www.cadanda.com
Computer-Aided Design AND APPLICATIONS

Straight Skeletons and Mitered Offsets of Polyhedral Terrains in 3D

Martin Held[1] , Peter Palfrader[2]

[1]FB Computerwissenschaften, University of Salzburg, Austria, held@cs.sbg.ac.at
[2]FB Computerwissenschaften, University of Salzburg, Austria, palfrader@cs.sbg.ac.at

Corresponding author: Martin Held, held@cs.sbg.ac.at

Abstract. We present a simple algorithm to compute the straight skeleton and mitered offset surfaces of a polyhedral terrain in 3D, i.e., of a z-monotone piecewise-linear surface. Like its 2D pedant, the 3D straight skeleton is the result of a wavefront propagation process, which we simulate in order to construct the skeleton in (worst-case) time $\mathcal{O}(n^4 \log n)$, where n is the number of vertices of the terrain. Any mitered offset surface can then be obtained from the skeleton in time linear in the combinatorial size of the skeleton.

Keywords: Straight Skeleton, Monotone Surface, Mitered Offset, 3D
DOI: https://doi.org/10.14733/cadaps.2019.611-619

1 INTRODUCTION

1.1 Offsetting

The task of offsetting is to compute an expanded or contracted version of an original solid object. To offset a solid S outwards by a (positive) distance t one adds to the solid all the points exterior to S that lie within a distance of at most t from the boundary of S. For an inwards offset one subtracts from S all the interior points that lie within a distance of at most t from its boundary. This concept is readily adapted to (oriented) surfaces by replacing the terms "inwards" and "outwards" by the terms "left" and "right" and considering only those points whose distance to the original surface equals t. If the boundary surface is piecewise-linear – i.e., if it consists only of patches formed by polygonal facets – then the offset surface will consist of expanded (or contracted) copies of the polygonal facets of the original surface such that the distance between every original facet and its translated copy equals t. Depending on the actual application, the offset surface of a piecewise-linear surface may be required to consist entirely of piecewise-linear facets again, or it might also contain cylindrical and spherical patches. In the first case we get a mitered offset whereas the second case corresponds to a constant-radius offset. Following the classification by Pottmann et al. [19], a mitered offset can also be seen as a "face-offset".

Offsetting is a fundamental operation both in CAD as well as in several application areas. Applications of offsetting comprise a wide range, such as cutter radius compensation for gouge-free surface machining, planning collision-free paths for robots, ensuring a minimum clearance between two conductive objects, shelling 3D models to produce a hollowed-out shell (e.g., in order to reduce the cost of rapid prototyping), generating volume boundary layer meshes and thickening surfaces (e.g., for solid fabrication), or modeling the removal or addition of a layer of material (such as soil or snow) of constant thickness in a GIS application.

The two predominant approaches to obtaining an offset surface of a (triangulated) piecewise-linear surface are as follows: (1) One considers a family of vectors of length t that originate at vertices of the input surface and point into the requested offset direction and uses these vectors to generate vertices of the offset facets. (2) One computes offset facets by translating the individual input facets and "somehow" closes the gaps that occur between the offset facets. See, e.g., Bolaños et al. [7], Kim, Yang et al. [15, 16], Liu and Wang [17], Wen et al. [22], or Yi et al. [23] for a selection of recent variants of these approaches. The proper handling of those gaps forms the obvious crux of the latter approach, while the first approach bears the risk of producing offsets with non-constant offset distance. Both approaches handle only the local geometry, and one still needs to ensure that no global self-intersections of the offset surface occur, such as done by Jung et al. [14]. Offsetting algorithms are also known for restricted types of input; see, e.g., Wang and Liu's work [21] on offsetting hexagonal meshes.

Despite of literally hundreds of publications on offsetting and of wide-spread practical use, the problem of offsetting polyhedral objects has received rather limited attention by theory. This is even more surprising once one realizes that the offsetting problem is not even well defined for mitered offsets: Suppose that one attempts to compute a mitered offset surface of a piecewise-linear surface $\mathcal{T}$ by moving its polygonal facets in a self-parallel manner and at unit speed, thereby maintaining incidences. As the polygonal facets move, various kinds of changes of the geometry, combinatorial nature and topology will occur. However, offset surfaces generated in this way may be ambiguous. That is, in general there exists more than one surface that can be regarded as the mitered offset surface of $\mathcal{T}$. See Aurenhammer and Walzl [3], who credit Erickson for this observation.

