speeds along their leaf edges. The trace of the edges of the kinetic polygon define a surface which is topologically either a disk or a ring (annulus). Two gluing operations are defined: one for *extending* a Lang surface by gluing it to a ring, and another for *combining* two Lang surfaces by gluing them along a part of their boundary. Finally, we focus on Lang surfaces with zero-curvature at internal vertices. For the surfaces constructed in this paper we allow the leaf nodes in a kinetic tree to move both inwards and outwards along the leaf edges. This process leads to intrinsically simple, possibly non-convex Lang surfaces. This is in contrast to our previous paper [5], where the leaves moved only inwards and the resulting surface was intrinsically convex.

Our main result can now be stated.

Theorem 1.1 (Main Theorem) *Let P_T be a doubling-polygon for a tree T on a flat, disk-like piecewise-linear surface D. Then a Lang surface S constructed on T and isometric to P_T exists* (*and is unique*) *if and only if P_T is a geodesic Lang polygon for T on D.*

Overview The main concepts needed for the statement of Theorem 1.1 are *geodesic Lang polygons*, the *geodesic Lang property* and *Lang surfaces*: the first two are defined in Sect. 3 and the latter in Sect. 5. In Sect. 4 we introduce the main tool used in the correspondence between geodesic Lang polygons and Lang surfaces, called a *generalized sweep*. The proof of Theorem 1.1 is broken into three parts. In Sect. 5 we prove the necessity of the geodesic Lang property on the Lang surface. The second part (sufficiency) is proven in Sect. 6 by describing an algorithm that takes as input a geodesic Lang polygon and produces as output a universal molecule, as illustrated in Fig. 3. The most intricate part of the paper is the proof of uniqueness (Sect. 7.2).

Origami on Flat Surfaces In addition to non-convex planar polygons, our generalization applies to any polygon bounding a piecewise linear surface that is topologically a disk, has zero curvature, and meets certain constraints on the geodesic distance between pairs of points of the polygon that come from the tree. Such a geodesic Lang polygon should not be thought of as lying in the plane, but rather on an intrinsic surface, on which it is (intrinsically) *simple*. But we remark that an open, flat placement of a geodesic Lang polygon in the Euclidean plane may self-overlap, as illustrated in Fig. 4.

The motivation for generalizing not just to non-convex polygons in the plane, but to these additional cases comes both from mathematical curiosity (*how far can this idea be generalized?*) as well as from recent work on extending origami constructions to paper with negative curvature [3]. Our result represents another family of crease patterns that can be drawn on such specially constructed paper.

Historical Perspective Lang's universal molecule algorithm was first described in [12], in connection with TreeMaker [13]. Further details appear in [10]. Lang polygons and surfaces were used in [5] to formalize the proof of correctness of the universal molecule, which allowed the investigation of further algorithmic properties in [7]. Rigid foldability of universal molecules was studied, starting with the special case considered in [8], where it was shown that when the universal molecule of a convex polygon is identical to the straight skeleton, then a

non-self-intersecting rigid folding motion exists. In contrast, we showed in [6] that for a larger class of universal molecules, the initial crease pattern (the open, flat state of the origami) is completely rigid.

Though TreeMaker appeared over 20 years ago, it remains one of the few "general purpose" origami design algorithms. Another notable method is Tachi's heuristic [14] for computing a crease pattern that when folded realizes an input polyhedral surface. The first phase of TreeMaker is a heuristic for an NP-hard optimization problem [9]. Origami design techniques and rigorous formalization and mathematical analysis are in increased demand for applications such as designing deployable structures [17,18], modeling rigidity [15,16], and formalization for proper handling of the algebraic calculations needed to do exact folding [11].

2 Preliminaries

2.1 Piecewise Linear Metric Surfaces

Our primary objects of interest, both Lang polygons and Lang surfaces, are *piecewise linear metric surfaces* (shortly referred to from now on as *surfaces*) obtained by gluing flat, polygonal faces together along whole edges. A *realization* of a surface is an *isometric map* taking each vertex to a point in $\mathbf{R}^3$, each edge to a straight-line segment, and each face to a flat polygon in $\mathbf{R}^3$ such that the edges and faces maintain their metric properties (size and shape).

Intrinsic Vs. Extrinsic Properties Properties of a surface that are true in any realization are *intrinsic*, while those that depend on a particular realization are *extrinsic*. This distinction is particularly important for our purposes, because two different foldings of the same origami crease pattern are intrinsically the same surface but differ in their extrinsic properties (such as the dihedral or "folding" angle between faces). Showing that a surface is a folding of another amounts to showing that the two surfaces only differ extrinsically. An example of an extrinsic property is the *dihedral angle* between two faces at an edge. Important intrinsic properties include orientation, topology, Gaussian curvature and metric. In this paper, we work only with connected surfaces that:

- Are *orientable*.
- Have the *topology* of a disk (are *disk-like*) or of an annulus (are *ring-like*). Hence, each edge is either incident to exactly one face (a *boundary edge*), or to two faces (an *interior edge*).
- Have a finite number of vertices of positive *(intrinsic) curvature* (defined in the next paragraph).
- Have a metric given by the *geodesic distance* between two points on the surface (defined below).

Curvature Since our surfaces are piecewise linear, the curvature is concentrated at the vertices. A vertex has a *face angle* in each of its incident faces, and its *angle sum* is the sum over all its face angles. The Gaussian *curvature* at a vertex is defined as 2π minus its angle sum. If every internal vertex of a surface has zero curvature then the surface is (intrinsically) *flat*, which does not require that it be realized in a single plane. A realization of a flat surface in which the dihedral angles at all interior edges are equal to π is an *open, flat realization*. Hence both the initial crease pattern drawn on the paper and the final folding of the origami are (intrinsically) flat, but only the first is in an open, flat realization. If for a given surface there exists an open, flat realization, then we say that the surface is *open flattenable*. Open flattenability implies that the surface is (intrinsically) flat. The converse is true for all disk-like surfaces, but not for all ring-like surfaces. For instance, if one removes the top and bottom face from a cube, the resulting ring-like surface is flat (its curvature is zero everywhere), but it is not open flattenable, since it has no open, flat realization. See also [2] for a related problem.

Geodesic Distances and Visible Pairs Given a surface S, the *geodesic distance* between two points p and q, denoted $d_S(p, q)$, is the length of the shortest path between them, called the *geodesic path*. On a piecewise linear surface, this path is a polygonal chain which is unique when the surface is a disk. If the geodesic path between two points p and q is (intrinsically) straight we say that (p, q) is a *visible pair*. Note that the geodesic distance $d_S(p, q)$ satisfies the usual triangle inequality: for all p, q, r, $d_S(p, q) \leq d_S(p, r) + d_S(r, q)$.

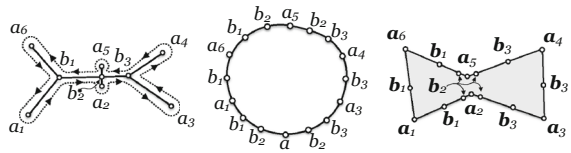

Fig. 5 A tree (*left*), doubling cycle (*middle*), and doubling-polygon (*right*)

Polygons We use this term to denote a polygon drawn on some underlying intrinsic piecewise linear zero-curvature disk-like surface S, rather than just in the plane. When it is simple, i.e. it does not self-touch or self-cross on the intrinsic surface, then we view it together with the piece of the surface that it bounds; in other words, a simple polygon on S is itself a disk-like surface. We reinforce the remark that if we flatten out S into the plane, the polygon may self-overlap even though it is simple on S (Fig. 4). A vertex of a polygon is said to be *convex* (resp. *reflex*, *marker*) if its intrinsic angle sum (i.e. of the vertex angles interior to the polygon surface) is less than (resp. greater than, equal to) π.

2.2 Metric Trees, Metric Doubling Cycles, and Doubling Polygons

A *metric tree* (T, w) is a tree T together with a weight function w that maps each arc[1] of T to a positive weight or *length*. We assume that a cyclic ordering, or rotation, is given for the incident arcs at each node.[2] The *metric doubling cycle* for T is the pair (C_T, w) where C_T is the cycle given by starting at any leaf node and listing the nodes encountered by walking around T while respecting the ordering of incident arcs and w maps each edge of C_T to the length of its corresponding tree arc. See Fig. 5. In such a walk, each edge is traversed once in each direction, and each vertex is visited a number of times equal to its degree. A *doubling polygon* P_T is a polygon that is combinatorially and metrically a doubling cycle for a tree T.

Notations and Conventions It is convenient to separate the n leaf nodes and m internal nodes of a tree T into two sets $A = (a_1, \ldots, a_n)$ and $B = (b_1, \ldots, b_m)$, respectively. In order to make clear the correspondences between a tree T and a doubling polygon P_T, we use **bold face** to denote vertices of the polygon and *italics* to denote corresponding nodes in the tree. For instance, the vertex $\mathbf{a}$ (resp. edge $\mathbf{ab}$) in the polygon P_T corresponds to the leaf node a (resp. leaf arc ab) in the tree T. Without loss of generality, we assume that a walk around the boundary of a polygon is in the direction that keeps the interior of the polygon to the left (as in Fig. 5), and refer to it as a *counter-clockwise* (*ccw*) *walk*. Given a geodesic path from a vertex a to a vertex b of the polygon, we can refer to the part of the polygon (surface and boundary) lying to the left or to the right of the path. In the case of a doubling polygon, this sidedness transfers to the underlying tree and allows us to refer to the left and right subtrees, relative to a path in the tree.

Splitting Trees, Cycles, and Polygons Given a tree T and two leaf nodes a_i and a_j, the *splitting operation* (see Fig. 6) returns two trees T_1 and T_2 corresponding to the part of the tree to the left of (and including) the path from a_i to a_j in T, and the part to the right (resp). To split a doubling cycle C_T between a_i and a_j, we first split C_T into two open chains C_1 and C_2, one from a_i to a_j and the other from a_j back to a_i. We then close each chain using a copy of the path from a_i to a_j in T. The chains C_1 and C_2 are then doubling cycles for T_1 and T_2 (resp). The operation can be naturally extended to doubling polygons. However, for reasons that will become clear in Sects. 3 and 4, we enforce a *stronger condition: in a doubling polygon P_T we allow the splitting operation only if* $(\mathbf{a}_i, \mathbf{a}_j)$ *is a visible pair, and if the geodesic distance from* $\mathbf{a}_i$ *to* $\mathbf{a}_j$ *is equal to the tree distance* $d_T(a_i, a_j)$. The split in the

[1] To avoid confusion, we use the terms *node* and *arc* to refer to the elements of a tree, and *vertex* and *edge* to refer to the elements of a polygon or embedded straight-line graph.

[2] Such a tree is sometimes called a ribbon tree, or a topologically embedded tree.

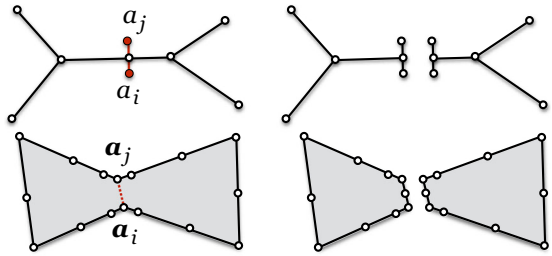

Fig. 6 Splitting a tree and corresponding doubling polygon between a_i and a_j

polygon is performed by introducing a *splitting edge* along the shortest path between $\mathbf{a}_i$ and $\mathbf{a}_j$ and subdividing it into edges so that it is metrically and combinatorially equivalent to the path between a_i and a_j in the tree.

3 Geodesic Lang Polygons

We turn now to the study of the first concept critical to the understanding of our results, namely the *geodesic Lang polygon*. Let S be a piecewise linear zero-curvature disk-like surface and let T be a metric tree with strictly positive weights (i.e. we do not allow degenerate zero-length tree arcs). We consider a doubling polygon P_T for T on a surface S which may be self-touching but not self-crossing, and thus has a well-defined *interior* on the surface. If the polygon is not self-touching, the interior is itself a disk-like surface; otherwise, it may have several disk-like components. A *geodesic* is the shortest path between two points on the polygonal boundary that lies entirely in the closure of the interior of the polygon. To define the main concepts for this section, the geodesic Lang property and geodesic Lang polygons, we first discuss the boundary curvature of polygons and restrict our discussion to a class of polygons we call "well-formed".

Definition 3.1 (*Geodesic Lang property*) Let P_T be a doubling polygon for a tree T on a surface S. We say that (T, P_T) satisfies the *geodesic Lang property* on S if and only if for all pairs of points $(\mathbf{u}, \mathbf{v})$ on the boundary of P_T, their geodesic distance is greater than the corresponding tree distance, i.e. $d_S(\mathbf{u}, \mathbf{v}) \geq d_T(u, v)$.

