Discrete Comput Geom (2008) 39: 213–238
DOI 10.1007/s00454-007-9043-9

Grid Vertex-Unfolding Orthogonal Polyhedra

Mirela Damian · Robin Flatland · Joseph O'Rourke

Received: 27 September 2006 / Revised: 1 November 2007 /
Published online: 27 November 2007

Abstract An *edge-unfolding* of a polyhedron is produced by cutting along edges and flattening the faces to a *net*, a connected planar piece with no overlaps. A *grid unfolding* allows additional cuts along grid edges induced by coordinate planes passing through every vertex. A *vertex-unfolding* allows faces in the net to be connected at single vertices, not necessarily along edges. We show that any orthogonal polyhedra of genus zero has a grid vertex-unfolding. (There are orthogonal polyhedra that cannot be vertex-unfolded, so some type of "gridding" of the faces is necessary.) For any orthogonal polyhedron P with n vertices, we describe an algorithm that vertex-unfolds P in $O(n^2)$ time. Enroute to explaining this algorithm, we present a simpler vertex-unfolding algorithm that requires a $3 \times 1 \times 1$ refinement of the vertex grid.

Keywords Vertex-unfolding · Grid unfolding · Orthogonal polyhedra · Genus-zero

This is a significant revision of the preliminary version that appeared in [2].

J. O'Rourke's research was supported by NSF award DUE-0123154.

M. Damian (✉)
Dept. Comput. Sci., Villanova Univ., Villanova, PA 19085, USA
e-mail: mirela.damian@villanova.edu

R. Flatland
Dept. Comput. Sci., Siena College, Loudonville, NY 12211, USA
e-mail: flatland@siena.edu

J. O'Rourke
Dept. Comput. Sci., Smith College, Northampton, MA 01063, USA
e-mail: orourke@cs.smith.edu

1 Introduction

Two unfolding problems have remained unsolved for many years [3, 5]: (1) Can every convex polyhedron be edge-unfolded? (2) Can every polyhedron be unfolded? An *unfolding* of a 3D object is an isometric mapping of its surface to a single, connected planar piece, the "net" for the object, that avoids overlap. An *edge-unfolding* achieves the unfolding by cutting edges of a polyhedron, whereas a *general-unfolding* places no restriction on the cuts. A net representation of a polyhedron finds use in a variety of applications [8]—from flattening monkey brains [10] to manufacturing, from sheet metal [12] to low-distortion texture mapping [11].

It is known that some nonconvex polyhedra cannot be unfolded without overlap with cuts along edges. However, no example is known of a nonconvex polyhedron that cannot be unfolded with unrestricted cuts. Advances on these difficult problems have been made by specializing the class of polyhedra, or easing the stringency of the unfolding criteria. On one hand, it was established in [1] that certain subclasses of *orthogonal polyhedra*—those whose faces meet at right angles and whose edges are parallel to coordinate axes—that are multiples of 90°—have an unfolding. In particular, the class of *orthostacks*, stacks of extruded orthogonal polygons, was proven to have an unfolding (but not an edge-unfolding). On the other hand, loosening the criteria of what constitutes a net to permit connection through points/vertices, the so-called *vertex-unfoldings*, led to an algorithm to vertex-unfold any triangulated manifold [6] (and indeed, any simplicial manifold in higher dimensions). A vertex unfolding maps the surface to a single, connected piece K in the plane, but K may have "cut vertices" whose removal disconnects K.

A second loosening of the criteria is the notion of grid unfoldings, which are especially natural for orthogonal polyhedra. A *grid unfolding* adds edges to the surface by intersecting the polyhedron with planes parallel to Cartesian coordinate planes through every vertex. The two approaches were recently married in [7], which established that any orthostack may be grid vertex-unfolded. For orthogonal polyhedra, a grid unfolding is a natural median between edge-unfoldings and unrestricted unfoldings.

Our main result is that any orthogonal polyhedron, without shape restriction except that its surface be homeomorphic to a sphere, has a grid vertex-unfolding. We present an algorithm that grid vertex-unfolds any orthogonal polyhedron with n vertices in $O(n^2)$ time. We also present, along the way, a simpler algorithm for $3 \times 1 \times 1$ *refinement* unfolding, a weakening of grid unfolding that we define in the following. We believe that the techniques in our algorithms may help show that all orthogonal polyhedra can be grid edge-unfolded.