Nowadays it is generally uncontested that computing an appropriate skeletal data structure as preprocessing constitutes the premier choice for offsetting 2D polygons with regard to both speed and reliability. See, e.g., constant-radius offsets based on Voronoi diagrams [9] and mitered offsets based on straight skeletons [18]. Hence, it seems natural to apply a similar approach to mitered offsetting in three dimensions and to resort to 3D pendants of 2D straight skeletons.

However, very little is known on such a suitable preprocessing for computing (families of) mitered offsets in 3D. Aurenhammer and Walzl [3] provide the first complete analysis of straight skeletons in three dimensions. However, it is not obvious how to implement their approach in full generality such that it can cope with arbitrary polyhedral objects in 3D.

1.2 Straight Skeletons

The concept of straight skeletons was introduced to computational geometry over 20 years ago by Aichholzer et al. [2]. Let P be a simple polygon in the plane and consider the following process. At time $t := 0$, each edge of P starts to move towards the interior of P at unit speed in a self-parallel manner, thereby maintaining incidences. The set of moving edges forms one or more polygons, called the wavefront $\mathcal{W}_P(t)$ of P at time t, see Figure 1. Note that each edge of $\mathcal{W}_P(t)$ is at all times at orthogonal distance t to its corresponding edge of P.

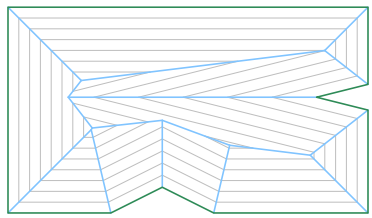

Figure 1: The straight skeleton $\mathcal{S}(P)$ (blue) of an input polygon P (green and bold) is the union of the traces of the vertices of P as it shrinks. Several instances (wavefronts) of the shrinking polygon are shown in gray.

The wavefront needs to be updated at times to remain a set of simple polygons: As edges shrink to zero length (*edge event*), they are removed, and edges are split and incidences updated when a previously non-incident vertex moves into their interior (*split event*). If the polygon is not in general position then also more complex interactions are possible. The straight skeleton $\mathcal{S}(P)$ of P is then defined as the geometric graph whose edges consist of the traces of wavefront vertices over the propagation process, see Figure 1.

A mitered offset of P at offset distance t corresponds to the wavefront $\mathcal{W}_P(t)$ at time t. Mitered offsets are inherently linked to the straight skeleton: Given the straight skeleton $\mathcal{S}(P)$ of a polygon P with n vertices, any mitered offset can be constructed in $\mathcal{O}(n)$ time and space [18]. We note that obtaining a mitered offset of P based on $\mathcal{S}(P)$ is not just easy to implement but it is also reliable and numerically stable.

Variations of the straight skeleton problem in the plane have also been investigated, by generalizing the input to arbitrary planar straight line graphs or by adding multiplicative or additive weights to input edges [1, 6, 8, 11]. The algorithm with the currently best worst-case complexity for computing the 2D straight skeleton of an arbitrary simple polygon is due to Eppstein and Erickson [8] and requires both $\mathcal{O}(n^{17/11+\varepsilon})$ time and space, for an arbitrarily small but positive ε. Better runtime bounds can be obtained when restricting the input to monotone polygons: Biedl et al. [5] present a simple and easy-to-understand algorithm which requires $\mathcal{O}(n \log n)$ time and linear space to compute the straight skeleton of a monotone polygon. That is, restricting the straight skeleton problem to monotone input allows for a more efficient and significantly simpler algorithm than required for handling arbitrary input. We attempt to mirror this in three dimensions.

