Unfolding Orthogonal Polyhedra with Quadratic Refinement: The Delta-Unfolding Algorithm

Unfolding Orthogonal Polyhedra with Quadratic Refinement: The Delta-Unfolding Algorithm

**Mirela Damian · Erik D. Demaine ·
Robin Flatland**

Abstract We show that every orthogonal polyhedron homeomorphic to a sphere can be unfolded without overlap while using only polynomially many (orthogonal) cuts. By contrast, the best previous such result used exponentially many cuts. More precisely, given an orthogonal polyhedron with n vertices, the algorithm cuts the polyhedron only where it is met by the grid of coordinate planes passing through the vertices, together with $\Theta(n^2)$ additional coordinate planes between every two such grid planes.

1 Introduction

One of the major unsolved problems in geometric folding is whether every polyhedron (homeomorphic to a sphere) has an "unfolding" [3, 10]. In general, an *unfolding* consists of cutting along the polyhedron's surface such that what remains flattens into the plane without overlap. Convex polyhedra have been known to unfold since at least the 1980s [10, Sec. 24.1.1].

A recent breakthrough for nonconvex polyhedra is the unfolding of any "orthogonal" polyhedron (homeomorphic to a sphere) [6]. A polyhedron is

M. Damian
Dept. of Computing Sciences, Villanova University, 800 Lancaster Avenue, Villanova, PA 19085, USA. E-mail: mirela.damian@villanova.edu.

E. D. Demaine
Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology, 32 Vassar St., Cambridge, MA 02139, USA. E-mail: edemaine@mit.edu

R. Flatland
Dept. of Computer Science, Siena College, 515 Loudon Road, Loudonville, NY 12211, USA. E-mail: flatland@siena.edu

orthogonal if all of its edges are parallel to a coordinate axis, and thus all edges and faces meet at right angles. While very general, a disadvantage of this unfolding algorithm is that the cutting is inefficient, making exponentially many cuts in the worst case, resulting in an unfolding that is long and thin ("epsilon thin").

In this paper, we show how to unfold any orthogonal polyhedron using only a polynomial number of cuts.

Grid refinement. To more precisely quantify the cuts required by an unfolding, several models of allowed cuts have been proposed. See [9, 10, 11] for surveys.

For convex polyhedra, the major unsolved goal is to just cut along the edges (which implies a linear number of cuts) [10, ch. 22]. For nonconvex polyhedra, however, this goal is unattainable, even when the polyhedron is "topologically convex" [3] or is orthogonal [4]. A simple example of the latter is a small box on top of a larger box. More generally, deciding whether an orthogonal polyhedron has an edge unfolding is strongly NP-complete [1].

For orthogonal polyhedra, it seems most natural to consider orthogonal cuts. The smallest extension from edge unfolding seems to be *grid unfolding* (a concept implicit in [4]), where we slice the polyhedron with all axis-aligned planes that pass through at least one polyhedron vertex, and allow cutting along all slice lines. Even with these additional edges, few nontrivial subclasses of orthogonal polyhedra are known to have grid unfoldings: "orthotubes" [4], "orthostacks" composed of orthogonally convex slabs [8], and "well-separated orthotrees" [5]. On the negative side, there are four orthogonal polyhedra with no *common* grid unfolding [2].

The next extension beyond grid unfolding is *grid refinement k*, which additionally slices with k planes in between every grid plane (as above), and allows cuts along any edges of the refined grid. With constant grid refinement, a few more classes of orthogonal polyhedra have been successfully unfolded: orthostacks [4], and Manhattan towers [7].

The breakthrough was the discovery that arbitrary orthogonal polyhedra (homeomorphic to a sphere) unfold with finite grid refinement [6]. Unfortunately, the amount of grid refinement is exponential in the worst case (though polynomial for "well-balanced" polyhedra). For this reason, the unfolding algorithm was called *epsilon-unfolding*.

Our results. We show how to modify the epsilon-unfolding algorithm of [6] to reduce the refinement from worst-case exponential ($2^{\Theta(n)}$) to worst-case quadratic ($\Theta(n^2)$), while still unfolding any orthogonal polyhedron (with n vertices) homeomorphic to a sphere. We call our algorithm the *delta-unfolding* algorithm, to suggest that the resulting surface strips are still narrow but wider than those produced by epsilon-unfolding.

Our central new technique in delta-unfolding is the concept of "heavy" and "light" nodes from "heavy-path decomposition" [12]. Interestingly, heavy-path decomposition is a common technique for balancing trees in the field of data structures, but not so well known in computational geometry.

Even with this technique in hand, however, delta-unfolding requires a careful modification and engineering of the techniques used by epsilon-unfolding. Thus, Sections 2 and 3 start with reviewing the main techniques of epsilon-unfolding; then Section 4 modifies those techniques; and finally Section 5 puts these techniques together to obtain our main result.

2 Overview of Epsilon-Unfolding

We begin with a review the epsilon-unfolding algorithm [6], starting in this section with a high-level overview, and then in Section 3 detailing those aspects of the algorithm that we modify to achieve quadratic refinement.

Throughout this paper, P denotes a genus-zero orthogonal polyhedron whose edges are parallel to the coordinate axes and whose surface is a 2-manifold. We take the z-axis to define the *vertical* direction, the x-axis to determine *left* and *right*, and the y-axis to determine *front* and *back*. We consistently take the viewpoint from $y = -\infty$. The faces of P are distinguished by their outward normal: forward is $-y$; rearward is $+y$; left is $-x$; right is $+x$; bottom is $-z$; top is $+z$.[1]

The epsilon-unfolding algorithm partitions P into *slabs* by slicing it with y-perpendicular planes through each vertex. Let $Y_0, Y_1, Y_2, \ldots$ be the slicing planes sorted by y coordinate. A *slab* s is a connected component of P located between two consecutive planes Y_i and Y_{i+1}. Each slab is a simple orthogonal polygon extruded in the y-direction. The cycle of {left, right, top, bottom} faces surrounding s is called a *band*, and the band edges in Y_i (and similarly in Y_{i+1}) form a cycle called a *rim*. A *z-beam* is a narrow vertical strip on a forward or rearward face of P connecting the rims of two bands. The order in which the bands unfold is determined by an *unfolding tree* T_U whose nodes are bands, and whose arcs correspond to z-beams, each of which connects a parent band to a child band in T_U. The unfolding tree T_U will be further described in Section 3.1 below.