2 Definitions

We distinguish between a *strict net*, in which the net boundary does not self-touch, and a *net* for which the boundary may touch but no interior points overlap. The latter corresponds to the physical model of cutting out the net from a sheet of paper, with perhaps some cuts representing *edge overlap*, and this is the model we use in this paper. We also insist as part of the definition of a vertex-unfolding, again keeping

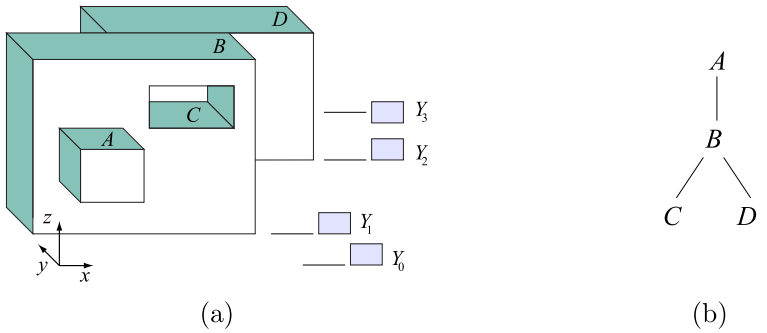

Fig. 1 Definitions. **a** Shaded connected pieces are bands; A, B and D are protrusions; C is a dent. **b** An unfolding tree captures band adjacency structure and determines the algorithm's recursive calls

in spirit with the physical model, that the unfolding "path" never self-crosses on the surface in the following sense. If (A, B, C, D) are four gridfaces incident in that cyclic order to a common vertex v, then the net does not include both the connections AvC and BvD.[1]

We use the following notation to describe the six type of faces of an orthogonal polyhedron, depending on the direction in which the outward normal points: *front*: $-y$; *back*: $+y$; *left*: $-x$; *right*: $+x$; *bottom*: $-z$; *top*: $+z$. We take the z-axis to define the vertical direction; *vertical* faces are parallel to the xz-plane or the yz plane. Directions clockwise and counterclockwise are defined from the perspective of a viewer positioned at $y = -\infty$. We distinguish between an original vertex of the polyhedron, which we call a *corner vertex* or just a *vertex*, and a *gridpoint*, a vertex of the grid (which might be an original vertex). A *gridedge* (*gridface*) is an edge (face) of the grid that lies on the surface of the polyhedron.

A $k_1 \times k_2 \times k_3$ *refinement* of a surface [4] starts with a grid unfolding and further partitions each gridface into a grid of edges. Positive integers k_1, k_2, and k_3 are associated with the amount of refinement in the x, y, and z dimensions, respectively; e.g., z-perpendicular gridfaces are refined into a $k_1 \times k_2$ grid, and similarly x-perpendicular (y-perpendicular) gridfaces are refined into a $k_2 \times k_3$ ($k_1 \times k_3$) grid. We will consider refinements of grid unfoldings, with the convention that a $1 \times 1 \times 1$ refinement is an unrefined grid unfolding.

Let O be a solid orthogonal polyhedron with the surface homeomorphic to a sphere (i.e., genus zero). Let Y_i be the plane $y = y_i$ orthogonal to the y-axis. Let $Y_0, Y_1, \ldots, Y_i, \ldots$ be the finite sequence of parallel planes passing through every vertex of O, with $y_0 < y_1 < \cdots < y_i < \cdots$. We define *layer* i to be the portion of O between planes Y_i and Y_{i+1}. Observe that a layer may include a collection of disjoint connected components; we call each such component a *slab*. The *band* of a slab is the connected surface piece composed of gridfaces parallel to the y axis that surround the slab. Referring to Fig. 1a, layer 0, 1, and 2 each contain one slab (with outer bands A, B, and D, respectively). Note that each slab is bounded by an outer (surface) band, but it may also contain inner bands, bounding holes. Outer bands are

[1] This was not part of the original definition in [6] but was achieved by those unfoldings.

called *protrusions* and inner bands are called *dents* (C in Fig. 1a). In other words, band A is a *protrusion* if a traversal of the rim of A in Y_i, counterclockwise from the viewpoint of $y = -\infty$, has the interior of O to the left of A, and a *dent* if this traversal has the interior of O to the right.