2 STRAIGHT SKELETON IN 3D

Straight skeletons of polyhedral objects in 3D were studied by Barequet et al. [4] and, more recently, by Aurenhammer and Walzl [3]. However, while combinatorial complexities have been established for the straight skeleton of polytopes, no runtime bounds have been investigated. Furthermore, it is not obvious how to implement Aurenhammer and Walzl's approach in full generality such that it can cope with arbitrary polyhedral objects in 3D.

In this work we consider straight skeletons and mitered offsets of polyhedral terrains in 3D. As usual, a polyhedral terrain $\mathcal{T}$ is a piecewise-linear, continuous function of two variables. E.g., triangulated irregular networks (TINs) known to GIS form polyhedral terrains. By definition, a polyhedral terrain is monotone with respect to the xy-plane of $\mathbb{R}^3$. That is, every line parallel to the z-axis intersects a polyhedral terrain $\mathcal{T}$ in at most one point. To simplify matters, we assume that $\mathcal{T}$ is defined over all of $\mathbb{R}^2$ and that all facets are simply-connected. Furthermore, we assume that $\mathcal{T}$ is in general position: No more than four supporting planes of the facets of $\mathcal{T}$ shall be tangent to a common sphere, and the degree of any vertex of $\mathcal{T}$ shall be at most a constant k. (Restricting the domain of $\mathcal{T}$ to a convex region in $\mathbb{R}^2$ would also be good enough for our purposes.)

2.1 Wavefront Propagation and Events

We consider the wavefront propagation of a polyhedral terrain $\mathcal{T}$. Similar to the 2D setting, the wavefront consists of wavefront facets which are at orthogonal distance t to their corresponding input facets at all times.

Formally, let f be a facet of $\mathcal{T}$, let $\bar{H}_f$ be its supporting plane, and let $\vec{n}_f$ be the unit normal of f with positive z-coordinate. Then we define the offset-supporting plane at distance t to be $H_f(t) := \bar{H}_f + t \cdot \vec{n}_f$. The wavefront, just like the input $\mathcal{T}$, is a continuous, piecewise-linear surface, i.e., a polyhedral terrain. Its facets at time t are embedded in the offset-supporting planes $H_f(t)$ of all facets f of $\mathcal{T}$.

Initially, at time $t := 0$, the wavefront $\mathcal{W}_\mathcal{T}(t)$ is identical to $\mathcal{T}$. When the propagation process starts, all facets of the wavefront move upwards, in positive z-direction. During this propagation, the wavefront remains piece-wise linear and continuous, and incidences are retained where possible.

For the initial offset at time $t := \delta$, for any sufficiently small $\delta > 0$, retaining the combinatorial structure is possible along edges. Furthermore, locally at vertices of degree three, a mitered offset of the same combinatorial structure is possible. However, at vertices of degree four or higher, any offset, even at an infinitesimally small δ, will generally have a combinatorial structure different from the input: The offset surface consists locally of several degree-three vertices that arise from the offsets of the planes incident at the input vertex of higher degree. This is due to the fact that the offset facets of four (or more) triangular input facets need not intersect in a point. See Aurenhammer and Walzl [3] for more details.

As the wavefront propagation continues, the combinatorial structure of the surface has to be updated, and the set of wavefront vertices and their trajectories change at discrete points in time at so-called *events*, when four or more wavefront facets pass through a common point. Aurenhammer and Walzl [3] differentiate between events that change the topology of the offset and events that merely change its geometry. They call the first class *solid events*, which includes *splitting events*, where the polytope disconnects, and *piercing events*, where a vertex runs into a facet. However, since our wavefront surface is z-monotone and continuous, these events cannot occur and we will only observe

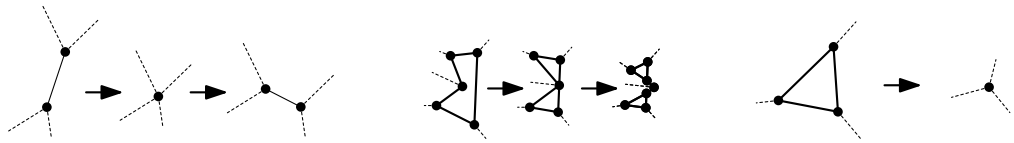

Figure 2: Edge event (a), split event (b), and face event (c) during the wavefront propagation.

the second class of events, *surface events*, in the wavefront propagation. Surface events comprise *edge events*, *(facet) split events* and *face event*.