The unfolding of a band b is determined by a thin surface spiral, denoted ξ, that starts on one of b's rims, cycles around b while displacing toward the other rim, where it turns around and returns to the point it started. As the spiral passes by a z-beam connecting b to one of its children, it enters through the z-beam to the child's rim, and then recursively visits the subtree rooted at the child. Once the complete spiral is determined, it can be thickened in the $\pm y$ direction so that it entirely covers all band faces. The thickened spiral is such that it can be laid flat in the plane to form a monotonic staircase strip. The forward and rearward faces of P can then be laid flat without overlap by attaching them in strips above and below the staircase.

---

[1] The $\pm y$ faces are given the awkward names "forward" and "rearward" to avoid confusion with other uses of "front" and "back" introduced later.

3 Epsilon-Unfolding Extrusions

Almost all algorithmic issues in epsilon-unfolding are present in unfolding poly-hedra that are z-extrusions of simple orthogonal polygons in the xy plane. Therefore, we follow [6] in describing the algorithm for this simple shape class, before extending the ideas to all orthogonal polyhedra. All modifications needed for delta-unfolding are also present in unfolding orthogonal extrusions, and so we describe them in terms of this simple shape class. We therefore review in detail the epsilon-unfolding algorithm for orthogonal extrusions.

3.1 Unfolding Tree

Let P be a polyhedron that is the vertical extrusion of a simple orthogo-nal polygon, such as that illustrated in Figure 1a. The algorithm begins by slicing P into slabs, which in this special case are all blocks (cuboids), using y-perpendicular planes through each vertex. The dual graph is a tree, T_U, hav-ing a node for each band and an edge between each pair of adjacent bands. In this special case, all z-beams are degenerate, i.e., of zero z-height. The root is selected arbitrarily from among all bands with a rim of minimum y coordinate. For example, the polyhedron in Figure 1a is sliced into nine blocks, with b_1 as the root and its unfolding tree as shown in Figure 1b.
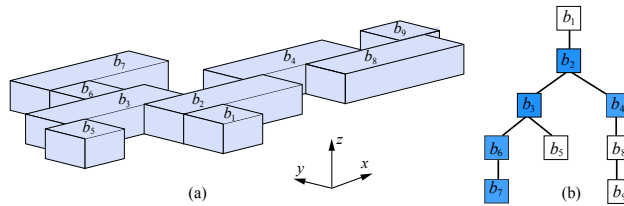
Fig. 1 (a) Extrusion of an orthogonal polygon, partitioned by y perpendicular planes. (b) Unfolding tree. Back children are represented by shaded nodes.

The rim of the root band with the smaller y coordinate is its *front rim*, and the other rim is its *back rim*. For any other band, the rim adjacent to its parent in T_U is its front rim, and its other rim is its back rim. Children attached along the front rim of their parent are *front children*; children attached along the back rim of their parent are *back children*. Note that "front" and "back" modifiers for rims and children derive from the structure of T_U, and are not related to the "forward" and "rearward" $\pm y$ directions. For example, b_9 is a front-child of b_8, although it is attached to the rearward face of b_8, and the front rim of b_5 lies on the rearward face of b_5.

3.2 Recursive Unfolding

The key to the epsilon-unfolding method is the existence of a thin, non-crossing *spiral* ξ that cycles around each band at least once, and unfolds to a staircase when flattened into the plane. A *staircase* is an orthogonal path in the plane whose turns alternate between 90° left and 90° right, and so is a monotone path. The path that ξ follows is determined recursively. We review this *spiral* ξ, starting with the base case.

3.2.1 Single Band Base Case

Figure 2a shows the path followed by ξ for a single band corresponding to a leaf of T_U. It starts at an *entering point s* on the top edge of the front rim and spirals in a clockwise direction around the top, right, bottom, and left band faces toward the back rim. We call this spiral piece up to the point it reaches the back rim the *entering spiral*. When it reaches the back rim, ξ crosses the rearward face upward toward the top face. From there, it retraces the entering spiral in the opposite (counterclockwise) direction toward an *exiting point t* lying next to s on the front rim. When ξ is cut out, unfolded, and laid horizontally in the plane, it forms a monotonic staircase strip, as shown in Figure 3a, because the turns alternate between left and right, 90° each. Observe that the x, z-parallel segments of ξ, corresponding to the cycling clockwise and counterclockwise around the band, form the stair "treads"; the y-parallel segments of ξ and the z-parallel strip from the rearward face form the stair "risers."
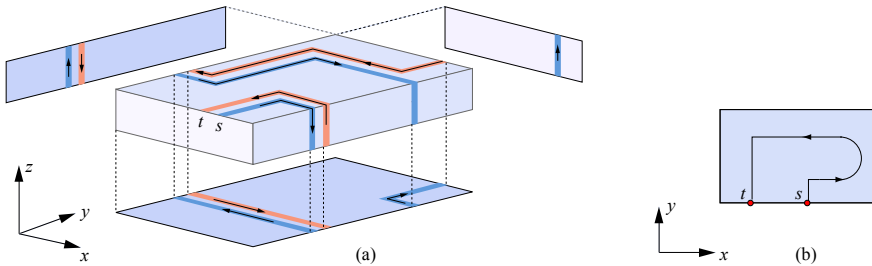

Fig. 2 (a) R_{ts} block spiral, with mirror views of faces that cannot be seen directly. (b) Abstract 2D representation.

Three-dimensional illustrations of ξ like that in Figure 2a are impractical for more complex orthogonal shapes. To easily illustrate more complex unfoldings, we use the 2D representation depicted in Figure 2b. Note that the 2D representation captures the direction of the entering spiral and the relative position of s and t. The arc connecting the entrance to the exit symbolizes the reversal of the unfolding direction using a rearward face strip.
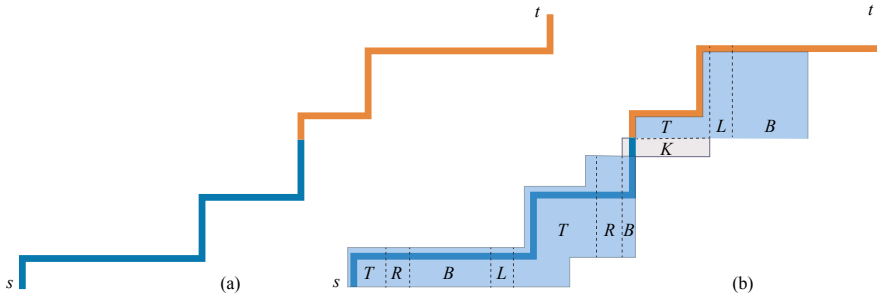
Fig. 3 (a) Spiral from Figure 2a unfolded. (b) Thickened spiral with the back face hung underneath. Dotted lines delineate surface pieces from different block faces and are labeled T (top), R (right), B (bottom), L (left), and K (back).