Negative Boundary Curvature In general we require that the interior angle measure at each vertex of P_T be less than 2π; however, we allow higher angle measures by the following construction. Suppose we have two pairs (T, P_T) and $(T', P'_{T'})$ of trees and doubling polygons (in ccw order) such that there is a side $\mathbf{a}_i\mathbf{a}_j$ in P_T that is *equivalent* to a side $\mathbf{a}'_j\mathbf{a}'_i$ in $P'_{T'}$. By 'equivalent' we mean that the path from a_i to a_j in T has the same number of arcs (with the same lengths) as the path from a'_i to a'_j in T'. We then construct a new doubling polygon (T'', P''_T) by gluing T to T' along the equivalent paths in the tree, and by gluing P_T to P'_T by identifying the sides $\mathbf{a}_i\mathbf{a}_j$ and $\mathbf{a}'_i\mathbf{a}'_j$. This is the inverse operation to the splitting of a tree and polygon depicted in Fig. 6. The interior angle at the vertex $\mathbf{a}''_i$ in $P''_{T''}$ is the sum of the interior angles of $\mathbf{a}_i$ and $\mathbf{a}'_i$ in P_T and $P'_{T'}$. This allows us to arbitrarily increase the interior angle sum at a vertex so long as the property above is satisfied. We call such a doubling polygon *well-constructed*. We now define:

Definition 3.2 (*Geodesic Lang polygon*) Let P_T be a doubling polygon for a tree T on a surface S. We say that (T, P_T) is a *geodesic Lang polygon* if (1) it satisfies the geodesic Lang property, (2) each vertex of P_T corresponding to an internal node of T is a marker (interior angle is π) and (3) (T, P_T) is well-constructed.

Next, we investigate two important properties of geodesic Lang polygons.

The Geodesic Lang Property on Visible Pairs Implies the Geodesic Lang Property on All Pairs We first show that for a well-constructed doubling polygon to be a Lang polygon, it is sufficient that it satisfies the Lang property

only for all visible pairs. Later on in Sect. 6 we use this property to give an algorithm for computing a crease pattern in a geodesic Lang polygon.

Lemma 3.3 *Let P_T be a well-constructed doubling polygon for a tree T on a surface S, such that each vertex* $\mathbf{b}$ *of P_T corresponding to an internal node b of T is a marker. If the geodesic Lang property holds for all pairs of visible corners on P_T, then (T, P_T) is a Lang polygon.*

Proof Let $p = \mathbf{A}_0, \mathbf{A}_1, \ldots, \mathbf{A}_k$ be the geodesic path between two arbitrary vertices $\mathbf{a}_i = \mathbf{A}_0$ and $\mathbf{a}_j = \mathbf{A}_k$ in P_T. Since each consecutive pair $(\mathbf{A}_m, \mathbf{A}_{m+1})$ is visible, we have $d_{P_T}(\mathbf{A}_m, \mathbf{A}_{m+1}) \geq d_T(A_m, A_{m+1})$ for all m and thus the lengths of path p, $\sum_{m=0}^{k-1} d_{P_T}(\mathbf{A}_m, \mathbf{A}_{m+1})$ is $\geq \sum_{m=0}^{k-1} d_T(A_m, A_{m+1})$. Applying the triangle inequality, the latter sum is $\geq d_T(A_1, A_k) = d_T(a_i, a_j)$. Hence the geodesic Lang property is satisfied for any pair of vertices $(\mathbf{a}_i, \mathbf{a}_j)$. □

TreeMaker Always Produces Geodesic Lang Polygons Finally, we make the simple observation that the polygons resulting from a solution to the optimization problem solved in the first phase of TreeMaker are geodesic Lang polygons.

Lemma 3.4 *Let P_T be a simple, possibly non-convex planar doubling polygon for a metric tree T, satisfying the* (*Euclidean*) *Lang property. Then P_T satisfies the geodesic Lang property, and hence is a geodesic Lang polygon for T.*

Proof We remind the reader that a solution to the first phase of TreeMaker is a collection of non-crossing segments $\mathbf{a}_i\mathbf{a}_j$ with the property that the Euclidean distance between them is equal to the corresponding tree distance $d_T(a_i, a_j)$, with strict inequality for all the other pairs. Moreover, the segments subdivide a planar region (typically a square or a rectangle), hence it produces polygonal faces. Since inside each polygonal face the geodesic distance is at least equal to the Euclidean distance, the geodesic Lang property follows. □

This guarantees that when the first phase of TreeMaker produces an output, the resulting polygonal faces, even when not convex, satisfy the preconditions required by the geodesic universal molecule algorithm of Sect. 6 to work.

4 Generalized Sweep of a Geodesic Lang Polygon

In this section we define an algorithmic process on a geodesic Lang polygon called a *generalized sweep*. This concept is used in several places in the remainder of the paper. In Sect. 5, we show how to construct a particular family of surfaces we call *Lang surfaces* by an extrusion process. The boundary of each Lang surface is shown to be a geodesic Lang polygon and we end the section by showing that *the extrusion process is equivalent to a generalized sweep starting from its boundary*. Next, in Sect. 6, we give our generalization of the universal molecule algorithm to geodesic Lang polygons, which *uses a generalized sweep on the interior of its input Lang polygon* to generate a crease pattern. These crease patterns are shown (in Sect. 7.1) to be equivalent to Lang surfaces. Finally, in Sect. 7.2 we use the concept of a generalized sweep to prove the uniqueness claim of Theorem 1.1 by showing, essentially, that any Lang polygon gives rise to exactly one generalized sweep. Finally, Sect. 7.2 gives the details of the uniqueness claim.

To get started, we recall that a Lang polygon is a pair (T, P_T) of a tree T and a polygon P_T which is drawn on some underlying surface of zero-curvature. To give the main definition we need two additional processes: a *kinetic stretching process* defined on the tree in which the leaves shrink and a *parallel sweep process* in the polygon by which its edges are moved inwards in parallel at unit speed. We define these in Sect. 4.1 and end by defining the generalized sweep in Sect. 4.2.

4.1 Kinetic Trees and Parallel Sweeps

Kinetic Trees A metric tree (T, w) is turned into a *kinetic* tree by attaching a *stretching speed* $s(ab)$ to each leaf arc ab. The length of an arc ab at time $t \geq 0$ is given by $w(ab) + t\,s(ab)$. This gives rise to a family of trees $T(t)$ parametrized by time t. When the tree is embedded, we extend this motion to the embedded tree by moving leaf nodes inwards or outwards along the supporting line of the leaf arc. If $s(ab)$ is positive, then we say the arc is *growing*, otherwise *shrinking*.[3] This is naturally extended to any doubling cycle C_T for T to form a family of doubling cycles $C_T(t)$: simply grow/shrink each edge of C_T at the same speed as its corresponding arc in T. The process trivially maintains the property that at each time t, the cycle $C_T(t)$ is a metric doubling cycle for $T(t)$.

Parallel Sweep of a Polygon By moving the edges of a polygon inwards at unit speed in such a way that each edge remains parallel to its initial position, we obtain what is called a *parallel sweep* of the polygon. Each edge grows or shrinks to maintain incidence with its adjacent edges, resulting in a polygon (called a *sweping* or *parallel offset* polygon) whose edges are in one-to-one correspondance and parallel to the edges of the original polygon. An example appears in Fig. 11. Note that *a parallel offset polygon is well-defined only for polygons without negative boundary curvature.* A parallel sweep can sometimes proceed to infinity (e.g. when moving the edges of a convex polygon outwards), or can be made to stop at certain events, and resumed under different conditions. Various types of parallel sweeps differ by the nature of the relevant events.

4.2 The Generalized Geodesic Sweep

Overview In this section we define a generalized sweep for a geodesic Lang polygon (T, P_T). The basic idea is to make the tree T kinetic and to grow/shrink its leaf arcs while simultaneously performing a parallel sweep of the polygon. At certain events we may split the tree and polygon (according to the splitting operation defined in Sect. 2). Thus, at any given point we may have multiple shrinking polygon and tree pairs (hence the term "generalized"). We define this process so that to maintain the geodesic Lang property of the kinetic tree and parallel polygon pair. In particular, this means that the sweeping polygon is a doubling polygon for the growing/shrinking kinetic tree and the geodesic Lang property is satisfied. Maintaining this invariant requires that we process two types of events that occur in the sweep: contraction events and splitting events.

Contraction Events The first event type occurs when an arc of the tree and its corresponding edges in the polygon shrink to zero-length. Combinatorially, the zero-length arc in the tree is removed and the zero-length edges in the sweeping polygon are replaced with a single vertex.

Splitting Events The second event type occurs when the geodesic Lang property is satisfied with equality for some non-consecutive visible pair of corner vertices $(\mathbf{a}_i, \mathbf{a}_j)$ in P_T. We call this a *potential splitting event* because at this point the splitting operation may be applied to the tree and polygon. A potential splitting event occurs because the rate at which the distance is changing for some pair $(\mathbf{a}_i, \mathbf{a}_j)$ in the sweeping polygon is not necessarily the same as the rate at which the corresponding distance between a_i and a_j is changing in the tree. Thus, a pair that satisfies the geodesic Lang property initially with *inequality* may satisfy the geodesic Lang property with *equality* at some future time. In this case *we allow* that a splitting operation be applied to the polygon for $(\mathbf{a}_i, \mathbf{a}_j)$ and to the tree for (a_i, a_j) to obtain geodesic Lang polygons (T_L, P_L) and (T_R, P_R). We then continue the sweep independently in each. Note that it is not obviously the case that we *must split* at such an event in order to maintain the geodesic Lang property. For instance, it may be that immediately after such an event the distance between the two vertices in the sweeping polygon increases more quickly than the distance between the corresponding tree nodes. In such a case, even though the geodesic Lang property holds with equality, we can choose either to split or not. On the

[3] Note that in [5], we only allowed a leaf arc to shrink. Here we must allow both shrinking and growing to maintain certain correspondences with parallel sweeps of non-convex polygons.

other hand, if immediately after such an event the geodesic Lang property is violated, then we are *forced to split*. We call a potential splitting event at which the splitting operation is actually applied simply a *splitting event*. In the special case of negative boundary curvature (see Sect. 3) we require that the sweep be split immediately so that each resulting sweep polygon does not have negative boundary curvature. This requirement comes from the fact that a parallel offset polygon is only well-defined for polygons without negative boundary curvature.

The Generalized Geodesic Sweep Let (T, P_T) be a Lang polygon. The tree T is made kinetic by assigning to each leaf a speed of $-1/\tan(\theta_{\mathbf{a}})$ where $\theta_{\mathbf{a}}$ is half the interior angle measure at vertex $\mathbf{a}$. This speed is not arbitrary: it was chosen so that the speed at which each leaf arc shrinks is the same as the speed at which its corresponding edges in the polygon shrink (this follows from elementary trigonometry). With these choices, the sweeping of the polygon and shrinking of the tree as described above maintain the tree-and-doubling-polygon invariant. In addition, we process all contraction events, and optionally split the polygon and tree at some of the splitting events. If, throughout the process, we maintain the geodesic Lang property, then the process is called a *generalized geodesic sweep*.

We emphasize that this definition allows for multiple possible generalized sweeps for the same polygon and tree depending on whether we actually split at potential splitting events; we remind the reader that this occurs because we allow the sweep *not to be split* at a potential splitting event, as long as this does not violate the geodesic Lang property. However, we will see in Sect. 7.2 that in order to maintain the geodesic Lang property we must *always split* at these events. In other words, any time the sweep arrives at a potential splitting event, to continue past the event without actually splitting causes the sweeping polygon and tree to violate the Lang property, and thus the geodesic Lang property fails to hold. Ultimately, we will see that this implies that there is exactly one sweep for a given geodesic Lang polygon. This is used in Sect. 7.2 to prove the uniqueness claim from Theorem 1.1.

Can a Generalized Sweep Self-Touch? Another important property of a generalized geodesic sweep is that the sweeping polygon never self touches. In contrast to this behavior in our case, we remark that this property does not hold for the related parallel sweep used in the definition of the straight skeleton of a non-convex polygon [1]. Were such an event to occur, we would need an entirely different type of "splitting event", in which the sweeping polygon is split at the point at which the polygon self-touched (as is the case for the straight skeleton of a non-convex polygon).