An *edge event* happens at time t when an edge of the wavefront collapses to zero length but none of its incident facets vanishes. The two vertices incident at the edge are merged, giving rise to a high-degree vertex. For the wavefront after the event, at time $t+\delta$, this high-degree vertex has to be resolved and generally split again similar to the process at the initial wavefront construction. See Figure 2(a).

A *(facet) split event* happens when a vertex v of the wavefront that is incident at facet f moves into the interior of another edge e of f without f collapsing. This case is similar to the split event known from 2D straight skeletons. Combinatorially, the edge e is split at the locus of v and made incident to v, creating a higher-degree vertex which then needs to be resolved again for the post-event wavefront. See Figure 2(b).

In the third event type, the *face event*, a facet f may collapse to an empty area. This coincides with one or more edges of f collapsing or a vertex of f moving into the interior of another edge of f. However, in the case of a face event the area of f becomes empty during the event. At the event time t, the facet is replaced by a set of edges that cover its boundary without overlapping, thereby merging vertices which now occupy the same locus (if such vertices exist). Again, for the post-event wavefront higher-degree vertices may need to be resolved. See Figure 2(c).

If the input is not in general position then two vertices that share a facet but not an edge can also meet. This will result in a higher-degree vertex (usually of degree at least six) that needs to be split again. For now we have ruled out such cases by our general position assumption.

The propagation of the wavefront is finished once no more events occur. The three dimensional straight skeleton $\mathcal{S}(\mathcal{T})$ of $\mathcal{T}$ is then the structure whose edges are the traces of wavefront vertices and whose facets are the traces of wavefront edges. To unambiguously refer to features of the 3D straight skeleton, Aurenhammer et al. [3] call the edges of $\mathcal{S}(\mathcal{T})$ *spokes* and its facets *sheets*. The volumes bounded by sheets are called *cells*.

Interior vertices correspond to events that have been observed in the propagation process. Any wavefront vertex or edge remaining in the wavefront at the end of the process induces an unbounded straight skeleton spoke or sheet which continues to infinity.

2.2 Simulating the Wavefront Propagation for Computing the Straight Skeleton

We compute the straight skeleton $\mathcal{S}(\mathcal{T})$ of $\mathcal{T}$ by simulating its wavefront propagation. This requires determining at every stage in the process what the next event will be. Efficient solutions for this part are at the core of many straight skeleton construction algorithms in two dimensions [1, 13]. To cope with this problem we maintain a priority queue of potential events: As initialization, we first create the initial wavefront for time $t := \delta$, where δ is infinitesimally small, splitting higher degree vertices of $\mathcal{T}$. (In practice, this means duplicating input vertices and setting up zero-length edges which have the combinatorial structure of the initial offset.) Then we store for every edge of the wavefront its collapse time, and we store for every vertex of the wavefront the instances of when it will move into any of the edges of its incident facets.

To advance time in our simulation of the wavefront propagation, we fetch the event from the priority queue with the earliest time. We process this event by modifying the wavefront combinatorics according to the event type, thereby merging and then splitting vertices as described in the previous section. We add new events to the priority queue for all edges and vertices that were affected or created by the event.

Then we proceed and fetch the next item from the priority queue. We need to verify that it still is a valid event. That is, we need to check whether the edge that is supposed to collapse or the vertex that is supposed to move into an edge are still elements of the wavefront. (Prior events may have already restructured the wavefront and invalidated

this event.) If it is a valid event then we process it as described. Otherwise we simply drop it. In either case, this process is repeated until the priority queue is empty.

2.3 Number of events

In general, an event happens at point p and time t when four (or more) wavefront facets become incident. (For simplicity reasons, our general position assumption guarantees that no more than four wavefront facets are involved in an event.) This provides a trivial upper bound of $\binom{n}{4}$ on the size of the priority queue, where n is the number of facets of the input surface. Based on our experience with different algorithms for computing straight skeletons in two dimensions, we conjecture that only a very small subset of those $\binom{n}{4}$ combinations will be relevant in practice.