Eight variations of the base case spiral are illustrated in Figure 4. They differ in the manner in which ξ enters and exits the band b to be unfolded. The four variations labeled L_{ts}, L_{st}, R_{ts}, R_{st} in the top row are used when the y-coordinate of b's front rim is smaller than the y-coordinate of its back rim. R_{st} is similar to R_{ts}, but with s and t, and the clockwise/counterclockwise cycling direction reversed; L_{ts} and L_{st} are (respectively) mirrors of R_{st} and R_{ts} in an x-perpendicular plane. Note that the R and L labels indicate the spiral's cycling direction when it enters the band: R is clockwise, L is counterclockwise. The spiral exits the band cycling in the opposite direction. The four variations in the bottom row are labeled $L_{ts}^+, L_{st}^+, R_{ts}^+, R_{st}^+$, and they are used when the y-coordinate of b's front rim is greater than the y-coordinate of its back rim. They are exact reflections of L_{ts}, L_{st}, R_{ts}, and R_{st}, respectively, in a y-perpendicular plane. The mirror symmetries imply that the 3D spiral corresponding to each 2D abstract representation can be easily derived from R_{ts} configuration, illustrated in Figure 2.
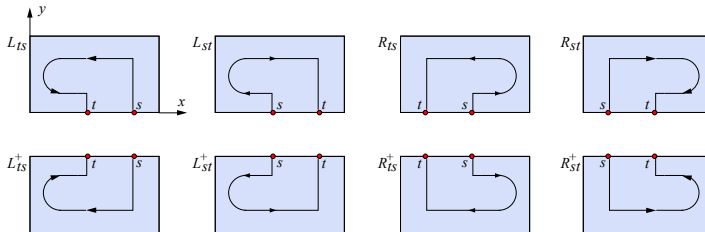

Fig. 4 Abstract 2D representations of the eight path types visiting one slab.

3.2.2 Recursive Path

For a node b in T_U that is not a leaf, we describe the path that the spiral ξ recursively follows when visiting b. We assume that b has one of the eight

configuration labels shown in Figure 4. As in the base cases, the label identifies the relative order of points s and t on b's front rim, the spiral's direction when entering b, and b's rim of lower y coordinate. Without loss of generality, we assume that b's label is R_{ts}; the other seven labels are equivalent by symmetry. The inductive assumption is that, for any subtree shorter than the subtree of T_U rooted at b, and for any configuration label assigned to the root band of the shorter subtree, there is a (non-crossing) path ξ consistent with that label that cycles around each band in the smaller subtree at least once, and unfolds in the plane as a staircase strip.

After ξ enters b at point s, it visits each of b's front children, starting with the front child, call it b_1, first encountered as it cycles clockwise along the front rim of b. (See Figure 5). For reasons soon to be explained, child b_1 is assigned the label R_{st}^+ with two points s_1 and t_1 identified on the top edge of its front rim, with t_1 right of s_1. The spiral ξ enters b_1 at point s_1 and recursively visits it (and the subtree it roots). By the inductive hypothesis, ξ exits b_1 at point t_1 cycling counterclockwise. The label R_{st}^+ is assigned to b_1 because ξ is cycling in the direction R (to the right, or clockwise) on b just before it enters b_1, and so it enters b_1 with that same direction; the $+$ superscript is necessary because the y-coordinate of the front rim of b_1 is higher than that of its back rim; and the $*_{st}$ ordering is necessary to prevent ξ from being trapped beneath the portion of ξ between s and s_1 upon returning to b, thus cutting itself off from reaching b's other children (because it cannot cross itself).
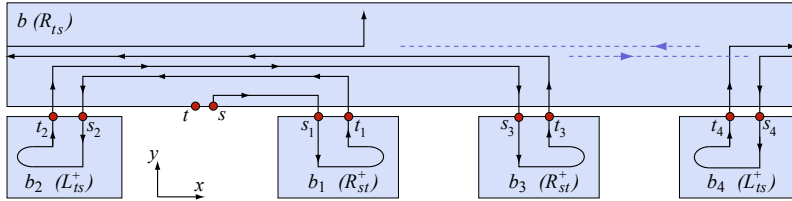

Fig. 5 Nested inside-out alternating path visits the front children. Dotted lines show ξ where it cycles underneath on the bottom face.

After recursively visiting b_1, ξ cycles counterclockwise on b to the first unvisited child it passes when on b's top face. This child, call it b_2, is assigned the label L_{ts}^+ with identified points t_2 and s_2 on its front rim, consistent with its label, and it is recursively visited. In this nested manner, ξ visits the children clockwise and counterclockwise from s from the inside out, alternately assigning the labels R_{st}^+ and L_{ts}^+. Figure 5 illustrates the path ξ takes when b has four front children. (To keep the example simple, only one level of recursion is illustrated, with all children leaves of T_U.)

After visiting the front children, ξ makes a complete cycle around b and then begins visiting the back children. Assume for concreteness that after visiting the front children, ξ is cycling clockwise on b, as shown in Figure 6. It then travels clockwise to the back child farthest to the right along the top
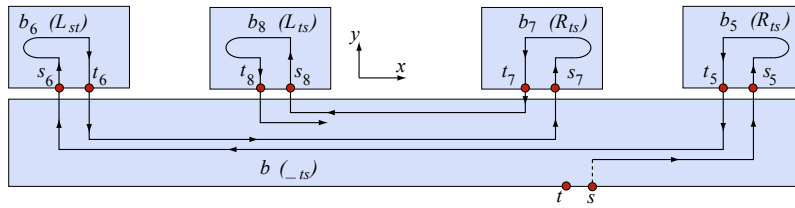
Fig. 6 Nested outside-in alternating path visits the back children.

face. (See child b_5 in Figure 6). When it returns from recursively visiting this back child, it will be cycling counterclockwise (by the inductive hypothesis). It is thus important that the child's exit point be to the left of its entering point so that ξ is not blocked from visiting other back children. (See points s_5 and t_5 in Figure 6.) Thus this first back child is assigned the configuration R_{ts} and is recursively visited. The spiral then moves to the unvisited child farthest to the left (see child b_6) and visits it in a similar way, assigning it the label L_{st}. Thus the nesting of ξ's alternating path is outside-in for back children, with the labels R_{ts} and L_{st} being alternately assigned.