We now show that in a generalized geodesic sweep such an event never occurs, for otherwise the sweep would violate the geodesic Lang property. We remind the reader that initially we have one sweeping polygon and stretching tree, but after a while they may have split at splitting events. Hence, that at any given time we have a collection of Lang polygons. But it is an easy observation that none of the distinct sweeping polygons cannot touch another one, since each one always moves its edges towards its interior. Thus as soon as a polygon is split, the resulting two sweep polygons diverge. Therefore we only need to prove that none of these parallel sweep polygons can self-touch, whioch would happen if either a vertex of the polygon "hits" some edge or two (or more) vertices "hit" each other. We now show that this is not the case:

Lemma 4.1 *A parallel sweep polygon remains simple* (*not self-touching*) *throughout a generalized geodesic sweep.*

Proof Suppose for contradiction that at some time t a parallel sweep polygon self touches, and take the minimum time t at which this occurs. There are two cases: (1) either some vertex of the polygon touches an edge elsewhere in the polygon, or (2) two vertices touch each other simultaneously but are not part of the same contraction event. We prove that in each case we arrive at a contradiction.

Case 1 Vertex $\mathbf{a}_j$ hits edge $\mathbf{a}_i\mathbf{a}_{i+1}$ in the polygon. This implies that $\mathbf{a}_j$ is reflex in P_T (otherwise the polygon won't be simple, and t won't be the minimum time when such an event occurs). Thus the corresponding leaf arc is growing. By definition, $\mathbf{a}_i\mathbf{a}_{i+1}$ corresponds to a path in T between leaf nodes a_i and a_{i+1} and $\mathbf{a}_j$ corresponds to the leaf node a_j in T. Since a_j is a leaf node and the corresponding leaf arc was growing (hence its length is strictly larger than zero), a_j does not lie on the path between a_i and a_{i+1} in T. Therefore $d_T(a_i, a_{i+1}) < d_T(a_i, a_j) + d_T(a_j, a_{i+1})$. Since $(\mathbf{a}_i, \mathbf{a}_{i+1})$ is an edge in the doubling polygon, then $d_{P_T}(\mathbf{a}_i, \mathbf{a}_{i+1}) = d_T(a_i, a_{i+1})$, and thus we have that

$d_{P_T}(\mathbf{a}_i, \mathbf{a}_j) + d_{P_T}(\mathbf{a}_j, \mathbf{a}_{i+1}) < d_T(a_i, a_j) + d_T(a_j, a_{i+1})$, which entails that the geodesic Lang property is violated in the polygon P_T for either the pair $(\mathbf{a}_i, \mathbf{a}_j)$ or $(\mathbf{a}_j, \mathbf{a}_{i+1})$ - a contradiction.
Case 2 Let $\mathbf{a}_i$ and $\mathbf{a}_j$ be the touching vertices that are not part of the same contraction event. At least one must be a reflex vertex in P_T; without loss of generality let this be $\mathbf{a}_i$. Then the leaf arc incident to a_i has positive, non-zero length (since it is growing), and so a_i is at least some positive distance from any other node in T. Since $\mathbf{a}_i$ and $\mathbf{a}_j$ are not part of the same contraction event, then a_i and a_j are distinct in T and thus $d_T(a_i, a_j) > 0$. But the distance from $\mathbf{a}_i$ to $\mathbf{a}_j$ is zero, hence the geodesic Lang property is violated. □

As a consequence we have:

Lemma 4.2 *For any given geodesic Lang polygon* (T, P_T) (*drawn on some flat surface* S) *there exists a generalized geodesic sweep.*

Proof Make T kinetic as in the definition of a generalized sweep and perform a parallel sweep of P_T and simultaneous stretching of T. Define the generalized sweep to split whenever a potential splitting event occurs. If multiple potential splitting events occur simultaneously, then we take one after another, splitting until no potential splitting events remain, and then continue. Note that here we leave open the possibility that "processing" one potential splitting event removes another by separating its two vertices on opposite sides of the split (although we will see in Sect. 7.2 that this is not possible).

The sweep moves the sides of P_T inwards towards its interior, and so as the sweep progresses, we must either encounter (1) a contraction event, (2) a potential splitting event, or (3) an event in which the sweeping polygon self-touches. By definition, the sweep maintains that for consecutive pairs $\mathbf{a}_i$ and $\mathbf{a}_{i+1}$ the distance in the tree and the distance in the sweeping polygon is equal (i.e. $d_T(a_i, a_{i+1}) = d(\mathbf{a}_i, \mathbf{a}_{i+1})$). From this we can rule out case (3). Assume we get to a point at which P_T self touches. Then by Lemma 4.1 (T, P_T) is no longer a geodesic Lang polygon. But this means that at some earlier time, there must have been a potential splitting event at which we did not split, a contradiction.

Thus, as long as we take as our rule that we always split at potential splitting events, then we will encounter events of type (1) and (2) only. But each contraction event removes at least one vertex from the sweeping polygon (and one leaf from the tree), and each splitting event splits the polygon and tree into two polygons and two trees each with strictly fewer vertices/leaf nodes and at least three vertices/leaf nodes (since splitting events always occur for non-consecutive vertices).

The result thus follows by induction on the number of events encountered in the sweep. □

5 Lang Surfaces

The next major concept needed for our result is that of a *Lang surface.* We first defined Lang surfaces in [5], where they had (intrinsically) convex boundary polygons. We now investigate the set of zero-curvature (intrinsically flat) Lang surfaces in full generality. Such a surface is constructed with respect to a tree T and is formed by combining a small set of basic building block elements, according to certain gluing rules, to form disk-like surfaces.

5.1 Overview

We define Lang surfaces constructively in Sect. 5.2. We need two families of building block surfaces and two operations for gluing them together to form Lang surfaces (extension and combination). Constructing the building blocks relies on the kinetic trees of Sect. 4.1. This follows the same construction as in [5], except that we allow tree edges to both grow and shrink, and we remove the restriction that Lang surfaces have a convex boundary. In Sect. 5.3 we investigate the properties of Lang surfaces with zero curvature and show that the boundary polygon P_T of a zero-curvature Lang surface S constructed on a tree T is a geodesic Lang polygon; this proves the necessity of the geodesic Lang property in Theorem 1.1.

Finally, in Sect. 5.4, we observe that the extrusion processes for creating the building blocks of a Lang surface can be chained together to form an intrinsic *parallel sweep* of the surface with splitting events. This sweeps the boundary of the surface inward in such a way that each edge of the polygon remains (intrinsically) parallel to its original position and all edges move at unit speed. A splitting event splits the sweeping polygon into two polygons and the sweep continues recursively in each. We used this fact in the convex case [5] to put the universal molecules into correspondence with the zero-curvature Lang surfaces with convex boundary. We observe that concepts from the convex case transfer to the general case studied in this paper because the parallel sweep of a Lang surface (convex or otherwise) locally proceeds in the same way as in the convex case in the plane. As in the convex case, in general a Lang surface may have non-zero curvature at its interior vertices. To define the special family of flat Lang surfaces, we impose zero-curvature on all interior vertices. This, however, explicitly allows for high curvature (angle sum $> 2\pi$) on the boundary, which presents a potential problem: what does the sweep look like locally at such a vertex? In our construction, these high curvature vertices only occur at combination operations, which correspond to a splitting of the sweep polygon. After the split each vertex of the sweeping polygon has angle less than 2π, and so the sweep looks locally like a sweep of a polygon in the plane.

5.2 Constructing Lang Surfaces

We first define two types of building blocks: extrusion disks and extrusion rings. Each is built with respect to a kinetic tree via an extrusion process. The boundary polygon(s) for an extrusion disk or ring are doubling cycles. We then give two gluing operations, extension and combination, for joining them. The gluing operations can be applied only if the two input surfaces meet certain conditions coming from the tree. In [5], all leaf arcs are required to shrink and the combination operation is restricted to produce only surfaces with convex boundary. This ensures that the Lang surfaces, as defined in [5], are convex, a restriction that is used to prove correspondences with Lang's original formulation. In this paper, however, we fully generalize to Lang surfaces with non-convex boundary by allowing leaf arcs to grow as well as shrink and by allowing the combination operation to result in surfaces with (intrinsically) non-convex boundary polygons. In the end, many of the properties of Lang surfaces studied in [5] are preserved; we point them out along the way.

Extrusion Surfaces The two types of building block surfaces are defined with respect to a kinetic tree (T, w, s) and a positive *extrusion height* h. The extrusion process is given by the following construction. First, we fix an embedding of the tree in the xy-plane. Next, we shrink or grow each leaf arc (according to its speed $s(ab)$) while simultaneously moving the plane containing the tree upwards in the positive z-direction at unit speed. We simulate both of these motions for t from 0 to h. At time t we have $T(t)$ embedded in the $z = t$ plane. Next, we obtain a doubling polygon $P_T(t)$ for $T(t)$ by embedding the doubling cycle $C_T(t)$ directly "on top of" $T(t)$ in the sweeping plane, meaning each vertex $\mathbf{v}$ of $P_T(t)$ is placed directly on top of its corresponding node v in the tree $T(t)$ in the $z = t$ plane. We call this polygon $P_T(t)$ (embedded in the $z = t$ plane) the *extrusion polygon* at height t. The *extrusion surface* of height h for (T, w, s) is the trace of the edges of P_T in this sweep. See Fig. 7. We restrict h to be not greater than the first time t at which an arc shrinks to zero length in $T(t)$.

Extrusion Disks If we assign each arc ab of T a stretching speed $s(ab)$ equal to $-h/d_T(a, b)$ and T has a single internal node, then all of the edges of the extrusion polygon shrink to a single point $\mathbf{p}$ at $t = h$. The resulting

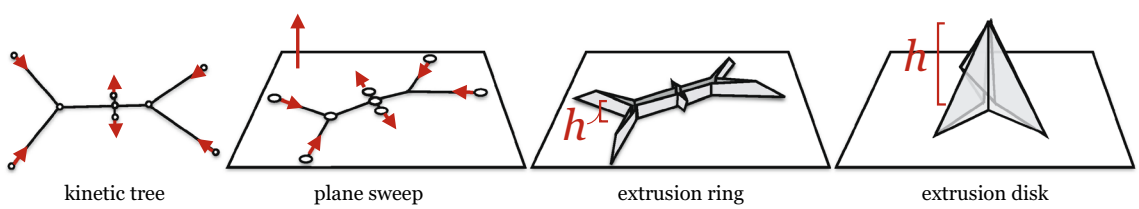

Fig. 7 A kinetic tree, the kinetic tree embedded in a sweeping plane, an extrusion ring of height h and an extrusion disk of height h. Note that we "open up" the ring and disk slightly for visualization purposes

extrusion surface is topologically a disk, which we refer to as an *extrusion disk*. The point **p** is the single interior vertex for the disk. Its curvature is $2\pi - 2\sum_{ab\in T} \arctan(d_T(a,b)/h)$, and there is a unique height h for each tree T resulting in an extrusion disk of zero curvature. Each face is a right triangle with one edge equal to the initial doubling polygon edge in the xy-plane, one edge equal to an edge of length h lying perpendicularly above the xy-plane, and the remaining edge a hypotenuse traced by a vertex of the extrusion polygon corresponding to a leaf node. There is another degenerate situation that results in a disk, namely when a tree $T(h)$ has all of its leaf arcs incident to only two distinct internal nodes. In this case, of course, the tree is a path. If we identify the edges of the corresponding extrusion polygon at height h, we obtain a disk surface. Handling this second case is a straightforward extension of the first, and so we focus only on the first.

Extrusion Rings The second type of extrusion surface is defined for a kinetic tree such that no leaf arc shrinks to zero-length at a time $t < h$ and the tree $T(h)$ has at least three leaf nodes. The resulting *extrusion ring* of height h is a ring-like surface with lower and upper boundary polygons that are doubling polygons of T and $T(h)$. See Fig. 7. We note that because each vertex of an extrusion ring is on the boundary, all extrusion rings are flat, though *a priori* a given extrusion ring may not have an open, flat realization (as pointed out in Sect. 2.1).

Boundary Curvature and Face Geometry of Extrusion Surfaces Recall that the boundary polygon of an extrusion surface is a doubling polygon for the tree T. Each vertex **v** of the boundary is incident to exactly two faces in the surface. If the vertex corresponds to an internal node of T, then its (x, y)-coordinates do not change throughout the extrusion process. This implies that the two face angles incident to **v** are each $\pi/2$ and the angle sum is π. This also implies that an edge in the extruding polygon corresponding to an internal arc in T traces out a rectangular face. On the other hand, a leaf node moves so that its incident leaf arc grows or shrinks according to the speed $s(ab)$. An edge of the extruding doubling polygon corresponding to a leaf arc in T traces out a right triangle if the arc shrinks to zero at $t = h$, or a right trapezoid otherwise. An edge corresponding to an internal arc of the tree traces out a rectangular face. In particular, this means that the vertices of the boundary of an extrusion disk or lower boundary of an extrusion ring have angle sum less than 2π; in other words, the curvature at these vertices is non-negative.