2.4 Splitting higher-degree vertices

As noted previously, an input vertex of degree four or higher will, in general, not translate to one offset vertex. As a matter of fact, it is not even obvious that there always exists a meaningful offset surface around such a vertex of higher degree. Wang and Liu's work [21] circumvent this problem by restricting their offsetting algorithm to purely hexagonal meshes, where all vertices are of degree three (or lower).

Aurenhammer and Walzl [3] show that an offset surface of a higher-degree vertex v always exists even though it is not necessarily unique. One offset that always exists corresponds to a wavefront where v has been replaced by a tree. In [3], they suggest as a simple approach to enumerate all combinatorially different trees and check whether they correspond to valid offset surfaces of v. Note that some combinatorial offset structures may result in offset facets that intersect each other and, thus, are invalid, see Figure 3. The geometry of a tree's element is dictated by its combinatoric properties. Such a valid tree will replace the vertex v in the propagating wavefront.

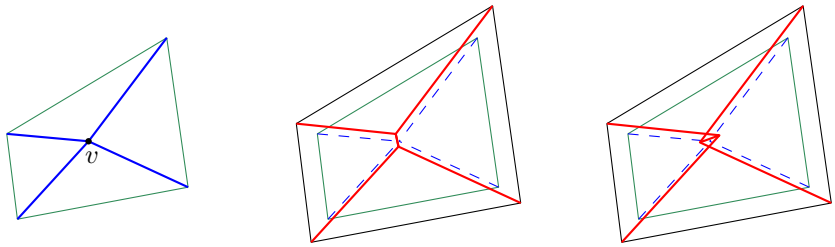

Figure 3: (a) A degree-four vertex v of the input roof. A geometrically valid offset is shown in (b) while the offset shown in (c) is invalid since two of its facets intersect.

Ross and Hambleton [20] seem to tackle the same problem in a similar manner: They use graph duality to describe the range of possible combinatorial outcomes at a vertex and then also enumerate tree structures to find a valid offset. Among all combinatorial structures which are also geometrically feasible they select the offset with lowest "cost", where their cost functions takes into account the length of the new edges and the fairness of the facets. Based on an old result by Goldbach and Euler (in 1751), their insight also seems to imply that a vertex of degree $k+2$ admits C_k many combinatorially different offset structures, where $C_k := \frac{1}{k+1}\binom{2k}{k}$ is the k-th Catalan number. (This observation is not stated in [20], though.)

By our general position assumption, all vertices of the input surface have at most constant degree k. Thus, finding this tree for a single vertex v is a constant-time operation as well. Furthermore, at most a constant number of elements need to be added to the wavefront per input vertex.

2.5 Vertex degrees during events

After having constructed the initial wavefront, all moving vertices will be of degree three in the generic case. We investigate the types of vertices that can appear in events. In an edge event, the edge that connects to degree-three

vertices collapses, giving rise to a degree-four vertex v, as shown in Figure 2(a). In the generic case, v will have to be split (at constant cost) into two new degree-three vertices connected by a new edge. In our general position assumption we stated that no more than four supporting planes of facets may be tangent to a common sphere. Thus, for our input we will always either split v into two vertices, or v will never again participate in an event.

In a split event, a degree-three vertex v comes to lie on previously non-incident wavefront edge e, which is split in two during the event, giving rise to a degree-five vertex (Figure 2(b)). In the generic case this vertex will be split into three new vertices, each of degree three. Again, by our general position assumption, this will be the case for our input sets.

For facet events we can distinguish two sub-types, see Figure 4. In the first type, a triangle facet will collapse as all its incident edges shrink to zero length. This will give immediate rise to a new degree-three vertex which can then propagate. In the second type a more complex polygon collapses as some of its edges collapse and possibly some vertices become incident at other edges of the polygon. The facet is replaced by one or more edges, and all resulting vertices will be of degree three and can propagate without any need to be split. Note that multiple facet collapses happening at the same time may cause an edge that has the same facet on both sides. Such an edge is not removed; instead it propagates like any other edge, similar to how ghost vertices propagate in Biedl et al. [6]. This ensures that all facets remain simply connected during the propagation.