The last back child visited, b_k, however, is an exception when it comes to its label assignment, for the following reason. When the spiral exits b_k (see b_8 in Figure 6), it will retrace its path (in reverse direction) back to the front rim of b and then exit at point t. For band b, define its *entering spiral*, $\xi_e(b)$, to be the portion of ξ that begins at s and ends at the exiting point t_k of b_k (t_8 in Figure 6). Its *exiting spiral*, $\xi_x(b)$, is the portion of ξ that begins at t_k and ends at t on the front rim of b. The exiting spiral $\xi_x(b)$ simply parallels alongside the entering spiral $\xi_e(b)$, retracing the portion of $\xi_e(b)$ from s to s_k but in the opposite direction. Since b has a $*_{ts}$ label, the exiting spiral must leave b with the entering spiral on its left, from the point of view of one walking on b along the path taken by $\xi_x(b)$. Thus b_k must also be assigned the label $*_{ts}$ (consistent with b's label), so that from the beginning of the retrace and throughout, $\xi_x(b)$ has the entering spiral to its left. We call this a *left retrace*; when $\xi_x(b)$ keeps the entering spiral on its right during a retrace, we call it a *right retrace*. We note that if b has no back children, then the spiral reverses direction using a strip from b's rearward face, as in the base cases.

3.3 Completing the Unfolding

We have focused on ξ's recursive path because that is where the modifications for delta-unfolding occur. But for completeness, we briefly summarize the remainder of the epsilon-unfolding algorithm for extrusions, and refer the reader to [6] for additional details. To complete the unfolding of P, ξ is thicken in the $+y$ and $-y$ direction (as viewed in the 3D coordinate system of Figure 2a) so that it completely covers each band. This results in a thicker unfolded staircase strip. Then the forward and rearward faces of P are partitioned by imagining the band's top rim edges illuminating downward light rays in these faces.

The illuminated pieces are then "hung" above and below the thickened staircase, along the corresponding illuminating rim segments which lie along the horizontal edges of the staircase. See Figure 3b.

3.4 Level of Refinement

In [6] it was shown that the unfolding technique discussed so far can make an exponential number of cuts on the family of polyhedra depicted in Figure 7. Each polyhedron consists of $n = 2k + 1$, $k \geq 1$, blocks arranged as shown for $k = 1$ in Figure 7a, and for $k = 2$ in Figure 7b. For analysis purposes, we formally define a *visit* to a band to begin when the spiral crosses its front rim to enter the band (either the first time, or in a retrace) and end when it crosses the front rim to exit the band. In Figure 7a, b_3's visit begins when ξ enters it at point s_3 cycling counterclockwise. The spiral visits back child b_1 and then b_2. The visit of b_2 triggers a retrace which involves a second visit of b_1, and then back through b_3 to exiting point t_3, which completes b_3's visit. We
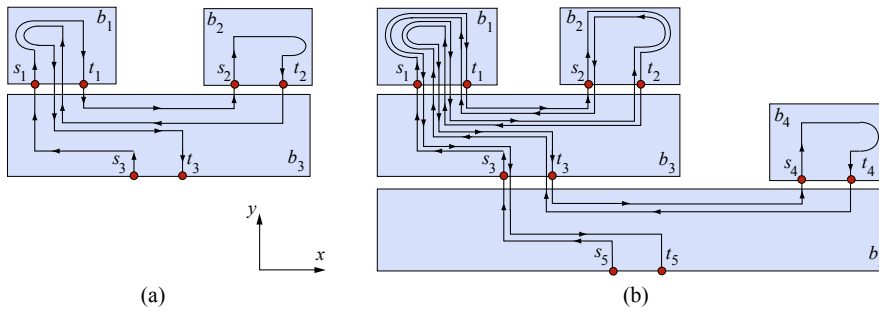
Fig. 7 Family of polyhedra requiring exponential refinement. Block b_1 is visited two times in (a), four times in (b), and in general $2^{\lfloor n/2 \rfloor}$ times for an n-block object.

can write this visit order using the string $Q_3 = (s_3 \ (s_1 \ t_1) \ (s_2 \ t_2) \ (s_1 \ t_1) \ t_3)$, where an open parenthesis followed by a starting point marks the start of a visit and an exiting point followed by a closing parenthesis marks the end. The subscript on Q is the number of blocks in the polyhedron. Observe that block b_1 is visited twice. For the five block polyhedron in Figure 7b, ξ starts at point s_5 on b_5, recursively visits block b_3 in the manner just described, then visits b_4 which triggers a retrace through b_3. After revisiting b_3, ξ returns to b_5 and exits at point t_5. The corresponding visit string is $Q_5 = (s_5 \ Q_3 \ (s_4 \ t_4) \ Q_3 \ t_5)$. The number of visits to b_1 doubles to 4. In general, an n block polyhedron in this family gives rise to $2^{\lfloor n/2 \rfloor}$ visits to b_1, resulting in an exponential number of cuts on b_1.

4 Delta-Unfolding Extrusions

To achieve quadratic refinement, we modify the order in which children are visited based on the heavy/light classification of nodes used in *heavy-path decomposition* [12]. In heavy-path decomposition, each tree node v is assigned a weight $n(v)$, which is the number of descendants in its subtree, including itself. An edge from parent p to child c is *heavy* if $n(c) > \frac{1}{2}n(p)$, and *light* otherwise. We say a child c is *heavy* (*light*) if the edge between c and its parent is heavy (light). Observe that a node can have at most one heavy child.

If a node b in T_U has a heavy child, then we modify the path of the entering spiral $\xi_e(b)$ so that it visits the heavy child last, to prevent the need for revisiting the heavy child; we will show that this strategy quadratically bounds the number of visits ξ makes to each child. For example, consider the polyhedron in Figure 7b, and observe that b_3 is a heavy child. With epsilon-unfolding, $\xi_e(b)$ visited child b_3 before b_4. The visit to b_4 triggered a complete retrace of the subtree rooted at b_3, thus leading to the visit string $Q_5 = (s_5\ Q_3\ (s_4\ t_4)\ Q_3\ t_5)$, and a total of four visits to b_1. But if we reverse the visit order so that $\xi_e(b)$ visits b_3 after b_4, then the visit string becomes $Q_5' = (s_5\ (s_4\ t_4)\ Q_3\ (s_4\ t_4)\ t_5)$, and no block is visited more than twice.