For each i, define $P_i = \partial O \cap Y_i$ as the portion of the surface of O lying in plane Y_i. P_i^+ is the portion of P_i with normal in direction $+y$ (composed of back faces), and P_i^- the portion with normal in direction $-y$ (composed of front faces). By convention, band points in P_i that are not incident to either front or back faces (e.g., when one band aligns with another), belong to both P_i^+ and P_i^-. Thus $P_i = P_i^+ \cup P_i^-$.

3 Dents vs. Protrusions

We observe that dents may be treated exactly the same as protrusions with respect to unfolding, because an unfolding of a 2-manifold to another surface (in our case, a plane) depends only on the intrinsic geometry of the surface, and not on how it is embedded in $\mathbb{R}^3$. Note that we are concerned only with the final unfolded "flat state" [3, 5], and not with possible intersections during a continuous sequence of partially unfolded intermediate states. Our unfolding algorithm relies solely on the amount of surface material surrounding each point: the cyclic ordering of the gridfaces incident to a vertex, and the pair of gridfaces sharing a gridedge. All these local relationships remain unchanged if we conceptually "pop-out" dents to become protrusions, i.e., a "Flatland" creature living in the surface could not tell the difference; nor can our algorithm. We note that the popping-out is conceptual only, for it could produce self-intersecting objects. Also dents are gridded independently of the rest of the object so as to avoid unnecessary surface cuts that would correspond to y-planes containing dent vertices only. From the point of view of unfolding, it does not matter whether dents are popped out or not.

Although the dent/protrusion distinction is irrelevant to the unfolding, the interrelationships between dents and protrusions touching a particular Y_i do depend on this distinction. To cite just the simplest example, there cannot be two nested protrusions to the same side of Y_i, but a protrusion could have a dent in it to the same side of Y_i (e.g., protrusion B encloses dent C to the same side of Y_1 in Fig. 1a). These relationships are crucial to the connectivity of the band graph G_b, discussed in Appendix.

4 Overview

The two algorithms we present share a common central structure, with the second achieving a stronger result; both are vertex-unfoldings that use orthogonal cuts only. We note that it is the restriction to orthogonal cuts that makes the vertex-unfolding problem difficult: if arbitrary cuts are allowed, then a general vertex-unfolding can be obtained by simply triangulating each face and applying the algorithm from [6].

The $(3 \times 1 \times 1)$-algorithm unfolds any genus-0 orthogonal polyhedron that has been refined in one direction 3-fold. The bands themselves are never split (unlike in [1]). The algorithm is simple. The $(1 \times 1 \times 1)$-algorithm also unfolds any genus-0 orthogonal polyhedron, but this time achieving a grid vertex-unfolding, i.e., without

refinement. This algorithm is more delicate, with several cases not present in the $(3 \times 1 \times 1)$-algorithm that need careful detailing. Clearly this latter algorithm is stronger, and we vary the detail of presentation to favor it. The overall structure of the two algorithms is the same:

1. A band "unfolding tree" T_u is constructed by shooting rays vertically from the top of bands. The root of T_u is a *frontmost* band (of smallest y-coordinate), with ties broken arbitrarily.
2. A forward and return *connecting* path of vertical front/back gridfaces is identified, each of which connects a parent band to a child band in T_u.
3. Each band is unfolded horizontally as a unit, but interrupted when a connecting path to a child is encountered. The parent band unfolding is suspended at that point, and the child band is unfolded recursively.
4. The vertical front and back faces of each slab are partitioned according to an illumination model, with variations for the more complex $(1 \times 1 \times 1)$-algorithm. Front/back gridfaces are attached below and above appropriate horizontal sections of the band unfolding.

The final unfolding lays out all bands horizontally, with the front and back gridfaces hanging below and above the bands. Nonoverlap is guaranteed by this strict two-direction structure.

Although our result is a broadening of that in [7] from orthostacks to all orthogonal polyhedra, we found it necessary to employ techniques different from those used in that work. The main reason is that, in an orthostack, the adjacency structure of bands yields a path, which allows the unfolding to proceed from one band to the next along this path, never needing to return. In an orthogonal polyhedron, the adjacency structure of bands is generally not linear. Thus in our algorithm, unfolding band-by-band leads to a tree traversal (e.g., Fig. 1b), which requires traversing each arc in both directions. It is this aspect which we consider our main novelty, and which leads us to hope for an extension to edge-unfoldings as well.