Figure 4: Two types of facet collapsing events.

3 OBTAINING OFFSET SURFACES

If only a single mitered offset surface at orthogonal offset distance t is sought, then one approach to construct this offset is to simply run the wavefront propagation process until time t. Then, the wavefront at this time is the offset surface sought. See Figure 5 for a sample input and offset surface.

If multiple offset surfaces at different distances should be constructed or if the straight skeleton is already available, then we apply the following process to obtain an offset surface in time linear in the size of the straight skeleton: For a given spoke s, we denote by $s(t)$ the three dimensional point obtained by intersecting s with a plane at distance t which is parallel to the base of any one of its incident cells. Equivalently, $s(t)$ is the location at time t of the wavefront vertex that traced out s.

For every spoke s of the skeleton which exists at (orthogonal) offset distance t, i.e., for which $s(t)$ exists, and for every cell c incident at s where (s, c) has not been processed before, we construct an offset facet as follows: Let f_1 be one of the sheets of $s_1 := s$ that is on the boundary of c. We walk along the boundary of f_1, moving in the direction of positive z, until we reach another spoke s_2 of f_1 that exists at distance t. Now let f_2 be the other sheet of c incident at s_2 and repeat the walk in f_2 to find a spoke s_3. Eventually, we will return to our initial spoke s_1. Let s_ℓ be the last one before we returned. (Special handling will be required to process the case of infinite elements.) The polygon with vertices $s_1(t), s_2(t), \ldots, s_\ell(t)$ is now a valid offset facet and we add it to the offset surface we are constructing. We then mark $(s_1, c), (s_2, c), \ldots, (s_\ell, c)$ as processed and continue with our main loop. Once all spokes have been processed, the set of offset facets represents the complete offset surface. This algorithm can be implemented such that all offset facets together with their adjacency relations are obtained.

The correctness of this approach hinges on the property that all offset facets are simple polygons and contain no holes. This property stems from the fact that the wavefront propagation does not experience any piercing event since $\mathcal{T}$ is a terrain.

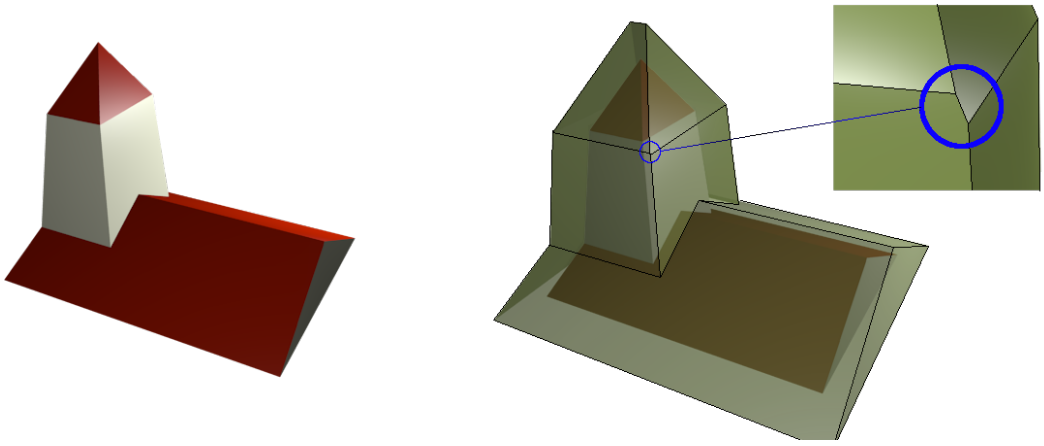

Figure 5: Steeple with a simple roof as a polyhedral terrain and one sample mitered offset of this terrain overlaid transparently. Note that terrain vertices of degree larger than three split into multiple vertices in the offset (see detail).