Operations for Constructing Lang Surfaces Lang surfaces are obtained by starting with extrusion disks and rings and combining them using the following two operations, *combination* and *extension*. A Lang surface constructed on an embedded kinetic metric tree (T, w, s) is a disk-like piecewise linear surface in $\mathbf{R}^3$ whose boundary is a doubling polygon T. Lang surfaces are defined inductively: (a) all extrusion disks are Lang surfaces, and (b) new Lang surfaces are formed by either applying the extension operation to a Lang surface and an extrusion ring, or by applying the combination operation to two Lang surfaces (in each case meeting certain preconditions).

The Extension Operation This operations takes as input an extrusion ring R of height h and a Lang surface S with the precondition that the upper boundary polygon of R and the boundary polygon of S are the same doubling polygon for the same tree (except that the upper boundary polygon of R is in the $z = h$ plane and the boundary of S is in the xy-plane). We *extend* S with R by translating S upwards in the positive z-direction by h, bringing its

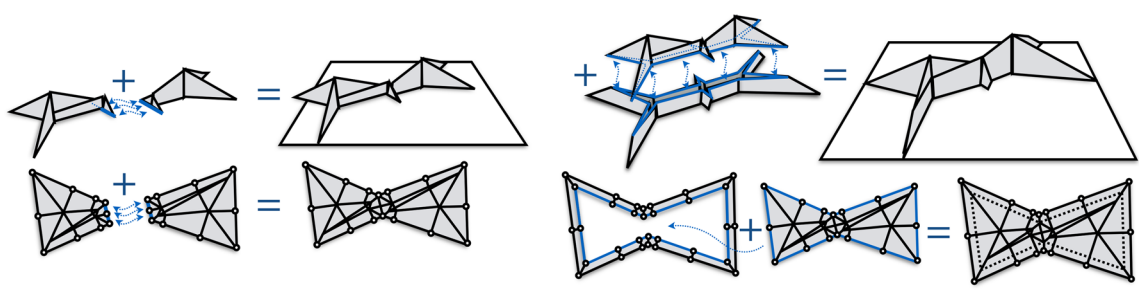

Fig. 8 The combination (*left*) and extension (*right*) operations depicted extrinsically (*top*) and intrinsically (*bottom*)

boundary polygon into the $z = h$ plane. We then identify the corresponding edges of the upper boundary polygon of R and the boundary polygon of S to form the output Lang surface. See Fig. 8.

The Combination Operation This operation takes as input two Lang surfaces S_1 and S_2 constructed on trees T_1 and T_2 (resp.). The precondition for this operation is that there exists a tree T and a pair of leaf nodes (a_i, a_j) in T, such that when T is split between a_i and a_j, the result is T_1 and T_2. In particular, this means that a_i and a_j are consecutive leaf nodes in T_1 and T_2 and thus appear as consecutive corner vertices $\mathbf{a}_i$ and $\mathbf{a}_j$ in the boundary polygons of both S_1 and S_2. The paths between $\mathbf{a}_i$ and $\mathbf{a}_j$ in both S_1 and S_2 correspond to the path in T between a_i and a_j. We *combine* S_1 to S_2 along this *gluing path* by identifying the corresponding edges along the path. Note that because a_i and a_j are consecutive in S_1 and S_2, then this gluing path is intrinsically straight. The output surface S is a Lang surface constructed on T. See Fig. 8.

Definition 5.1 (*Lang surface*) A *Lang surface* is any surface formed by joining a collection of extrusion disks and rings using the combination and extension operations.

The boundary polygon for a Lang surface S is, by definition, a doubling polygon for a tree T, which we call its *boundary tree*. We say that S is *constructed on* T. Note that each Lang surface is *tree projectible*, meaning that its projection onto the xy-plane is a tree T', which is combinatorially equivalent to its boundary tree and geometrically contains it. It should also be noted that different embeddings of the boundary tree give rise to the same intrinsic surface. This entails, in particular, that a continuous motion of the boundary tree in the plane corresponds to a continuous motion of the Lang surface that maintains the shape of each of its faces. We can then align all of the arcs of the tree along a single line, which "lines up" the boundary edges along a single axis. For this reason, a Lang surface is called *uniaxial*, a term used by origamists to describe structures of this type.

5.3 Properties of Zero-Curvature Lang Surfaces

We are primarily interested in conditions under which the operations described above produce flat Lang surfaces, since these serve as a generalization of flat, polygonal sheets of paper. For the combination operation, it is necessary and sufficient that both input surfaces be flat. For the extension operation, it is necessary and sufficient that (1) the input Lang surface is flat (as we have seen, all extrusion rings are flat by definition), and (2) the sum of the two angle sums at each vertex along the gluing path is 2π.

Negative Boundary Curvature In [5], we considered only Lang surfaces with zero curvature and (intrinsically) convex boundary polygons. Here we remove the restriction on convex boundary polygons. This results in flat Lang surfaces with non-convex boundary and even with negative curvature along the boundary. This never occurs as the result of an extension operation, because, as we have seen, the lower boundary of an extrusion ring always has non-negative curvature; however, we can perform an arbitrary number of consecutive combination operations, which allows us to make the angle sum at a boundary vertex as large as we want.

An example is shown in Fig. 9, where we construct a Lang surface with a boundary vertex v of high curvature using successive combination operations. A single "unit" Lang surface is first shown, intrinsically. The boundary

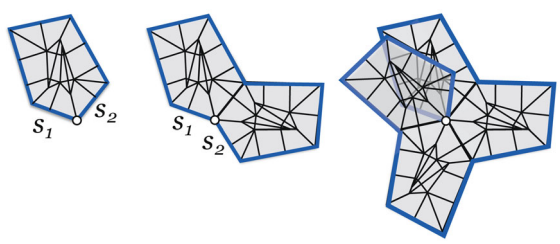

Fig. 9 Constructing a Lang surface with a boundary vertex v of high curvature using successive combination operations

chains s_1 and s_2 match metrically and combinatorially and we can glue a copy of the surface to itself. Then, iteratively, we arbitrarily increase the angle sum at v.

We note two properties of flat Lang surfaces with negative curvature on the boundary. First, as soon as we have introduced negative curvature to the boundary, we can no longer apply extension operations, since to do so would necessarily involve creating an interior vertex of negative curvature, and thus the resulting surface would not be flat. Second, the boundary polygon of this surface is, by construction, a well-formed doubling polygon (in the sense of Sect. 3).

Necessity of the Geodesic Lang Property The main result of this section can now be described: the boundary polygon P_T of a Lang surface S for a tree T is a geodesic Lang polygon for T. The proof is the same as in Lang's original paper [12], the main difference being that we use geodesic paths rather than straight line segments. We reproduce it here for completeness:

Lemma 5.2 *Let S be a Lang surface for a tree T and P_T be its boundary polygon. Then (T, P_T) is a geodesic Lang polygon on S.*

Proof Let p denote the geodesic path between any two vertices $\mathbf{u}$ and $\mathbf{v}$ of P_T. In the realization of S, p is a polygonal path in $\mathbf{R}^3$. Recall that the projection of S onto the xy-plane is an embedding of T, thus the projection of p onto the xy-plane contains the path from u to v in T. The projection of p has length less than or equal to the length of p, which proves that the distance between u and v in T is less than or equal to the distance in S between $\mathbf{u}$ and $\mathbf{v}$. □

5.4 The Extrusion Process as a Generalized Sweep

The building blocks of a Lang surface are generated by tracing a polygon as its edges grow or shrink. We now take one such block, say a ring R and "replay" the motion of the extrusion process across the surface, from the bottom up. We observe that each edge of the extruding polygon, across the face that it generates, moves in such a way that it remains (intrinsically) parallel to its original position. See Fig. 10 (left). This process is referred to as the *extrusion sweep* for disks and rings, and can be generalized, recursively, to an *extrusion sweep* for the entire Lang surface S, as follows.

In all cases the extrusion sweep starts as the boundary polygon of S. If S is formed by an extension operation on a Lang surface S' and a ring R, then we first perform the extrusion sweep of R, and then recursively continue the extrusion sweep of S'. If S is formed by a combination operation on Lang surfaces S_1 and S_2, then the sweep is defined by first splitting the sweep polygon along the gluing edge between S_1 and S_2, and then continuing the sweep independently in each. In the base case that S is an extrusion disk, we simply perform the extrusion sweep of S and stop. Note that by construction the state of the extrusion sweep at time t is equivalent to the intersection of the $z = t$ plane with S. The edges of S are given by the trace of the vertices of the sweep together with the splitting edges

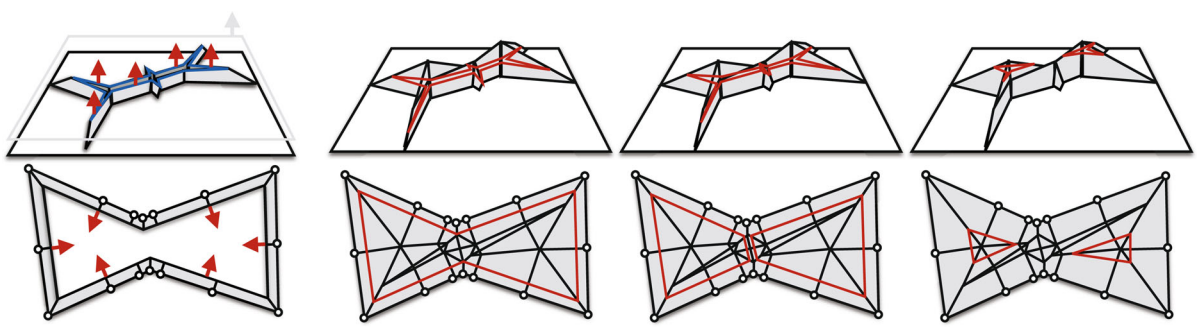

Fig. 10 The extrusion sweep of a ring (*left*) and of an entire Lang surface (*right*)

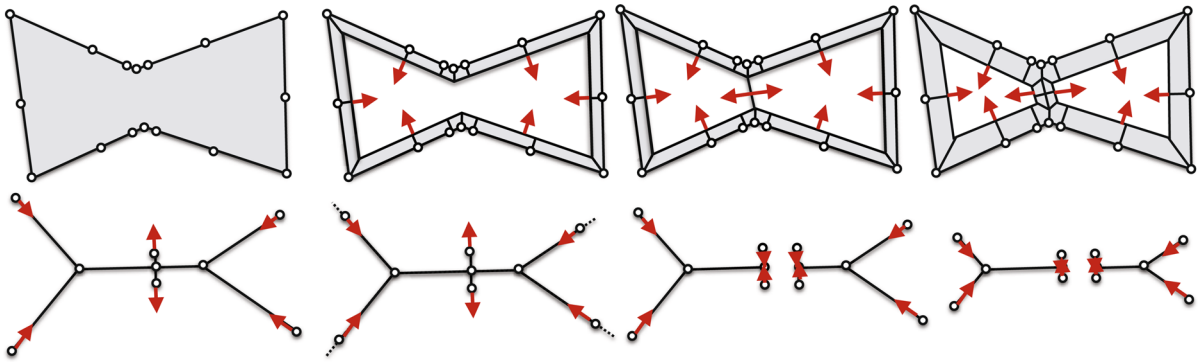

Fig. 11 A visualization of the sweep of an input geodesic Lang polygon (T, P_T) including a splitting event

introduced for combination operations. The faces of S are the traces of the edges of the sweep. Figure 10 shows a Lang surface formed by combining two extrusion disks using a combination operation and then an extension of the resulting surface with an extrusion ring. When we replay the extrusion "from the bottom up" we first (intrinsically) perform a parallel sweep of of the ring, then split the sweep into two, and then simultaneously perform parallel sweeps of the the two extrusion disks. We prove:

Lemma 5.3 *The extrusion sweep of a Lang surface is a generalized geodesic sweep of its boundary polygon and tree.*

Proof By construction, the extrusion sweep as defined above is a *parallel sweep* (as defined in Sect. 4) of the boundary polygon P_T of S with splitting events that occur when the sweep hits the base of a part of S formed by the combination operation. We only need to show that when we attach the appropriate speeds to T making it kinetic and perform the extrusion sweep while shrinking the tree, then (1) it maintains the Lang property throughout the sweep and (2) a polygon in the extrusion sweep is split only when the Lang property holds with equality.

For (1) we first note that the speed assigned to each leaf arc in T in a generalized sweep of (T, P_T) across S is the same speed at which the same leaf arc shrinks during the extrusion process. The claim then follows by the same argument that geodesic Lang property holds on S.

Claim (2) follows from the definition of the extrusion sweep, and the definition of the combination operation, which guarantees that the gluing path is straight, and the length of the path is equal to the corresponding distance in the tree. By definition, this gluing path is used to split the extrusion sweep polygon. From this and claim (1) we infer that the geodesic Lang property holds with equality for the the pair of vertices on which we split and thus the extrusion sweep is a generalized sweep. □

Next we argue that every generalized geodesic sweep corresponds to the extrusion sweep of some Lang surface.