5 $(3 \times 1 \times 1)$-Algorithm

5.1 Computing the Unfolding Tree T_u

Define a *z-beam* to be a front or back rectangle on the surface of O whose top and bottom edges are gridedges on two bands. In the degenerate case, a z-beam has height zero and connects two rims along a section where they coincide. We say that two bands, b_i and b_j, are *z-visible* if there is a z-beam connecting a gridedge of b_i to a gridedge of b_j. There can be many z-beams connecting two bands, so for each pair of bands we select a representative z-beam of minimal (vertical) height. Let G be the graph that contains a node for each band of O and an arc for each pair (b_i, b_j) of z-visible bands such that $i \neq j$.

Lemma 1 *G is connected.*

Proof First observe that every gridface of O is either part of a band or part of a z-beam (possibly a z-beam connecting a band to itself). Now consider making vertical

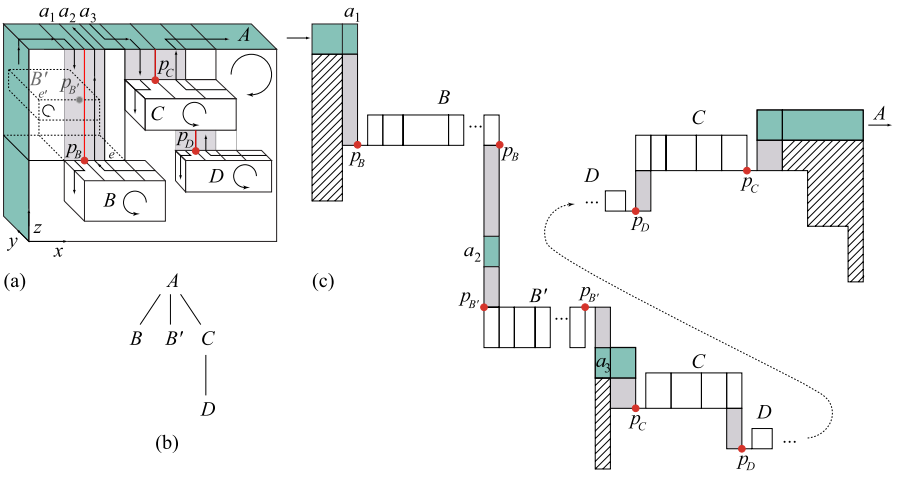

Fig. 2 **a** Orthogonal polyhedron. **b** Unfolding tree T_u. **c** Unfolding of bands and front (hachured) gridface pieces connecting to A. Vertex connection through the pivots points p_B, $p_{B'}$, p_C, p_D is shown exaggerated for clarity

cuts on the surface of O along the extent of the left and right sides of each z-beam. Since O is connected and only vertical cuts are made, the resulting structure remains connected and can be viewed as a multigraph, where bands are nodes and z-beams are edges. Since G is the subset of this multigraph obtained by removing self-loops and duplicate edges, G is also connected. □

Let the unfolding tree T_u be any spanning tree of G, with the root selected arbitrarily from among all bands adjacent to Y_0. We apply the $3 \times 1 \times 1$ refinement procedure to partition each front, back, top, and bottom gridface of O into three congruent subfaces, by adding two new gridedges orthogonal to the x-axis. This partitions the top and bottom edges of each z-beam into three refined gridedges and divides the beam itself into three vertical columns of refined gridfaces. See Fig. 2a. Let A be an arbitrary band, let B be one of its children in T_u, and let e be the gridedge on B's rim where the z-beam from A attaches. We define the *pivot point* p_B for band B to be the $\frac{1}{3}$-point of e (or, in circumstances to be explained later, the $\frac{2}{3}$-point), and so it coincides with a point of the $3 \times 1 \times 1$-refined grid. The unfolding of O will follow the connecting vertical ray that extends from p_B on B to A. Note that if e belongs to both A and B, then the ray connecting A and B degenerates to a point. To either side of a connecting ray we have two *connecting paths* of gridfaces, the *forward* and *return* path. In Fig. 2a, these connecting paths are the shaded strips on the front face of A.