4 DISCUSSION

Wavefront propagation yields a simple algorithm to compute the straight skeleton of a terrain. The processing cost of each event is constant for generic input, and a (worst-case) bound for the number of events is $\binom{n}{4}$. Likely, this bound is far too loose but no tighter bound is known to theory. Maintaining the events in a priority queue results in a (pessimistic) runtime bound of $\mathcal{O}(n^4 \log n)$. A general $O(n^2)$ upper bound on the combinatorial complexity of $\mathcal{S}(\mathcal{T})$ is established by Aurenhammer and Walzl [3]. A construction by Held [10] yields a matching worst-case $\Omega(n^2)$ lower bound even for a terrain $\mathcal{T}$. His construction can be adapted to yield the same bound for the combinatorial complexity of one mitered offset.

For descriptive simplicity, our general-position assumption bounds the maximum degree of a vertex that may appear in the propagating wavefront by a small constant. However, using larger constants does not change the process significantly and only results in more complex event handling requirements.

Furthermore, we can relax the bound on the maximum degree of vertices of the input surface. Resolving higher-degree vertices where the degrees are not bound by a constant for the initial wavefront will require more than constant work, but at least for pointed vertices, where all incident faces are confined to one half space, offsetting can be reduced to computing weighted 2D straight skeletons [4], which are well studied [6] and for which implementations exist [11, 18]. Vertices that are saddle-points can still be handled by one of the methods described by Aurenhammer and Walzl [3].

It is known that computing the 2D straight skeleton of a planar straight-line graph or of a polygon with holes is P-complete [12], and as such is in a class of problems that are widely viewed as being difficult to parallelize effectively. In this paper we investigated its 3D pendant, albeit with input restricted to monotone polyhedral surfaces consisting of simply-connected facets. It is an open question whether the wavefront propagation for this type of input can be computed in parallel.

ACKNOWLEDGMENTS

This work was supported by Austrian Science Fund (FWF): P25816-N15 and ORD 53-VO.

ORCID

Martin Held http://orcid.org/0000-0003-0728-7545
Peter Palfrader http://orcid.org/0000-0002-5796-6362

REFERENCES

[1] Aichholzer, O.; Aurenhammer, F.: Straight Skeletons for General Polygonal Figures in the Plane. In Voronoi's Impact on Modern Sciences II, vol. 21, 7–21. Institute of Mathematics of the National Academy of Sciences of Ukraine, 1998. http://doi.org/10.1007/3-540-61332-3_144.

[2] Aichholzer, O.; Aurenhammer, F.; Alberts, D.; Gärtner, B.: A Novel Type of Skeleton for Polygons. Journal of Universal Computer Science, 1(12), 752–761, 1995. http://doi.org/10.1007/978-3-642-80350-5_65.

[3] Aurenhammer, F.; Walzl, G.: Straight Skeletons and Mitered Offsets of Nonconvex Polytopes. Discrete & Computational Geometry, 56(3), 743–801, 2016. http://doi.org/10.1007/s00454-016-9811-5.

[4] Barequet, G.; Eppstein, D.; Goodrich, M.T.; Vaxman, A.: Straight Skeletons of Three-Dimensional Polyhedra. In 16th Annual European Symposium on Algorithms (ESA), vol. 5193 of Lecture Notes in Computer Science, 148–160. Springer-Verlag, 2008. http://doi.org/10.1007/978-3-540-87744-8_13.

[5] Biedl, T.; Held, M.; Huber, S.; Kaaser, D.; Palfrader, P.: A Simple Algorithm for Computing Positively Weighted Straight Skeletons of Monotone Polygons. Information Processing Letters, 115(2), 243–247, 2015. http://doi.org/10.1016/j.ipl.2014.09.021.

[6] Biedl, T.; Held, M.; Huber, S.; Kaaser, D.; Palfrader, P.: Weighted Straight Skeletons in the Plane. Computational Geometry: Theory and Applications, 48(2), 120–133, 2015. http://doi.org/10.1016/j.comgeo.2014.08.006.

[7] Bolaños, G.S.; Bedi, S.; Mann, S.: A Topological-free Method for Three-Axis Tool Path Planning for Generalized Radiused End Milled Cutting of a Triangular Mesh Surface. International Journal of Advanced Manufacturing Technology, 70(9–12), 1813–1825, 2014. http://doi.org/10.1007/s00170-013-5450-7.