Lemma 5.4 *Let (T, P_T) be a Lang polygon. Then there exists a Lang surface S_T constructed on T such that the extrusion sweep of S_T is the generalized sweep of (T, P_T).*

Proof The details of this proof are the same as in [5] but for clarity we have introduced extrusion and generalized geodesic sweeps (these concepts were implicit in [5] but were not given names). We sketch the proof here with the new terminology, and point to the previous paper for the details.

We keep track of events during a generalized sweep using an *event tree*: a node is an event and the sweep between events is represented by a directed arc from the later event to the earlier. A splitting event results in multiple incoming edges, one for each of the split polygons in the sweep, whereas a contraction event (that is not also a splitting event) results in one single incoming edge. The root node of the tree is a special starting event denoting the initial boundary polygon, and each leaf is an event at which the one of the sweeping polygons shrinks to a single point as is stopped. We only record splitting events when the polygon and tree are actually split.

The proof now proceeds by induction on the depth of the event tree, showing that each edge of this tree corresponds to an extrusion surface and each node corresponds to operations on Lang surfaces.

In the base case the event tree has one edge: one event occurs in the generalized sweep, namely a simultaneous contraction to a single point. If t is the time at which this event occurs, S_T is the extrusion disk of height t constructed on T and f the face traced by an edge e of the generalized sweep in (T, P_T), then (from the definitions of the generalized sweep and the extrusion disk) it follows that the face traced by the corresponding edge in S_T is congruent to f. Hence S_T is the same intrinsic surface as P_T and thus the generalized sweep of (T, P_T) is equal to the extrusion sweep of S_T.

For the inductive step, let (T, P_T) be a geodesic Lang polygon with event tree of depth $d + 1$. There are two cases we need to handle. The first is when we split at time $t = 0$ in the generalized sweep of (T, P_T) and the second is when the first event occurs at some time $t > 0$. In the first case, the polygon P_T is split into polygons $P_1, \ldots, P_k$ at time $t = 0$. Each of these correspond to a sub-tree of the event tree. By inductive hypothesis, there are Lang surfaces $S_1, \ldots, S_k$ corresponding to $P_1, \ldots, P_k$. The result then follows by observing that the inverse of the splitting process on P_T is the combination operation on $S_1, \ldots, S_k$. In the second case, the root node of the event tree has in-degree one. That edge corresponds to a sweep from P_T to $P_T(t)$. By the same argument as in the base case above, the surface traced by the sweep between P_T and $P_T(t)$ is equivalent to the extrusion ring R constructed on T where the speeds assigned to T for the extrusion process are the same as for the generalized sweep. Then by inductive hypothesis, we have a Lang surface S' constructed on $T(t)$ and observe that applying the combination operation to S' and R gives us a Lang surface S_T constructed on T for which the extrusion sweep is equivalent to the generalized sweep of (T, P_T). □

Summary We have defined a family of surfaces, called Lang surfaces, that are built on top of a tree. A Lang surface is formed by gluing together extrusion disks and rings using the extension and combination operations. The boundary of a Lang surface is a geodesic Lang polygon. We then put the family of zero-curvature Lang surfaces into correspondence with the generalized sweeps of a Lang polygon by showing that the extrusion process that generates a given Lang surface corresponds to a generalized sweep of its boundary polygon (Lemma 5.3), and that a generalized sweep of a Lang polygon always corresponds to the extrusion process for some Lang surface (Lemma 5.4). In the next section we describe an algorithm for simulating the sweep. Finally, we showed that for any given Lang polygon there is a unique generalized sweep. This puts the Lang polygons into one-to-one correspondence with the zero-curvature Lang surfaces.

6 Computing the Geodesic Universal Molecule

In the previous section we defined Lang surfaces, showed that the boundary of a Lang surface is a geodesic Lang polygon, and showed that the extrusion process is equivalent to a generalized sweep. We now go in the opposite direction. We start with a geodesic Lang polygon (T, P_T) and perform a particular generalized sweep of the polygon maintaining that the sweeping polygon is a geodesic Lang polygon. So far we have not shown that there is a unique generalized sweep for a Lang polygon; instead we have stated that it is possible that at a potential *splitting event*, we may be able to choose whether to actually split or not and in either case maintain the geodesic Lang property. Regardless of the choice we make, we get a generalized sweep. The algorithm described below simulates one particular generalized sweep–namely the one in which we always split at a potential splitting event. We showed in Lemma 4.2 that this sweep must exist. We use it to compute a subdivision of a geodesic Lang polygon into vertices, edges, and faces. We call this subdivision the *geodesic universal molecule* of (T, P_T). The edges of the subdivision are given by tracing the vertices of the sweeping polygon (along with the edges introduced at a splitting event).

In this section we describe an algorithm for simulating the sweep and computing the geodesic universal molecule. Then in Sect. 7.1 we prove that there exists a Lang surface with the same boundary Lang polygon and generalized sweep (using the connection between generalized sweeps and extrusion sweeps given in Lemmas 5.3 and 5.4). Finally, in Sect. 7.2 we show that there is *exactly one generalized sweep* for any given Lang polygon (T, P_T). This implies a one-to-one correspondence between Lang surfaces and geodesic universal molecules.

Let us note again that the sweep in the convex case is a generalization of the straight skeleton sweep [1]. For non-convex polygons, the straight skeleton sweep may encounter an event in which a reflex vertex "hits" another edge of the sweep necessitating a split (not to be confused with our splitting events). *We emphasize that such an event cannot occur in our case, because our sweep maintains the invariant that the sweeping polygon is a geodesic Lang polygon, and therefore by Lemma 4.1 remains simple. This implies that before such an event occurs a splitting event must precede it.*

The reader should keep in mind that this sweep is performed intrinsically on the surface of P_T; however, in the case that P_T flattens out onto a simple polygon in the plane (convex or non-convex), then we can perform the sweep explicitly in the plane. The algorithm we describe solves the following:

Geodesic Universal Molecule Problem *Given a geodesic Lang polygon* (T, P_T) *compute a flat Lang surface constructed on* T *that is intrinsically equivalent to* P_T.

The Algorithm The *input* is a geodesic Lang polygon (T, P_T) and the *output* is a planar graph G embedded on the surface of P_T such that the subdivision of P_T induced by G is (intrinsically) equivalent to a Lang surface S. We call G the *geodesic universal molecule* for (T, P_T). We assume the existence of primitive operations for computing (intrinsic) parallel offset polygons at a given height h and a predicate for determining whether two vertices of a polygon are a visible pair.

The algorithm follows the basic procedure similar to the case of convex, planar Lang polygons. We make the tree T kinetic by attaching a stretching speed $s(ab)$ to each leaf arc ab. The stretching speed is determined with respect to interior angle of $\mathbf{a}$ in P_T, so that the arc ab maintains the same length in T (as it grows/shrinks) as the length of the corresponding edge in the sweeping polygon. By elementary trigonometry, we have that $s(ab)$ must be $-1/\tan(\theta_{\mathbf{a}})$ where θ_a is the interior angle measure of $\mathbf{a}$ in the sweeping polygon. We refer to the kinetic process in the tree and parallel sweep in the polygon collectively as *the sweep*.

We remind the reader that for the sweep to be a generalized sweep it must maintain the invariant that the kinetic tree and sweeping polygon form a geodesic Lang polygon (Sect. 4). Maintaining this invariant requires processing two types of events. A *contraction event* occurs when an arc of T and its corresponding edges in P_T shrink to zero length. In this case we remove (or contract) the zero length arcs/edges. A *splitting event* occurs when for two non-consecutive corners in P_T, say $\mathbf{a}_i$ and $\mathbf{a}_j$, the geodesic Lang property holds with equality: $d_{P_T}(\mathbf{a}_i, \mathbf{a}_j) = d_T(a_i, a_j)$. As a consequence of Lemma 3.3, this pair $(\mathbf{a}_i, \mathbf{a}_j)$ is a visible pair. We then split the tree between a_i and a_j and split the polygon between $\mathbf{a}_i$ and $\mathbf{a}_j$ to obtain a left tree and polygon (T_L, P_L) and right tree and polygon (T_R, P_R) both of which form geodesic Lang polygons with the visible pair now on the boundary. Finally, we continue the sweep independently in each. The output crease pattern is the union of the trace of the vertices throughout the sweep and the splitting edges introduced at a splitting event. See Fig. 11 for a visual overview of the algorithm. The recursive pseudocode below summarizes the sweep process:

ALGORITHM 1 GeodesicUMAlgorithm(T, P_T)

```
function GeodesicUMAlgorithm(T, P_T):
    if (T, P_T) is a baseCase:
        return handleBaseCase(T, P_T)

    nextEvent := findNextEvent(T, P_T)

    (T', P_T'), R := AdvanceSweepAndTileRing(T, P_T, nextEvent)

    if nextEvent is a contraction event:
        (T', P_T') := Contract(T', P_T')
        G' := GeodesicUMAlgorithm(T', P_T')
    else:
        (T_L, P_L), (T_R, P_R) := Split(T', P_T', nextEvent)
        G_L := GeodesicUMAlgorithm(T_L, P_L)
```

```
    G_R := GeodesicUMAlgorithm(T_R, P_R)
    G' := MergeCreasePatterns(G_L, G_R)
endif

G := MergeCreasePatternWithRing(G', R)
return G
```

Algorithm Overview Each call to GEODESICUMALGORITHM takes as input a geodesic Lang polygon (T, P_T) and recursively "fills in" the interior of P_T with a crease pattern G. The occurrence of the next event is handled by the sub-routine FINDNEXTEVENT, which stores in the NEXTEVENT structure the time at which the event occurs and a pointer to the contracting edge/arc in P_T and T (in case of a contraction event), or the splitting pair of vertices/nodes in P_T and T. If simultaneous contraction and splitting events occur, the contraction events are processed first. How FINDNEXTEVENT finds the next event is described below under the heading "Computing the events." Once the next event has been identified, the ADVANCESWEEPANDTILERING subroutine computes the state of the tree and polygon at the event (T', P'_T) by moving all the nodes/vertices of (T, P_T) inwards by the appropriate amount. It then "tiles" the annular region between P_T and P'_T by adding edges between corresponding vertices (see "Simulating the sweep" below). If the next event is a contraction event, then P'_T will contain zero-length edges which must be contracted to a single vertex. The associated arc of T' must also be pruned. This is handled by a call to CONTRACT, which is a subroutine for contracting zero-length edges/arcs in P_T and T. The crease pattern for the interior of (T', P'_T) is then computed recursively and merged with the computed annulus R by MERGECREASEPATTERNWITHRING. If the next event is a splitting event, then the SPLIT subroutine splits the tree and polygon using the splitting pair stored in NEXTEVENT (using the operation described in Sect. 2). The splitting operation returns the split of (T, P_T) into polygons P_L and P_R and trees T_L and T_R. The crease patterns G_L and G_R on the interiors of P_L and P_R are then recursively computed, merged together, and finally merged with the tiling of the annulus R. The entire process is illustrated in Fig. 12.

Computing the Events Since corresponding leaf arcs and polygon edges shrink/grow at the same rate, to find the next *candidate contraction event* it suffices to check each leaf arc ab for the smallest value of t for which $d_T(a, b) - 1/\tan(\theta_{\mathbf{a}}/2) = 0$ over all leaf arcs ab.

Finding the next splitting event is a bit trickier. We first note that since we maintain that the sweeping polygon and tree form a geodesic Lang polygon, by Lemma 4.1 we never get to a point where a reflex vertex "hits" another edge of the polygon. By Lemma 3.3, the next splitting event must occur for a visible pair; however, as the sweep progresses the set of visible pairs changes, and there is no guarantee that the current visible pairs will be visible at the next splitting event. Let $\bar{P}_T$ denote the open, flat realization of P_T, $\bar{\mathbf{a}}_i$ be the position of $\mathbf{a}_i$ in $\bar{P}_T$, and $\bar{V}_{\mathbf{a}_i}$ denote the velocity of $\bar{\mathbf{a}}_i$ in the sweep of $\bar{P}_T$.