5.2 Unfolding Bands into a Net

Starting at a frontmost *root band*, each band is unfolded as a conceptual unit, but interrupted by the connecting rays incident to it from its front and back faces. In Fig. 2, band A is unfolded as a rectangle, but interrupted at the rays connecting to

(front children) B, C and (back child) B'. At each such ray the parent band unfolding is suspended, the unfolding follows the forward connecting path to the child, the child band is recursively unfolded, then the unfolding returns along the return connecting path back to the parent, resuming the parent band unfolding from the point it left off.

Figure 2 illustrates this unfolding algorithm. The clockwise unfolding of A, laid out horizontal to the right, is interrupted to traverse the forward path down to B, and B is then unfolded as a rectangle (composed of its contiguous gridfaces). The base p_B of the connecting ray is called a *pivot point* because the counterclockwise unfolding of B is rotated 180° counterclockwise about p_B so that the unfolding of B is also to the right. It is only here that we use point-connections that render the unfolding a vertex-unfolding. The unfolding of B proceeds counterclockwise back to p_B, crosses over A to unfold B', then a clockwise rotation by 180° around the second image of pivot $p_{B'}$ orients the return path to A so that the unfolding of A continues to the right. Note that the unfolding of C is itself interrupted to unfold child D. Also note that there is edge overlap in the unfolding at each of the pivot points, and this overlap could not be eliminated without violating the condition that all surface pieces face the same way (up, in our case).

The reason for the $3 \times 1 \times 1$ refinement is that the upper edge e' of the back child band B' has the same (x, z)-coordinates as the upper edge e of B on the front face. In this case, the gridfaces of band A induced by the connecting paths to B would be "overutilized" if there were only two. Let a_1, a_2, a_3 be the three faces of A induced by the $3 \times 1 \times 1$ refinement of the connecting path to B, as in Fig. 2. Then the unfolding path winds around A to a_1, follows the forward connecting path to B, returns along the return connecting path to a_2, crosses over A and unfolds B' on the back face, with the return path now joining to a_3, at which point the unfolding of A resumes. In this case, the pivot point $p_{B'}$ for B' is the $\frac{2}{3}$-point of e'. Other such conflicts are resolved similarly. It is now easy to see that the resulting net has the general form illustrated in Fig. 2c:

1. The faces of each band fall within a horizontal rectangle whose height is the band width.
2. These band rectangles are joined by front/back connecting paths on either side, connecting through pivot points.
3. The strip of the plane above and below each band face that is not incident to a connecting path, is empty.
4. The net is therefore an orthogonal polygon monotone with respect to the horizontal.

5.3 Attaching Front and Back Faces to the Net

Finally, we "hang" front and back faces from the bands as follows. The front face of each band A is partitioned by imagining A to illuminate downward lightrays from the rim in the front face. The pieces that are illuminated are then hung vertically downward from the horizontal unfolding of the A band. The portions unilluminated will be attached to the obscuring bands.

In the example in Fig. 2, this illumination model partitions the front face of A into three pieces (the striped pieces in Fig. 2c). These three pieces are attached under A;

the portions of the front face obscured by B but illuminated downward by B are hung beneath the unfolding of B (not shown in the figure), and so on. Because the vertical illumination model produces vertical strips, and because the strips above and below the band unfoldings are empty, there is always room to hang the partitioned front face. Thus, any orthogonal polygon may be vertex-unfolded with a $3 \times 1 \times 1$ refinement of the vertex grid.

Although we believe this algorithm can be improved to $2 \times 1 \times 1$ refinement, the complications needed to achieve this are similar to what is needed to avoid refinement entirely, so we instead turn directly to $1 \times 1 \times 1$ refinement.