[8] Eppstein, D.; Erickson, J.: Raising Roofs, Crashing Cycles, and Playing Pool: Applications of a Data Structure for Finding Pairwise Interactions. Discrete & Computational Geometry, 22(4), 569–592, 1999. http://doi.org/10.1145/276884.276891.

[9] Held, M.: On the Computational Geometry of Pocket Machining, vol. 500 of Lecture Notes in Computer Science. Springer-Verlag, 1991. ISBN 978-3-540-54103-5. http://doi.org/10.1007/3-540-54103-9.

[10] Held, M.: On Computing Voronoi Diagrams of Convex Polyhedra by Means of Wavefront Propagation. In 6th Canadian Conference on Computational Geometry (CCCG), 128–133, 1994.

[11] Held, M.; Palfrader, P.: Straight Skeletons with Additive and Multiplicative Weights and Their Application to the Algorithmic Generation of Roofs and Terrains. Computer-Aided Design, 92, 33–41, 2017. http://doi.org/10.1016/j.cad.2017.07.003.

[12] Huber, S.; Held, M.: Approximating a Motorcycle Graph by a Straight Skeleton. In 23rd Canadian Conference on Computational Geometry (CCCG), 489–494, 2011.

[13] Huber, S.; Held, M.: A Fast Straight-Skeleton Algorithm Based On Generalized Motorcycle Graphs. International Journal of Computational Geometry & Applications, 22(5), 471–498, 2012. http://doi.org/10.1142/S0218195912500124.

[14] Jung, W.; Shin, H.; Choi, B.K.: Self-Intersection Removal in Triangular Mesh Offsetting. Computer-Aided Design and Applications, 1(1–4), 477–484, 2004. http://doi.org/10.1080/16864360.2004.10738290.

[15] Kim, S.J.; Lee, D.Y.; Yang, M.Y.: Offset Triangular Mesh Using the Multiple Normal Vectors of a Vertex. Computer-Aided Design and Applications, 1(1–4), 285–291, 2004. http://doi.org/10.1080/16864360.2004.10738269.

[16] Kim, S.J.; Yang, M.Y.: Triangular Mesh Offset for Generalized Cutter. Computer-Aided Design, 37(10), 999–1014, 2005. http://doi.org/10.1016/j.cad.2004.10.002".

[17] Liu, Y.; Wang, W.: On Vertex Offsets of Polyhedral Surfaces. In 1st Symposium on Advances in Architectural Geometry (AAG), 61–64. Institut für Diskrete Mathematik und Geometrie der TU Wien, 2008. ISBN 9-783902-233035.

[18] Palfrader, P.; Held, M.: Computing Mitered Offset Curves Based on Straight Skeletons. Computer-Aided Design and Applications, 12(4), 414–424, 2015. http://doi.org/10.1080/16864360.2014.997637.

[19] Pottmann, H.; Asperl, A.; Hofer, M.; Kilian, A.: Architectural Geometry. Bentley Institute Press, 2007. ISBN 978-1934493045.

[20] Ross, E.; Hambleton, D.: Exact Face-Offsetting for Polygonal Meshes. In Association for Computer-Aided Design in Architecture International Conference (ACADIA), 202–209, 2015.

[21] Wang, W.; Liu, Y.: A Note on Planar Hexagonal Meshes. In Nonlinear Computational Geometry, 221–233, 2009. http://doi.org/10.1007/978-1-4419-0999-2_9.

[22] Wen, H.; Gao, J.; Chen, X.: Polygonal Model Based Cutter Location Data Generation with Offset Error Compensation. Rapid Prototyping Journal, 22(3), 559–568, 2016. http://doi.org/10.1108/RPJ-01-2015-0001.

[23] Yi, I.L.; Lee, Y.S.; Shin, H.: Mitered Offset of a Mesh Using QEM and Vertex Split. In ACM Symposium on Solid and Physical Modeling (SPM), 315–320, 2008. http://doi.org/10.1145/1364901.1364945.