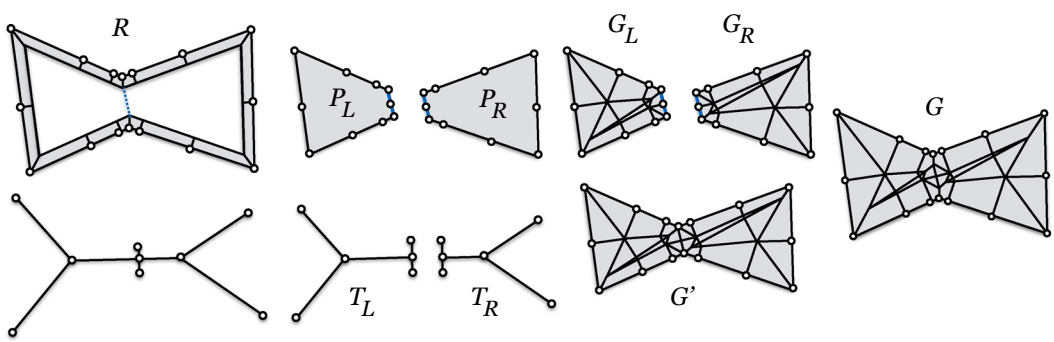

Fig. 12 An illustration of the recursive procedure in Algorithm 1. We first compute the ring R. Then contract or split (in this case split), fill in each side of the split recursively, and merge this with R to produce an output

For each pair of corners $(\mathbf{a}_i, \mathbf{a}_j)$ find the smallest value of t satisfying

$$||(\bar{\mathbf{a}}_i + t\bar{V}_{\mathbf{a}_i}) - (\bar{\mathbf{a}}_j + t\bar{V}_{\mathbf{a}_j})|| = d_T(a_i, a_j) - t(1/\tan(\theta_{\mathbf{a}_i}/2) + 1/\tan(\theta_{\mathbf{a}_j}/2)). \tag{6.1}$$

The left side of Eq. 6.1 is the *Euclidean distance* between the two vertices in the flattened sweep at time t and the right side is the corresponding tree distance in the shrinking tree. Solving for t gives the time at which the Euclidean distance becomes equal to the shrinking tree distance. For a visible pair the geodesic distance in P_T and the Euclidean distance in $\bar{P}_T$ are identical. Thus Eq. 6.1 is valid for checking the geodesic Lang property for any visible pair. We still do not know which pair will be visible. We check this in a straightforward manner by sorting the times t satisfying Eq. 6.1 over all pairs of non-consecutive corners in P_T. Then, in increasing order over the times t_i of these candidate events, advance the sweep to each time t_i, and check whether the corresponding pair of corners $(\mathbf{a}_i, \mathbf{a}_j)$ is visible in the sweep polygon at t_i. The first pair we find (smallest value t_i) is the pair for the *candidate splitting event*. The next event is whichever candidate event (contraction or splitting) occurs at a smaller time t.

Base Cases The base case is when the next event is a contraction of all leaf arcs simultaneously to a single node. (And the degenerate case where all leaf arcs contract simultaneously leaving a path, rather than a single node, in the tree.) This can only occur if all arcs are shrinking, and thus the sweeping polygon is convex. This is the same as in the planar, convex case [5] and occurs if and only if all the angle bisectors intersect at a single point. Let $\mathbf{p}$ be the point on the interior of P_T at which sweep contracts. The HANDLEBASECASE sub-routine returns the subdivision of P_T into faces given by adding a vertex at $\mathbf{p}$ and adding an edge from that vertex to each vertex of P_T.

Analysis The recursive algorithm we describe above does not simulate all parts of the sweep simultaneously. When the sweep is split into two it first recursively simulates the sweep on the interior of one side of the split, and then simulates the sweep on the interior of the other side of the sweep. We store the output universal molecule G, and the tiled ring R as a doubly-connected edge list (DCEL) [4]. Using this structure, computing the ring R, splitting a polygon, and merging crease patterns trivially requires $O(n)$ time. Contracting an edge requires $O(1)$ time. Thus, the main work of each recursive invocation is computing the time of the next event.

Let n denote the number of nodes in the input tree. To find the next event, we first compute the time at which each edge would contract (assuming no other event occurred first) in constant time per edge. The minimum is the candidate next contraction event. This takes $O(n)$ time. Then, for each pair of non-adjacent vertices, we compute the time at which Eq. 6.1 is satisfied to obtain a list E of candidate splitting events. We then sort the list, which takes $O(n^2 \log n)$ time [since there are $O(n^2)$ candidate splitting events]. We then look at the first candidate splitting event, compute a representation of the polygon at that event, and check whether the candidate splitting pair of vertices is still visible when the sweep reaches that point. This takes $O(n)$ time using standard techniques. If the candidate splitting pair is visible, then we have found the candidate next splitting event. Otherwise, we check the second candidate event in E, and so on. We continue until we have either found a candidate splitting event for which the pair is visible, or the event time of the candidate splitting event we are considering is greater than the event time of the next candidate contraction event. In the worst case, then, finding the time of the next event requires $O(n^3)$ time, since there are $O(n^2)$ candidate splitting events, and testing the visibility pair in each requires an additional $O(n)$ time. Thus one invocation of the algorithm takes $O(n^3)$ time total and $O(n^2)$ space.

To bound the running time of the algorithm, then, we need to bound the number of events that occur during the sweep. In order to reduce the number of events that occur, it is convenient to first compute only the traces of the corner vertices. The traces of the markers can be computed in a post-processing step. This is the method employed both in Lang's original paper [12] and our paper on faster implementations of the universal molecule algorithm for convex polygons [7]. By the same reasoning as in [7], the number of contraction and splitting events is $O(n)$ leading to an $O(n^4)$ time algorithm. We note, however, that the naive method described above recomputes almost all of the same candidate splitting events on each recursive call. Using the same techniques as in [7] this can be improved to $O(n^3 \log n)$ time.

7 Proof of Main Theorem

We can now proceed to the proof of:

Theorem 1.1 *Let P_T be a doubling-polygon for a tree T on a flat, disk-like piecewise-linear surface D. Then a Lang surface S constructed on T and isometric to P_T exists (and is unique) if and only if (T, P_T) is a geodesic Lang polygon.*

Proof Outline Let (T, P_T) be a geodesic Lang polygon. We show in Lemma 7.2 in the next section that the subdivision G returned by Algorithm 1 is equivalent to a Lang surface S_T constructed on T with boundary polygon P_T. In particular this proves that for any geodesic Lang polygon there exists a Lang surface constructed on T which has P_T as its boundary polygon, namely the Lang surface that is equivalent to the geodesic universal molecule of (T, P_T).

We have already established a correspondence in the other direction via Lemma 5.2, which shows that the boundary of each Lang surface is a geodesic Lang polygon.

It remains to establish a one-to-one correspondence, which we do by showing that there exists exactly one generalized sweep for any given Lang polygon, which implies that for a given Lang polygon (T, P_T), there exists a *unique* Lang surface S_T constructed on T that has P_T as its boundary. We prove this in Lemma 7.4 in Sect. 7.2.

Therefore, if we are given a geodesic Lang polygon (T, P_T), then there exists a Lang surface S_T constructed on T that is isometric to P_T, and by Lemma 5.2, if we start with a Lang surface S_T constructed on T and isometric to P_T, then (T, P_T) is a geodesic Lang polygon. □

7.1 Proof of Algorithm Correctness: Geodesic Universal Molecule Crease Patterns are Lang Surfaces

We prove that Algorithm 1 correctly simulates a generalized sweep of the input (T, P_T) and that the resulting subdivision G returned by the algorithm is equivalent to a Lang surface S_T constructed on T with boundary polygon P_T.

Lemma 7.1 *Algorithm 1 simulates a generalized sweep.*

Proof By definition the algorithm advances a parallel sweep in P_T and grows/shrinks the leaf arcs of T with the same speed assigned to each leaf arc as in the definition of a generalized sweep. We only need to show is that the sweep maintains the geodesic Lang property throughout this sweep. Otherwise, at some intermediate point the Lang property is violated in the simulated sweep between two consecutive events processed by the algorithm. But the algorithm always processes contraction events it encounters, and any splitting events for visible pairs (since Eq. 6.1 gives the time of a splitting event). Therefore, the Lang property was violated for a non-visible pair but not for a visible pair contradicting Lemma 3.3. This concludes the proof.

Lemma 7.2 *The subdivision G returned by Algorithm 1 on a geodesic Lang polygon (T, P_T) is the same subdivision of P_T into vertices, edges, and faces as a Lang surface S_T constructed on T with P_T as its boundary polygon.*

Proof By Lemma 7.1 the algorithm simulates a generalized sweep of (T, P_T). The trace of the vertices of the sweep together with the splitting edges introduced at splitting events induce the subdivision G. By Lemma 5.4, this generalized sweep is equivalent to an extrusion sweep of a Lang surface S_T constructed on T with P_T as its boundary polygon. By definition all the edges of S_T are given by the traces of the vertices of the extrusion sweep and the splitting segments introduced at splitting events. Thus G induces the same subdivision of P_T into vertices, edges, and faces as those given by the construction of S_T. □

7.2 Uniqueness

Thus far, we have seen that the vertices, edges, and faces of each flat Lang surface are defined by an extrusion sweep, which, by Lemma 5.3, is a generalized sweep of the surface's boundary polygon and tree. We have also seen that

any generalized sweep of a flat geodesic Lang polygon is equivalent to an extrusion sweep of some Lang surface. In Sect. 6, we gave an algorithm for producing at least some of the flat Lang surfaces, by simulating one particular generalized sweep of a Lang polygon, namely the one in which we always process splitting events, regardless of whether or not it is necessary to do so to maintain the geodesic Lang property on the sweep. We now show that this is the *only possible* generalized sweep of a flat geodesic Lang polygon. In other words, there is no "choice" to make. If we fail to split the polygon and tree at a potential splitting event, then whatever the resulting sweep is, it is not a generalized geodesic sweep. Summarizing,

Theorem 7.3 *Let* (T, P_T) *be a flat geodesic Lang polygon. Then there exists a unique generalized sweep of* (T, P_T).

We prove this presently but first note that as a direct consequence we have the following, which is the final step in the proof of the Main Theorem:

Lemma 7.4 *Let* (T, P_T) *be a flat geodesic Lang polygon. Then there exists a unique Lang surface* S_T *constructed on* T *that is isometric to* P_T.

Proof Assume not. Then there are at least two different Lang surfaces with P_T as its boundary, and thus two different extrusion sweeps. But by Lemma 5.3, these constitute two different generalized sweeps of (T, P_T), contradicting Theorem 7.3. □

The remainder of this section concerns proving Theorem 7.3. Our main work is to show that when a generalized sweep encounters a potential splitting event, it *must actually split* in order to maintain the geodesic Lang property. *We note that this proof is significantly more involved than in the convex case in* [5]. The proof is based on elementary geometry and vector calculus and requires a careful case analysis of 36 possible cases, only one of which is the convex case. We first outline the proof of Theorem 7.3 and then fill in the details in the theorems and lemmas that follow.

Proof Outline of Theorem 7.3 The possibility of the existence of multiple generalized sweeps for the same geodesic Lang polygon (T, P_T) comes from our distinction between potential and actual splitting events. We have thus far allowed that, as long as the geodesic Lang property is maintained, we do not care if the sweep is *actually split* at each potential splitting event. There are two possibilities we need to consider regarding potential splitting events that may give rise to multiple generalized sweeps for the same geodesic Lang polygon.

First, it may be the case that if multiple potential splitting events occur simultaneously, then actually splitting across one event removes one of the others as a potential splitting event. Theorem 7.5 shows that this does not occur. Thus, when we arrive at simultaneous potential splitting events $(\mathbf{u}, \mathbf{v})$ and $(\mathbf{w}, \mathbf{x})$, splitting at one, say $(\mathbf{u}, \mathbf{v})$ leaves the other as a potential splitting event. In other words both $\mathbf{w}$ and $\mathbf{x}$ are in the same split polygon/tree after splitting at $(\mathbf{u}, \mathbf{v})$ which entails that we still have a potential splitting event, since splitting does not effect the distances between $\mathbf{w}$ and $\mathbf{x}$ in the either the polygon or tree.

Second, it may be the case that we can simply ignore some potential splitting events. This would mean that there is a potential splitting event for a pair $(\mathbf{u}, \mathbf{v})$ at some time t such that immediately before the event and immediately after the event the geodesic Lang property is satisfied *if we do not split*. Thus we can choose to either actually split or not to obtain different generalized splitting events. We show in Theorem 7.7 that this is not the case–we *must split* at potential splitting events.

Thus, together with Theorems 7.5 and 7.7, we have that the generalized sweep of a geodesic Lang polygon on a flat surface is unique. □

Event Order Does Not Matter. We now show that when simultaneous potential splitting events occur, choosing to split across one of the events does not invalidate any of the others as potential splitting events. This completes the first part of the proof of Theorem 7.3 above.