6 $(1 \times 1 \times 1)$-Algorithm

Although the $(1 \times 1 \times 1)$-algorithm follows the same general outline as the $(3 \times 1 \times 1)$-algorithm, there are significant complications, which we outline before going into detail. First, without the refinement of z-beams into three strips to allow avoidance of conflicts on opposite sides of a slab (e.g., B and B' in Fig. 2a), we found it necessary to replace the z-beams by a pair of z-rays that are in some sense the boundary edges of a z-beam. Selecting two rays per band permits a 2-coloring algorithm (Theorem 4) to identify rays that avoid conflicts. Generating the ray-pairs (Sect. 6.1.1) requires care to ensure that the band graph G_b is connected (Appendix). This graph, and the 2-coloring, lead to an unfolding tree T_u (Sect. 6.2). From here on, there are fewer significant differences compared to the $(3 \times 1 \times 1)$-algorithm. Without the luxury of refinement, there is more need to share vertical paths on the front or back face of a slab (Fig. 11). Finally, the connecting paths obscure the illumination of some grid faces, which must be attached to the connecting paths. We now present the details, in this order:

1. Determine Conflict-Free Pivot Points (Sect. 6.1) via
    a. Ray-Pair Generation (Sect. 6.1.1)
    b. Ray Graph (Sect. 6.1.2)
2. Construct T_u (Sect. 6.2)
3. Select Connecting Paths (Sect. 6.2.1)
4. Determine Unfolding Directions (Sect. 6.2.2)
5. Recurse:
    a. Unfold Bands into a Net (Sect. 6.3)
    b. Attach Front and Back Faces to the Net (Sect. 6.4)

6.1 Determining Conflict-Free Pivot Points

The pivot p_A for a band A is the gridpoint of A where the unfolding of A starts and ends. The y-edge of A incident to p_A is the first edge of A that is cut to unfold A.

Let A be an arbitrary band delimited by planes Y_i and Y_{i+1}. Say that two gridpoints $u \in Y_i$ and $v \in Y_{i+1}$ are *in conflict* if the upward rays emerging from u and v hit first the endpoints of the same y-edge of A; otherwise, u and v are *conflict-free*. If u lies either on a vertical edge, or on a vertically extreme horizontal edge, then the ray at u degenerates to u itself.

Our goal is to select conflict-free pivots for all bands in T_u, which will help us later avoid competition over the use of certain gridfaces in the unfolding, an issue that will become clear in Sect. 6.3. Selecting these pivots is the most delicate aspect of the $(1 \times 1 \times 1)$-algorithm. Ultimately, we represent pivoting conflicts in the form of a graph G_r (Sect. 6.1.2), from which T_u will be derived.

6.1.1 Ray-Pair Generation

In order to avoid pivoting conflicts, for each band we will need two choices for its connecting ray. Thus the algorithm generates the rays in pairs. Because there is no refinement, the two rays originate at grid points on the same band, but they may terminate on different bands. A simple example is shown in Fig. 3a, where the ray pair originating on band D hits two different bands, B and C. This example also suggests that one cannot consider ray pairs connecting pairs of bands, as in the $(3 \times 1 \times 1)$-algorithm (which would connect D to A in this example), but instead we focus on shooting pairs of rays upward from strategic locations on the boundary of each band, and then selecting a subset of these rays so that the conflicts can be resolved and T_u is connected. To ensure connectedness of all bands, several ray-pairs must be issued upward from each band. Figure 3b shows an example: no pair of rays can emanate upward from the top of $B \cap P_i^-$ or $C \cap P_i^-$; one pair of rays shoots upward from the top of each component of $A \cap P_i^-$: (r_1, r_2) connects A to B and (r_3, r_4) connects A to C; finally, one pair of rays (r_5, r_6) issues from the top of $A \cap P_i^+$, which connects A to D. So, overall, three pairs of rays are generated for band A. We now turn to describing in detail the method for generating ray-pairs.

Let band A intersect plane Y_i. The algorithm is a for-loop over all A. We identify *chunks*, $A_1, A_2, \ldots, A_m$, of the rim $A \cap Y_i$, where each chunk A_j is a connected component of either $A \cap P_i^-$ or $A \cap P_i^+$ that contains at least one horizontal gridedge. (Note that these chunks do not necessarily cover $A \cap Y_i$.) We define $S(A_j)$ as the set of all vertical segments $s = (a, b)$, with $a \in A_j$, such that