Since any front or back child could be heavy, we focus first on the challenge of finding a route for the entering spiral so that it visits any specified child last. If we can achieve this, then we can organize the visits to minimize retracing. We then formally present the algorithm and analyze the resulting level of refinement.

4.1 Front Child Visited Last

We start with the case when we desire to visit a front child, call it b_ℓ, last. The idea is to visit all the front children excluding b_ℓ, and all the back children in exactly the manner described in Section 3.2.2, as if b_ℓ were not present. Figure 8 shows the entering spiral $\xi_e(b)$ visiting all but b_ℓ. (Note that the complete cycle that $\xi_e(b)$ makes between visiting the front and back children is not fully depicted in the 2D representation.) After visiting the last back child (b_9 in the figure), ξ retraces its path in reverse. It is during this retrace step that child b_ℓ is visited. We explain the modifications necessary to accomplish this for a parent block b with a label of type R_* or L_*; labels of type R_*^+ and L_*^+ are handled symmetrically.

Observe first that since $\xi_e(b)$ alternately visits all the front children except for b_ℓ and then makes a complete cycle around b, some contiguous section of it, call it f, runs alongside the top edge of b_ℓ's front rim. Specifically, f is the section of $\xi_e(b)$ hit by y-parallel rays shot from b_ℓ's top front rim edge toward the back rim of b. See Figure 8 where f is marked. Note that since $\xi_e(b)$ is cycling toward the back rim of b, f represents the first time $\xi_e(b)$ passes by b_ℓ's top edge. All subsequent passes are behind f.
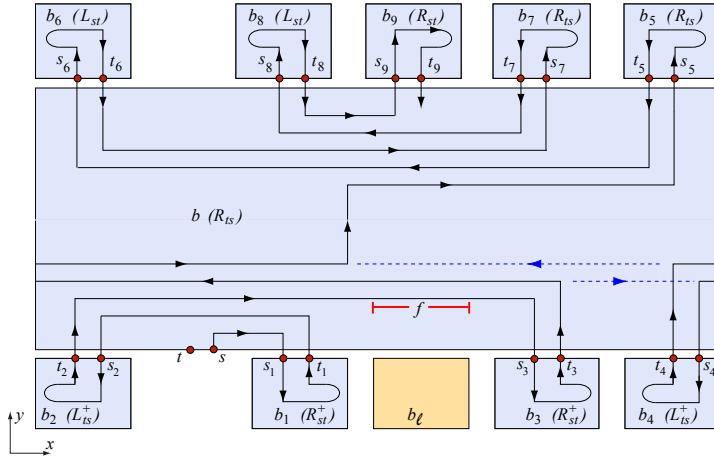
Fig. 8 Entering spiral visits front and back children, with the exception of front child b_ℓ, which gets visited last (see Figure 9).
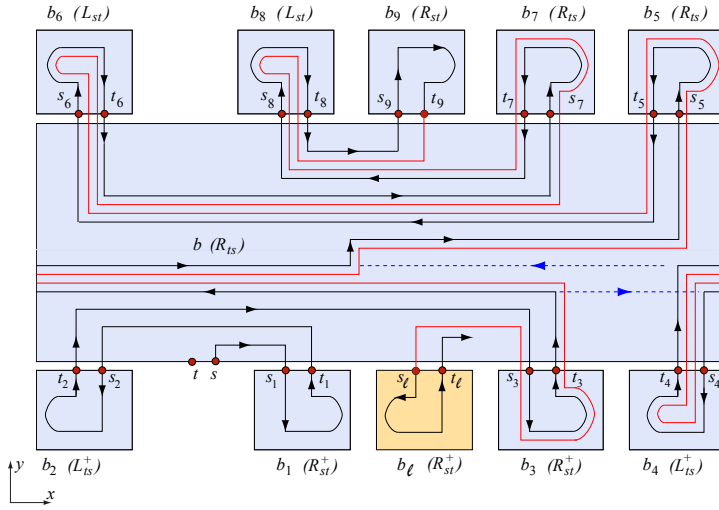
Fig. 9 Entering and return spirals. The return spiral passes by b_ℓ so that b_ℓ can be visited.

During the retrace step, ξ needs to run in front of f, so that it has unobstructed access to b_ℓ. If the entering spiral is cycling clockwise in section f, then the retracing spiral (which runs alongside f in the opposite direction) needs to right-retrace, because that will keep the entering spiral to its right and position it in front of f. (Recall that clockwise is to the right and counterclockwise is to the left.) To trigger a right-retrace, we assign the last visited back child the label $*_{st}$. If the entering spiral is cycling counterclockwise in section f, then the retracing spiral needs to left-retrace, thus keeping the entering spiral on its left. To trigger a left-retrace, we assign the last back child

the label $*_{ts}$. So instead of matching the $*_{st}$ or $*_{ts}$ label of the last visited back child to that of b (as in epsilon-unfolding), we instead assign it so that the retracing spiral passes alongside b_ℓ. When the retracing spiral reaches section f, it suspends the retrace, enters b_ℓ at point s_ℓ, and visits it. We call the portion of ξ from the exiting point of the last visited back child to s_ℓ the *return spiral* and label it $\xi_r(b)$. See Figure 9 which shows $\xi_r(b)$ in gray extending from t_9 to s_ℓ.

Upon exiting b_ℓ at point t_ℓ, the spiral retraces its path in the reverse direction, bringing it to the exit point t on b. Specifically, it follows the entire path from s to s_ℓ in reverse. This second retrace is b's exiting spiral, $\xi_x(b)$. (In Figure 9, $\xi_x(b)$ is not illustrated, but it begins at t_ℓ and follows the gray and then the black path to t, keeping them to its left.) The label $*_{st}$ or $*_{ts}$ assigned to b_ℓ must be consistent with b's label in the following way. If b has the label $*_{ts}$, a left retrace starting from t_ℓ is needed so that the spiral exits at t on the correct side of s consistent with b's $*_{ts}$ label. Thus, b_ℓ is assigned the label $*_{st}$, the opposite of b's label. If, however, b has the label $*_{st}$, a right retrace is needed, and so b_ℓ is assigned the label $*_{ts}$.