Theorem 7.5 *Let* $(\mathbf{u}, \mathbf{v})$ *and* $(\mathbf{w}, \mathbf{x})$ *be simultaneous potential splitting events encountered at some time in a generalized sweep of a Lang polygon. Then after actually splitting for one, say* $(\mathbf{u}, \mathbf{v})$*, the other* $(\mathbf{w}, \mathbf{x})$ *remains a potential splitting event.*

Proof Let T and P_T denote the tree and sweeping polygon at the time of the event. Let (T_L, P_L) and (T_R, P_R) denote the split polygons obtained by splitting (T, P_T) at $(\mathbf{u}, \mathbf{v})$. Then without loss of generality, either both $\mathbf{w}$ and $\mathbf{x}$ are in P_L or $\mathbf{w}$ is in P_L and $\mathbf{x}$ is in P_R. In the first case, the distances $d_{P_T}(\mathbf{w}, \mathbf{x})$ and $d_T(w, x)$ are not changed by the split and thus $(\mathbf{w}, \mathbf{x})$ remains a potential splitting event. In the second case, we note that the cyclic ordering of $\mathbf{u}, \mathbf{v}, \mathbf{w}, \mathbf{x}$ along the boundary of P_T is (without loss of generality) $\mathbf{u}, \mathbf{w}, \mathbf{v}, \mathbf{x}$. In other words, the pairs $(\mathbf{u}, \mathbf{v})$ and $(\mathbf{w}, \mathbf{x})$ *cross*, which contradicts Lemma 7.6 below. □

The main work of the proof above is Lemma 7.6 below, which shows that potential splitting events do not cross. We note that the proof is the same as Lemma 12 in [5] since it relies only on properties of the tree and the triangle inequality on the geodesic distances, and refer the reader to our previous paper for the argument. Thus we have:

Lemma 7.6 *Potential splitting pairs do not cross in a geodesic Lang polygon.*

The Sweep Must Split at All Potential Splitting Events We now show that when a generalized sweep encounters a potential splitting event, *it necessarily actually splits at the event*. Otherwise, we will show, the sweep fails to maintain the geodesic Lang property and thus is not a generalized sweep of a geodesic Lang polygon. This completes the remainder of the proof of Theorem 7.3. The proof is significantly more involved than in the convex case (Lemma 16 in [5]). This is because in the convex case, it is fairly straightforward to show that the distances in the plane between vertices of the sweeping polygon always decreases at a rate faster than the corresponding distances in the tree. In the present case, however, distances between points in the sweeping polygon and in the tree may be either increasing or decreasing (depending on the geometry), and it is not necessarily the case that tree distances decrease slower than distances in the sweeping polygon. For this reason, a more careful case analysis is needed. We now prove:

Theorem 7.7 *A generalized sweep of a Lang polygon always splits at each potential splitting event.*

Proof Suppose that we reach a potential splitting event $(\mathbf{a}_i, \mathbf{a}_j)$. We prove the theorem by comparing the rates at which the distances are changing in the sweeping polygon and tree. Let $d_S(t)$ denote the distance at time t between $\mathbf{a}_i$ and $\mathbf{a}_j$ in the sweep and $d_T(t)$ denote the distance between the corresponding leaf nodes a_i and a_j in the kinetic tree. Let $d(t)$ denote the difference between them, i.e. $d(t) = d_S(t) - d_T(t)$. For simplicity, we will shift the event times so that the event occurs at time $t = 0$, and thus $d_S(0) = d_T(0)$, or equivalently $d(0) = 0$. Let $\Delta t > 0$ be a small value near 0. Just before the event (i.e. at time $-\Delta t$), the geodesic Lang property holds, and so $d_S(-\Delta t) > d_T(-\Delta t)$, and thus $d(-\Delta t) > 0$. We want to show that just after the event the geodesic Lang property is violated, i.e. $d(\Delta t) < 0$ for all small enough Δt. To do this, we need to show that $d(t)$ is not at a local minimum at $t = 0$. It suffices to show that it is not at a critical point, meaning that its derivative $d'(0) = d'_S(0) - d'_T(0) \neq 0$, or equivalently $d'_S(0) \neq d'_T(0)$. We prove this in Lemma 7.8 below.

Now assume that a generalized sweep encounters a potential splitting event but does not split. By Theorem 7.5, we have that the potential event cannot be removed by actually splitting for some other simultaneously occurring potential splitting event. But then, by the discussion above we have that immediately after the event, the geodesic Lang property is violated, contradicting that our sweep is a generalized sweep. □

Lemma 7.8 *The instantaneous rate of change in the geodesic distance between two non-consecutive visible corners* $\mathbf{a}_i$ *and* $\mathbf{a}_j$ *in the parallel sweep is not equal to the instantaneous rate of change in the corresponding distance in the kinetic tree.*

Proof Set-Up and Outline Here we only outline the proof of Lemma 7.8 and give the geometric set-up. The details of the proof are then organized into the lemmas below. We then give the full proof at the end this section starting from the paragraph titled "Proof of Lemma 7.8".

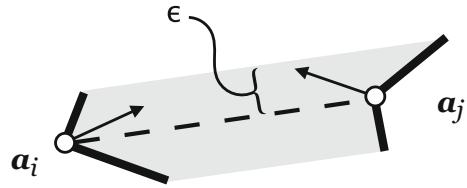

Fig. 13 Flattening out of a small ϵ band around the visibility segment between $\mathbf{a}_i$ and $\mathbf{a}_j$

Initial Set-Up As in Theorem 7.7, we denote the distance function between $\mathbf{a}_i$ and $\mathbf{a}_j$ in the sweep by $d_S(t)$ and the distance function in the tree between a_i and a_j by $d_T(t)$. To prove the lemma, we show that the instantaneous rate of change in the geodesic distance, $d'_S(0)$ is not equal to the instantaneous rate of change, $d'_T(0)$ in the tree. In principle our argument holds for all values of t so long as $(\mathbf{a}_i, \mathbf{a}_j)$ is a visible pair.

Overview Instead of deriving a closed form for $d'_S(t)$ and $d'_T(t)$, we show in Lemmas 7.9 and 7.10 how to determine the value of $d'_S(0)$ and $d'_T(0)$ at $t = 0$ geometrically. We show that the values of $d'_S(0)$ and $d'_T(0)$ are determined by two vectors defined at each vertex, which we label V_i and W_i at $\mathbf{a}_i$ and V_j and W_j at $\mathbf{a}_j$. Then, by analyzing the relative magnitudes of V_i and W_i and the relative magnitudes of V_j and W_j we prove that $d'_S(0) \neq d'_T(0)$. To do this we first initiate a study of the relative magnitudes of V_i and W_i at $\mathbf{a}_i$, from which we derive 6 distinct cases, labelled A–F. These depend on whether $\mathbf{a}_i$ is convex or not in the polygon and the angle made between the edges incident to $\mathbf{a}_i$ and the visibility segment $\mathbf{a}_i\mathbf{a}_j$. We then complete the proof of the lemma by showing in all 36 possible cases (where both vertex $\mathbf{a}_i$ and vertex $\mathbf{a}_j$ may be any of the six cases A–F), $d'_S(0) \neq d'_T(0)$. □

Before we fill in the details of the proof, we fix some useful notation and simplify the discussion by providing a local view around the visibility edge $\mathbf{a}_i\mathbf{a}_j$.

Notation In the remainder of this section we use upper cased non-bold type with subscripted i or j to denote vectors, such as U_i or U_j, defined at $\mathbf{a}_i$ and $\mathbf{a}_j$ resp. We denote the magnitude of a vector U_i by its lower-case $u_i = ||U_i||$.

Local View In order to simplify the discussion that follows we "flatten out" the polygon and sweep in the plane. This flat realization, as we have seen, may have self-intersections; however, if we restrict ourselves to a small enough patch, say all points within some small ϵ distance of the visibility segment between $\mathbf{a}_i$ and $\mathbf{a}_j$, then the resulting *flattened ϵ-patch* is realized in the plane as a small planar region without self intersections. See Fig. 13 for an example. There, the dashed line denotes the visibility segment, the dotted line denotes the sweep, and the two arrows denote the motion vectors of $\mathbf{a}_i$ and $\mathbf{a}_j$ in the unit-speed parallel sweep.

In the remainderof this section we use this "local" view of the flat realization in the plane, which allows us to use elementary plane geometry to analyze the geometry near the visibility segment.

Deriving the Vectors V_i and W_i Let U_i denote the instantaneous velocity vector of $\mathbf{a}_i$. By definition, U_i points along the interior angle bisector at $\mathbf{a}_i$. We now use U_i to define two vectors, V_i and W_i at $\mathbf{a}_i$. Project U_i onto the visibility segment $\mathbf{a}_i\mathbf{a}_j$. Let L denote the supporting line through one of the edges incident to $\mathbf{a}_i$. Project U_i onto L to obtain W_i. Note that because U_i points along an angle bisector, and because we are really only interested in the magnitude w_i and not the direction, it does not matter which edge incident to $\mathbf{a}_i$ is chosen to construct L (the magnitude w_i is the same regardless of the choice). We define vectors U_j, V_j, and W_j at $\mathbf{a}_j$ similarly. Examples are shown in Fig. 14. The left shows the convex case. The middle shows a reflex case in which $\mathbf{a}_i$ is moving towards $\mathbf{a}_j$. The right shows a reflex case in which $\mathbf{a}_i$ is moving away from $\mathbf{a}_j$. U_i denotes the motion vector of $\mathbf{a}_i$ along the interior angle bisector in the sweep. V_i is the projection of U_i onto the line between $\mathbf{a}_i$ and $\mathbf{a}_j$, and W_i is the projection of U_i onto the line supporting one of its edges.

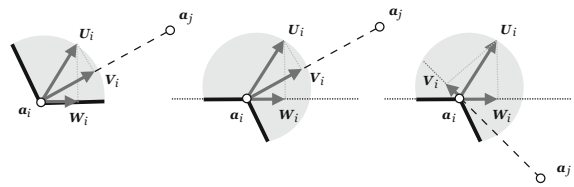

Fig. 14 An illustration of the vectors defined in the proof of Lemma 7.8 for the vertex $\mathbf{a}_i$ in three different situations

Table 1 A summary of the relationship between the magnitudes v_i and w_i of the vectors V_i and W_i from Fig. 14 organized by cases A–F, and the sign in front of each in $d'_S(0)$ and $d'_T(0)$ for all possible cases of the vertex $\mathbf{a}_i$

Case	$v_i ? w_i$	Sign of v_i in $d'_S(0)$	Sign of w_i in $d'_T(0)$
A	$v_i > w_i$	$-$	$-$
B	$v_i < w_i$	$+$	$+$
C	$v_i < w_i$	$v_i = 0$	$+$
D	$v_i < w_i$	$-$	$+$
E	$v_i = w_i$	$-$	$+$
F	$v_i > w_i$	$-$	$+$

What Remains In Lemma 7.9 we show how $d'_S(0)$ relates to the magnitudes v_i and v_j. In Lemma 7.10 we show how $d'_T(0)$ relates to the magnitudes of w_i and w_j. We end with the proof of Lemma 7.8.

Lemma 7.9 *The instantaneous rate of change in the distance between* $\mathbf{a}_i$ *and* $\mathbf{a}_j$ *in the polygon is given by*

$$d'_S(0) = \pm v_i \pm v_j \tag{7.1}$$

where the sign in front of v_i *(resp.* v_j*) is '+' if* V_i *points towards* $\mathbf{a}_j$ *(resp.* V_j *points towards* $\mathbf{a}_i$*), otherwise the sign is '-'.*

Proof The proof follows from elementary vector calculus. □

We now derive $d'_T(0)$:

Lemma 7.10 *The instantaneous rate of change in the distance between* a_i *and* a_j *in the tree is given by*

$$d'_T(0) = \pm w_i \pm w_j \tag{7.2}$$

where the sign in front of w_i *(resp.* w_i*) is '-' if* $vertex a_i$ *(resp.* $\mathbf{a}_j$*) is convex in the polygon, '+' otherwise.*

Proof The vector W_i is the orthogonal component of $\mathbf{a}_i$'s instantaneous motion towards the other endpoint of one of the edges incident to $\mathbf{a}_i$. The proof then follows from the fact that we defined the speeds at the leaf arcs so as to maintain the length in the tree between edges in the sweeping polygon and their corresponding leaf arcs. □

Characterizing the Relative Magnitudes of v_i and w_i We now characterize the relative magnitudes of V_i and W_i, and the signs in front of each in Eqs. 7.1 and 7.2. There are six possible cases, which we label A–F, which depend on whether $\mathbf{a}_i$ is convex or not, and the angle made by the visibility segment $\mathbf{a}_i\mathbf{a}_j$ with the edges incident to $\mathbf{a}_i$. The construction of cases A–F, detailed shortly, is illustrated in Fig. 15 and the relative magnitudes of v_i and w_i in each case is summarized in Table 1 below. A is the convex case, cases B–F are illustrated in Fig. 15. Those for $\mathbf{a}_j$ are similar. Using the table together with Eqs. 7.1 and 7.2 allows us to determine the values of $d'_S(0)$ and $d'_T(0)$ based on which cases A–F are $\mathbf{a}_i$ and $\mathbf{a}_j$. For instance, if $\mathbf{a}_i$ is case B and $\mathbf{a}_j$ is case D, then using the table and the two equations we see that $d'_S(0) = v_i - v_j$ and $d'_T(0) = w_i + w_j$.

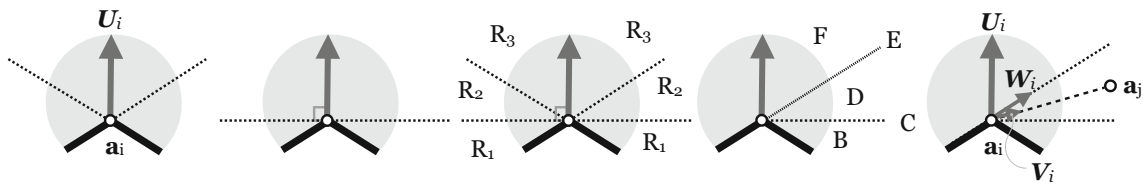

Fig. 15 Constructing the five non-convex cases B–F used to define the cases in the case analysis in the proof of Lemma 7.8. *From left to right, first:* extend the *lines* supporting the two edges incident to a vertex (*dashed*). *Second:* the *line* (*dashed*) orthogonal to the interior angle bisector (*gray vector*). *Third:* the wedge regions R_1 through R_3 defined by the *lines* extending the edges and the *line* orthogonal to the interior angle bisector (the wedges to the *left* of the bisector are symmetric to those on the *right*). *Fourth:* the labelling of the cases to the *right* of the interior angle bisector (the *left* is symmetric). *Fifth:* an example where the position of a_j follows in region R_2, making a_i case D. In this case the length of W_i is greater than the length of V_i

Constructing Cases A–F Case A is when $\mathbf{a}_i$ is convex, cases B–F are when $\mathbf{a}_j$ is non-convex. To construct cases B–F, first extend lines through the edges incident to $\mathbf{a}_i$. Next construct the line through $\mathbf{a}_i$ perpendicular to the interior angle bisector of $\mathbf{a}_i$. See the first two parts of Fig. 15. These three lines together with the interior angle bisector divide the wedge around $\mathbf{a}_i$ into six wedge "slices". Those on the left side of the angle bisector are symmetric to those to the right, so we label them R_1, R_2, and R_3 symmetrically and in the remainder concentrate on the right side—see the center of Fig. 15. Case B is when $\mathbf{a}_j$ falls in the wedge R_1. Case C is when it falls on the line between R_1 and R_2. Case C is when it falls in R_2. Case E is when it falls on the line between R_2 and R_3. Case F is when it falls in R_3. We now have:

Lemma 7.11 *Table 1 summarizes the relationship between v_i and w_i and the signs in front of each in Eqs. 7.1 and 7.2.*

Proof The lines extended in the construction of regions R_1, R_2, and R_3 are precisely those where relative magnitudes and the signs in Eq. 7.1 change. Extending the lines through the edges in the construction of R_1 through R_2 divides the plane into four regions (see the left-most illustration in Fig. 15). One is outside the the polygon. The next two are incident to the two edges incident to $\mathbf{a}_i$. In these, by elementary geometry it follows that $v_i < w_i$. In the remaining, $v_i > w_i$, and if $\mathbf{a}_j$ lies on the line through one of the edges then $v_i = w_i$. The line through $\mathbf{a}_i$ perpendicular to the interior angle bisector is the dividing line such that if $\mathbf{a}_j$ is below it (i.e. in R_1), then $\mathbf{a}_i$ is moving away from $\mathbf{a}_j$ (and hence v_i has a '+' in Eq. 7.1). If $\mathbf{a}_j$ is on the line, then $\mathbf{a}_i$ is moving perpendicularly relative to $\mathbf{a}_i$ and $\mathbf{a}_j$ and so $v_i = 0$. Otherwise $\mathbf{a}_j$ is moving towards it. The table simply summarizes these facts. □

Proof of Lemma 7.8 We now complete the proof of Lemma 7.8 by showing that for all possible cases A–F of $\mathbf{a}_i$ and all possible cases A–F of $\mathbf{a}_j$, the instantaneous rate of change in the geodesic distance $d'_S(0)$ is not equal to the instantaneous rate of change in the tree.

We have 36 cases to consider, depending on which cases A–F are the two vertices $\mathbf{a}_i$ and $\mathbf{a}_j$. Each case is labeled with the two case letters for $\mathbf{a}_i$ and $\mathbf{a}_j$. For instance, if vertex $\mathbf{a}_i$ is B and $\mathbf{a}_j$ is D, then we label it BD. Note that symmetric cases use the same proof, so we only list cases in lexicographical order (BD and DB are the same so we use BD). In each case we start by using Table 1 to derive $d'_S(0)$. We then use the relationship between v_i and w_i and the relationship between v_j and w_j to show that $d'_S(0) \neq d'_T(0)$ (which is found by plugging in the appropriate values from Table 1 into Eq. 7.2).

Below we prove each case on a single line, however, to give the reader a full sense of the line of proof we do one expanded case here, the case where $\mathbf{a}_i$ is B and $\mathbf{a}_j$ is E (i.e. case BE). Looking up B for $\mathbf{a}_i$ and E $\mathbf{a}_j$ in Table 1 and plugging the appropriate values into Eqs 7.1 and 7.2 we have that $d'_S(0) = v_i - v_j$ and $d'_T(0) = w_i + w_j$. We now start with $d'_S(0)$ and show that it is not equal to $d'_T(0)$. $d'_S(0) = v_i - v_j < w_i - v_j$ since by Table 1 $v_i < w_i$ for case B. $w_i - v_j < w_i + w_j$ since all magnitudes are positive, and thus subtracting v_i from w_i is less than adding any positive value to w_i. But $w_i + w_j = d'_T(0)$, and thus $d'_S(0) < d'_T(0)$. We now show the full case by case analysis. All inferences are either derived from Table 1 as above, or follow from the fact that all magnitudes are positive.

- AA: $d'_S(0) = -v_i - v_j < -w_i - v_j < -w_i - w_j = d'_T(0)$.
- AB, AC: $d'_S(0) = -v_i + v_j < -w_i + v_j < -w_i + w_j = d'_T(0)$.
- AD, AE, and AF: $d'_S(0) = -v_i - v_j < -w_i - v_j < -w_i + w_j = d'_T(0)$.
- BB, BC: $d'_S(0) = v_i + v_j < w_i + v_j < w_i + w_j = d'_T(0)$.
- BD: $d'_S(0) = v_i - v_j < v_i + w_j < w_i + w_j = d'_T(0)$.
- BE, BF: $d'_S(0) = v_i - v_j < w_i - v_j < w_i + w_j = d'_T(0)$.
- CC: $d'_S(0) = 0$. $d'_T(0) = w_i + w_j > 0$.
- CD, CE, CF: $d'_S(0) = -v_j < w_j < w_i + w_j = d'_T(0)$.
- DD, DE, DF, EE, EF, FF: $d'_S(0) = -v_i - v_j < 0 < w_i + w_j = d'_T(0)$.

The remaining cases are symmetric. In all cases we have shown that $d'_S(0) \neq d'_T(0)$, which proves that $d(t) = d_S(t) - d_T(t)$ is not at a local minimum at $t = 0$ and thus to proceed in the sweep constitutes a violation of the geodesic Lang property. □

Summary We have shown that the instantaneous rate of change in the geodesic distance between $\mathbf{a}_i$ and $\mathbf{a}_j$ is not equal to the instantaneous rate of change in the tree distance between a_i and a_j. In particular, this shows that the derivative $d'(0) = d'_S(0) - d'_T(0)$ does not vanish, meaning that $d(0)$ is not a critical point. Since $d(0)$ is not a critical point, then to continue the sweep past a splitting event *without splitting* results in a violation of the geodesic Lang property. This completes the proofs of Theorems 7.3 and 7.7. Thus for any given flat geodesic Lang polygon, there exists a unique generalized sweep fro the polygon. This sweep is precisely the sweep simulated by the geodesic universal molecule algorithm. Furthermore, together with Lemmas 5.4 and 7.1 this entails that there is a unique geodesic Lang surface constructed on T having P_T as its boundary and the subdivision of P_T into vertices, edges, and faces given by S_T is the geodesic universal molecule of P_T. This in turn is the final ingredient in the proof of the Main Theorem, Theorem 1.1.

8 Conclusion

In this paper we generalized the universal molecule algorithm to geodesic Lang polygons which form the boundary of piecewise-linear disk-like surfaces in $\mathbf{R}^3$ with zero-curvature. Restricted to simple, non-convex polygons in the plane our algorithm extends the TreeMaker algorithm to cases where before it could not produce an output. A further open problem is an extension of the algorithm to surfaces of non-zero curvature. Our Lang surfaces are more general and can be used to construct surfaces with non-zero curvature. Can the algorithm be extended to compute these from the boundary polygon? If we are given a geodesic Lang polygon drawn on a surface with singular points of non-zero curvature, can we always compute a geodesic universal molecule on the polygon that is equivalent to some Lang surface?

References

1. Aichholzer, O., Alberts, D., Aurenhammer, F., Gärtner, B.: A novel type of skeleton for polygons. J. Univers. Comput. Sci. **1**(12), 752–761 (1995)
2. Aloupis, G., Demaine, E.D., Langerman, S., Morin, P., O'Rourke, J., Streinu, I., Toussaint, G.T.: Edge-unfolding nested polyhedral bands. Comput. Geom. Theory Appl. **39**(1), 30–42 (2008)
3. Alperin, R., Hayes, B., Lang, R.: Folding the hyperbolic crane. Math. Intell. **34**(2), 38–49 (2012)
4. Aurenhammer, F.: Voronoi diagrams—a survey of a fundamental geometric data structure. ACM Comput. Surv. (CSUR) **23**(3), 345–405 (1991)
5. Bowers, J., Streinu, I.: Lang's universal molecule algorithm. Ann. Math. Artif. Intell. 1–30 (2014)
6. Bowers, J.C., Streinu, I.: Rigidity of origami universal molecules. In: Ida, T., Fleuriot, J.D. (eds.) Automated Deduction in Geometry. Lecture Notes in Computer Science, vol. 7993, pp. 120–142. Springer, New York (2012)
7. Bowers, J.C., Streinu, I.: Computing origami universal molecules with cyclic tournament forests. In: Proc. 15th Intern. Symp. on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC'13), pp. 42–52. IEEE (2013)

8. Demaine, E.D., Demaine, M.L.: Computing extreme origami bases. Technical Report CS-97-22, Dept. of Computer Science, University of Waterloo, Waterloo (1997)
9. Demaine, E.D., Fekete, S.P., Lang, R.J.: Circle packing for origami design is hard. In: Wang-Iverson, P., Lang, R.J., Yim, M. (eds.) Origami 5: Fifth International Meeting of Origami Science, Mathematics, and Education, pp. 609–626. Taylor and Francis, London (2011)
10. Demaine, E.D., O'Rourke, J.: Geometric Folding Algorithms: Linkages, Origami, and Polyhedra. Cambridge University Press, Cambridge (2007)
11. Ida, T., Ghourabi, F., Takahashi, K.: Formalizing polygonal knot origami. J. Symb. Comput. **69**, 93–108 (2015)
12. Lang, R.J.: A computational algorithm for origami design. In: Proceedings of the 12th Annual ACM Symposium on Computational Geometry, pp. 98–105 (1996)
13. Lang, R.J.: Treemaker 4.0: A program for origami design. http://www.langorigami.com (1998)
14. Tachi, T.: Origamizing polyhedral surfaces. IEEE Trans. Vis. Comput. Graph. **16**(2), 298–311 (2010)
15. Tanaka, H.: Bi-stiffness property of motion structures transformed into square cells. Proc. R. Soc. A Math. Phys. Eng. Sci. **469**(2156), 20130063 (2013)
16. Wu, W., You, Z.: Modelling rigid origami with quaternions and dual quaternions. Proc. R. Soc. A Math. Phys. Eng. Sci. **466**(2119), 2155–2174 (2010)
17. Wu, W., You, Z.: A solution for folding rigid tall shopping bags. In: Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences, vol. 467, pp. 2561–2574. The Royal Society (2011)
18. Yasuda, H., Yein, T., Tachi, T., Miura, K., Taya, M.: Folding behaviour of Tachi–Miura polyhedron bellows. Proc. R. Soc. A Math. Phys. Eng. Sci. **469**(2159), 20130351 (2013)