Because the label assigned to the last back child visited depends on the direction of $\xi_e(b)$ in the f-section of the path, we show here that determining that direction is straightforward. We discuss the case in which $\xi_e(b)$ enters b cycling clockwise; the case when it is cycling counterclockwise is symmetric. We also assume that there are at least two front children (not including b_ℓ) and they are labeled $b_1, b_2, b_3, \ldots$, in the order in which they are visited along the alternating path (as in Figure 8). Observe that, if b_ℓ is located between s and b_1 (as viewed from above), then $\xi_e(b)$ first passes by b_ℓ's top edge cycling clockwise, and the same is true if it is located between b_i and b_{i+2}, for i odd ($i \in \{1, 3, 5, \ldots\}$). Thus in these cases, f is traversed clockwise. Figure 8 illustrates the case when b_ℓ is between b_1 and b_3. If b_ℓ is located between s and b_2 or between b_i and b_{i+2}, for $i \in \{2, 4, 6, \ldots\}$, then $\xi_e(b)$ first passes by the top edge of b_ℓ cycling counterclockwise. Thus in these cases f is traversed counterclockwise. If the top edge of b_ℓ is to the right of the last odd numbered child or to the left of the last even numbered child, then $\xi_e(b)$ first passes over b_ℓ during its complete cycle around b. During this cycle, $\xi_e(b)$ is heading clockwise if the last visited child was even and counterclockwise if the last visited child was odd. Cases when there are fewer than two front children are easily handled: if b_ℓ is the only front child, or if it is located between s and b_1, then f is traversed clockwise; otherwise, f is traversed counterclockwise.

4.2 Back Child Visited Last

In this section we discuss the situation in which we desire to visit a particular back child b_ℓ last. In this case, $\xi_e(b)$ visits the front children as described in Section 3.2.2. It then visits the back children as described in Section 3.2.2 but with an altered visiting order. We consider the case when b has a L_* or R_* type

configuration label and the entering spiral $\xi_e(b)$ is cycling counterclockwise after visiting the front children; the other cases are symmetric.

Let $m \geq 0$ be the number of back children of b not including b_ℓ, and let $b_1, b_2, \ldots b_j$ be the front children, for $j \geq 0$. Consider the back children of b in the cyclic clockwise order in which their top edges occur around b's back rim. When m is odd, we label the m back children $(b_{j+1}, b_{j+3}, \ldots, b_{m-2}, b_m, b_\ell, b_{m-1}, \ldots, b_{j+4}, b_{j+2})$, according to their positions relative to b_ℓ in this cyclic ordering. When m is even, the labeling is $(b_{j+1}, b_{j+3}, \ldots, b_{m-1}, b_\ell, b_m, \ldots, b_{j+4}, b_{j+2})$, as depicted in Figure 10. The spiral $\xi_e(b)$ visits the back children from the outside-in, following the visit order $b_{j+1}, b_{j+2}, \ldots, b_{m-1}, b_m, b_\ell$. It is always possible to visit b_{j+1} first, with a full cycle of the spiral around b (if necessary) to get the spiral to the top edge of b_{j+1}. This is illustrated in Figure 10 for five back children (and no front children).
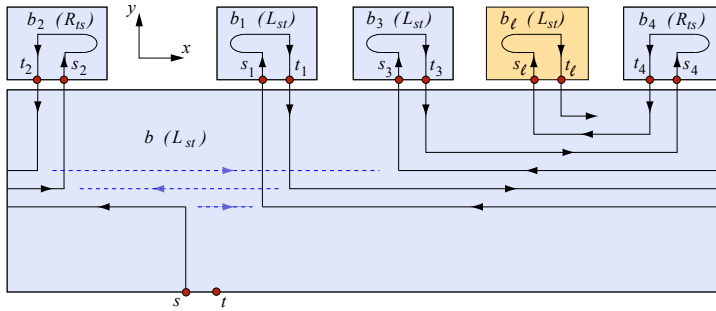
Fig. 10 Labels and path followed by spiral when visiting back children, when the last child to be visited is back child b_ℓ; dashed lines depict spiral pieces on the bottom of the parent block.

The assignment of L_* and R_* labels to the back children of b is the same as described in Section 3.2.2. Specifically, the labels for the children alternate between L_{st} and R_{ts} with respect to the visiting order. The spiral $\xi_e(b)$ is cycling counterclockwise (to the left) when it reaches b_{j+1}, which matches b_{j+1}'s L_* label. The recursive unfolding of b_{j+1} reverses the direction of the spiral, so that it enters b_{j+2} cycling clockwise (to the right), thus matching b_{j+2}'s R_* label, and similarly for the other back children. The alternating $*_{st}$, $*_{ts}$ labels of the children ensures an outside-in nesting of ξ, which enables it to reach each back child. As in Section 3.2.2, the one exception to the alternating labels is the last visited child b_ℓ, whose $*_{st}$ or $*_{ts}$ label needs to match that of its parent b. After visiting b_ℓ, the exiting spiral follows the entering spiral in reverse to t as in Section 3.2.2, thus completing the visit of b.

4.3 The Delta-Unfolding Algorithm for Extrusions

What we call the *delta-unfolding* algorithm is a modified version of the epsilon-unfolding algorithm, which requires that at each node b in T_U with a heavy child, the spiral ξ visits the heavy child last. Specifically, if the heavy child is a front child, then ξ follows the path described in Section 4.1; if the heavy child is a back child, then ξ follows the path described in Section 4.2. If b has no heavy child, then its children are visited in the epsilon-unfolding order (Section 3.2.2). All remaining steps of the delta-unfolding algorithm for extrusions—the thickening of ξ, the unfolding of ξ as a staircase in the plane, and the partitioning and hanging of the frontward and rearward faces from the flattened staircase—are the same as for epsilon-unfolding.

4.4 Refinement Analysis

We now turn to analyzing the refinement for extrusions. The path taken by ξ on a band is composed of a series of axis-parallel segments. We determine an asymptotic upper bound on the number of such segments on any band face, because this is an asymptotic upper bound on the total number of cuts on a grid face in the unfolding. We compute this by bounding the number of segments on any top face, as the number of segments on all four faces of a band is asymptotically bounded by the number of segments on its top face.

   Define the *first visit* of ξ to a band b to begin when ξ *first* enters b at point s, includes the recursive visiting of b's children, and ends when it exits b at point t. Band b and the bands in its subtree may be revisited by ξ many times during subsequent retracings, but each of these retracings merely follows the path traced during the first visit to b. Let $R(n(b))$ be an asymptotic upper bound on the number of segments that ξ's first visit to b induces on a top face of any band in the unfolding subtree rooted at b. Then a bound on the number of segments on any top face in b's subtree induced by ξ (in its entirety) is $R(n(b))$ multiplied by the total number of times ξ visits b. We now establish three properties of ξ's first visit to b:

   (i) ξ induces at most $O(n(b))$ segments on b's top face;
  (ii) the light children of b are each visited at most four times; and
 (iii) if b has a heavy child, the heavy child is visited only once.

   For (i), the worst case occurs when b has $O(n(b))$ children and a heavy front child b_ℓ. In this case, the alternating paths of b's entering spiral $\xi_e(b)$ that have it visit each front child (excluding b_ℓ) may induce $O(n(b))$ segments on b's top face, and similarly for the alternating paths to each back child. Then b's return spiral $\xi_r(b)$ retraces these alternating paths up to the point that it reaches b_ℓ, which at most doubles the number of segments. After visiting b_ℓ, the exiting spiral $\xi_x(b)$ retraces the path $\xi_r(b)$ and then the path $\xi_e(b)$ in reverse back to point t on b, which again at most doubles the number of segments on b. Thus the total number of segments is $O(n(b))$.

For (ii), the maximum visits to light children occur when b has a heavy front child. In this case, $\xi_e(b)$ visits each light child once. Then $\xi_r(b)$ visits each light child at most once on its way to the heavy front child. After visiting the heavy front child, $\xi_x(b)$ retraces $\xi_r(b)$ and then retraces $\xi_e(b)$ to the entering point of b, thus visiting each light child at most twice more. Therefore, each light child is visited at most four times.

For (iii), if b has a heavy front child, then the path traversed by ξ (detailed in Section 4.1) immediately establishes that the heavy front child is visited only once. Similarly, if b has a heavy back child, the path detailed in Section 4.2 establishes that the heavy back child is visited exactly once.

Properties (i), (ii) and (iii) established above imply that $R(n(b))$ is determined by the larger of three quantities:

(a) the number of segments on b's top face induced during ξ's first visit to b;
(b) $4 \max_{i=1\ldots k} R(n(b_i))$, where $b_1, b_2, \ldots b_k$ are b's light children;
(c) $R(n(b_\ell))$, where b_ℓ is b's heavy child, if it has one.

A multiplier of four is necessary in case (b) because light children may be visited up to four times during b's first visit; no multiplier is necessary for the heavy child (c) because it is visited only once. For the base case, $R(1) = c$, for some constant $c > 1$, because the first visit of ξ to a leaf node band (as described in Section 3.2.1) induces a constant number of segments. And in general,

$$
\begin{aligned}
R(n(b)) &= \max\left\{ O(n(b)),\ 4 \max_{i=1\ldots k} R(n(b_i)),\ R(n(b_\ell)) \right\} \\
&\leq \max\left\{ O(n(b)),\ 4 \max_{i=1\ldots k} R\left(\tfrac{1}{2}n(b)\right),\ R(n(b)-1) \right\} \\
&= \max\left\{ O(n(b)),\ 4R\left(\tfrac{1}{2}n(b)\right),\ R(n(b)-1) \right\}
\end{aligned}
$$

noting that the light children's subtrees contain at most $\frac{1}{2}n(b)$ nodes, and the heavy child's subtree contains at most $n(b)-1$ nodes. It is straightforward to verify by induction that $R(n(b)) = O(n(b)^2)$. Applying this to the root r of T_U with $n = n(r)$ nodes and noting that ξ visits r only once in the delta-unfolding algorithm, yields a maximum of $O(n^2)$ parallel segments on any top face.

This also bounds the number of cuts on any grid face in the unfolding. Specifically, in the thickening step ξ expands in the $+y$ and $-y$ direction so as to cover the entire band, but this does not asymptotically increase its number of edges. After the thickening, disjoint sections of ξ run along the entirety of both band rims. In the partitioning step, the disjoint sections along the top rim edges induce the division of the frontward and rearward faces into strips; i.e., each disjoint section delimits the vertical strip beneath it. Because $O(n^2)$ bounds the number of disjoint sections along the top edge, it also bounds the number of strips a frontward/rearward face is partitioned into.

4.4.1 A worst case refinement example.

A simple example establishes that the bound $O(n^2)$ is tight: a polyhedron with $n = 2^{h+1} - 1$ blocks, whose unfolding tree T_U is a perfect binary tree of height h (i.e., each internal node has two children, and all leaves are at the same level). There are no heavy nodes in T_U, and the number of cuts in a visit of the root is given by the recurrence relation

$$R(n) = 4R((n-1)/2) = 4^h R(1) = (n+1)^2 R(1)/4 \ ,$$

because

$$4^h = 4^{\log_2(n+1)-1} = 2^{\log_2(n+1)^2}/4 = (n+1)^2/4 \ .$$

And since $R(1) = c$, for some constant c, it follows that $R(n) = O(n^2)$, establishing our claim.

5 Delta-Unfolding of Genus-Zero Orthogonal Polyhedra

The delta-unfolding algorithm and its refinement analysis generalizes to all genus-zero orthogonal polyhedra in the same way the epsilon-unfolding algorithm does, so we summarize the idea here and refer the reader to [6] for details. Instead of partitioning P into blocks, the general algorithm partitions P into slabs as defined in Section 2. It then creates an unfolding tree, T_U, where each node corresponds to a band surrounding a slab. Each parent-child arc in T_U corresponds to a z-beam, which is a vertical strip from a frontward or rearward face connecting the parent's rim to the child's rim. For a parent band b, its front (back) children are those whose z-beams connect to b's front (back) rim.

The spiral ξ enters and exits b at points s and t located at the intersection between b's front rim and the z-beam connecting b to its parent. Observe that there is a natural cyclic ordering of b's front (back) children that is determined by their z-beam connections around b's front (back) rim. Using this cyclic ordering, it is straightforward to generalize the paths that ξ follows to reach the front and back children, described in Sections 4.1 and 4.2. See for example Figure 11 that shows a band with its faces flattened in the plane (the lighter color marks top/bottom faces, and the darker color marks right/left faces). Also depicted are the z-beam connections (flattened into the plane) and the path $\xi_e(b)$ follows to visit the children, assuming b_ℓ is a heavy child. Observe that the path is the same as in Figure 8, except that it extends across multiple band faces. When ξ visits a child, it moves from b to the connecting z-beam and travels vertically (in 3D) along the z-beam to reach the child; when it exits the child it travels along the z-beam back to b. In the unfolded staircase, the portion of ξ on the z-beam corresponds to a vertical riser. Thickening ξ is done as in the case of extrusions. The partitioning of the forwards and rearwards faces is also done as in the case of extrusions, but in addition to shooting illuminating rays down from top rim edges, bottom rim edges that are not

hit by these rays must themselves shoot rays upward to illuminate portions of faces not illuminated by the top edges. The face pieces resulting from this partitioning method are hung from the staircase as described in Section 3.3.
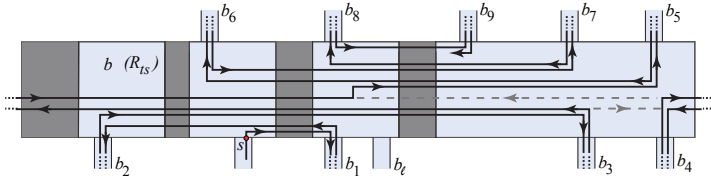
Fig. 11 Band b of a slab, cut and laid flat with top/bottom faces light gray and right/left faces dark gray. z-beam connections to b's parent, children $b_1 \ldots b_9$, and child b_ℓ are marked along the front and back rim. The path that $\xi_e(b)$ follows when b_ℓ is heavy is depicted.

The $O(n^2)$ upper bound on the level of grid refinement for extrusions also applies to general orthogonal polyhedra by the following argument. In the case of extrusions, for any block b with children, ξ makes turns only on b's top face, because all access to the children is from the top face; it makes no turns on the other three faces of b. Therefore, we analyze the number of segments (each corresponding to a turn) on the top face. For a band b of an arbitrary orthogonal polyhedra, ξ visits b's children in the same manner as for an extrusion, except that the turns made to access the children are made on whatever top or bottom face has the connecting z-beam, as in Figure 11. In particular, for a band b with a given number of front and back children, the same number of turns are made, whether b surrounds a block of an extrusion or a slab of a arbitrary orthogonal polyhedron. In terms of maximum refinement, the worst case occurs when all the turns are concentrated on a single face, which is exactly the situation handled by our upper bound analysis in the case of extrusions.

6 Conclusion

We present modifications to the epsilon-unfolding algorithm from [6] that reduce the level of grid refinement necessary to grid-unfold any genus-zero orthogonal polyhedron from exponential to quadratic. The next natural step is to seek a refined grid edge-unfolding of all genus-zero orthogonal polyhedra that requires subquadratic refinement of the grid faces, to date only achieved for highly restricted classes of orthogonal polyhedra [4,7,8]. It is unlikely that the technique used in this paper could be extended to produce such an unfolding, due to the backtracking nature of our recursive unfolding algorithm. However, our preliminary investigations embolden us to conjecture that a constant refinement of the vertex grid suffices to grid-unfold all orthogonal polyhedra.

References

1. Zachary Abel and Erik D. Demaine. Edge-unfolding orthogonal polyhedra is strongly np-complete. In *Proceedings of the 23rd Canadian Conference on Computational Geometry*, August 2011.
2. Greg Aloupis, Prosenjit K. Bose, Sebastien Collette, Erik D. Demaine, Martin L. Demaine, Karim Douieb, Vida Dujmović, John Iacono, Stefan Langerman, and Pat Morin. Common unfoldings of polyominoes and polycube. In *Revised Papers from the China-Japan Joint Conference on Computational Geometry, Graphs and Applications*, volume 7033 of *Lecture Notes in Computer Science*, pages 44–54, November 2010.
3. Marshall Bern, Erik D. Demaine, David Eppstein, Eric Kuo, Andrea Mantler, and Jack Snoeyink. Ununfoldable polyhedra with convex faces. *Comput. Geom.: Theory and Appl.*, 24(2):51–62, February 2003.
4. Therese Biedl, Erik Demaine, Martin Demaine, Anna Lubiw, Mark Overmars, Joseph O'Rourke, Steve Robbins, and Sue Whitesides. Unfolding some classes of orthogonal polyhedra. In *Proceedings of the 10th Canadian Conference on Computational Geometry*, August 1998.
5. Mirela Damian, Robin Flatland, Henk Meijer, and Joseph O'Rourke. Unfolding well-separated orthotrees. In *Abstracts from the 15th Annual Fall Workshop on Computational Geometry*, November 2005.
6. Mirela Damian, Robin Flatland, and Joseph O'Rourke. Epsilon-unfolding orthogonal polyhedra. *Graphs and Comb.*, 23(1):179–194, 2007.
7. Mirela Damian, Robin Flatland, and Joseph O'Rourke. Unfolding Manhattan towers. *Comput. Geom.: Theory and Appl.*, 40:102–114, 2008.
8. Mirela Damian and Henk Meijer. Edge-unfolding orthostacks with orthogonally convex slabs. In *Abstracts from the 14th Annual Fall Workshop on Computational Geometry*, pages 20–21, November 2004. `http://cgw2004.csail.mit.edu/talks/34.ps`.
9. Erik D. Demaine and Joseph O'Rourke. A survey of folding and unfolding in computational geometry. In Jacob E. Goodman, János Pach, and Emo Welzl, editors, *Discrete and Computational Geometry*, Mathematical Sciences Research Institute Publications, pages 167–211. Cambridge University Press, 2005.
10. Erik D. Demaine and Joseph O'Rourke. *Geometric Folding Algorithms: Linkages, Origami, Polyhedra.* Cambridge University Press, July 2007.
11. Joseph O'Rourke. Unfolding orthogonal polyhedra. In J. E. Goodman, J. Pach, and R. Pollack, editors, *Surveys on Discrete and Computational Geometry: Twenty Years Later*, pages 231–255. American Mathematical Society, 2008.
12. Daniel D. Sleator and Robert Endre Tarjan. A data structure for dynamic trees. *J. of Comput. and Syst. Sci.*, 24(3):362–391, June 1983.