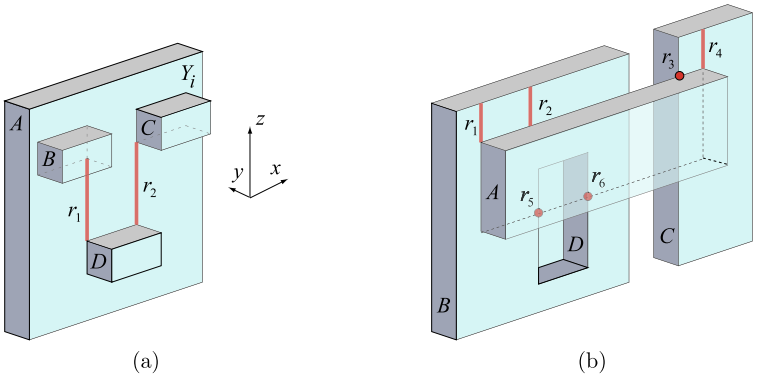

Fig. 3 **a** The ray pair (r_1, r_2) connects band D to two different bands B and C. **b** To ensure connectivity, three pairs of rays must be issued for A: (r_1, r_2), (r_3, r_4), and (r_5, r_6)

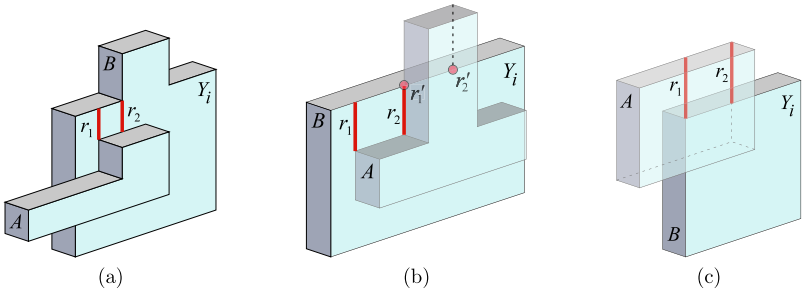

Fig. 4 Generating ray-pairs: **a** (r_1, r_2) for A; $S(B) = \emptyset$. **b** (r_1, r_2) for A (note that r_2 runs along source band A); degenerate ray-pair (r_1', r_2') for B. **c** $S(A) = \emptyset$; (r_1, r_2) for B

1. s is either a point, with $b = a$, or a front/back segment, with a below b.
2. $b \in B$ for some band $B \neq A$.
3. The open segment $s \setminus \{a, b\}$ may contain points of A (see r_2 in Fig. 4b), but no points of other bands.

For each band A, for each chunk $A_j \subseteq A$, if $S(A_j)$ contains at least two rays connecting A to the same band B, we select one ray pair (r_1, r_2) that satisfies two restrictions: (i) among the segments in $S(A_j)$ incident to a highest x-gridedge in A_j, r_1 is the leftmost one, and (ii) r_2 is the segment one x-gridedge to the right of r_1. Figure 4 shows a few examples. As mentioned earlier, several ray pairs could be generated for any one band, and indeed several pairs could connect two bands (e.g., see Fig. 4b where bands A and B are connected by two ray pairs).

Let G_b be the *band graph* whose nodes are bands. Two bands are connected by an arc in G_b if the ray-pair algorithm generates a ray connecting them. We call a collection of bands in G_b *ray-connected* if they are in the same connected component of G_b. We establish that G_b is a connected graph, i.e., all bands are ray-connected to one another, even if only one ray per pair is employed:

Lemma 2 *G_b is connected. Furthermore, the subgraph of G_b induced by exactly one ray per ray-pair (arbitrarily selected) is connected.*

Whereas the connectedness of bands by z-beams in the $(3 \times 1 \times 1)$-algorithm is straightforward, the complex possible relationships between bands makes connectedness via rays more subtle. We relegate the proof to the Appendix (Appendix) in order to not interrupt the main flow of the algorithm.

The over-generation of ray-pairs noted above is designed to ensure connectedness. Eventually many rays will be discarded by the time T_u is constructed in Sect. 6.2.

6.1.2 Ray Graph G_r

One pair of rays per pair of bands suffices to ensure that all bands are ray-connected. If multiple pairs of rays exist for a pair of bands, pick one pair arbitrarily and discard the rest. Then define a ray graph G_r as follows. The nodes of G_r are vertical rays, perhaps degenerating to points, connecting gridpoints between two bands that both intersect a common Y_i plane. The arcs of G_r record two types of potential pivoting conflicts:

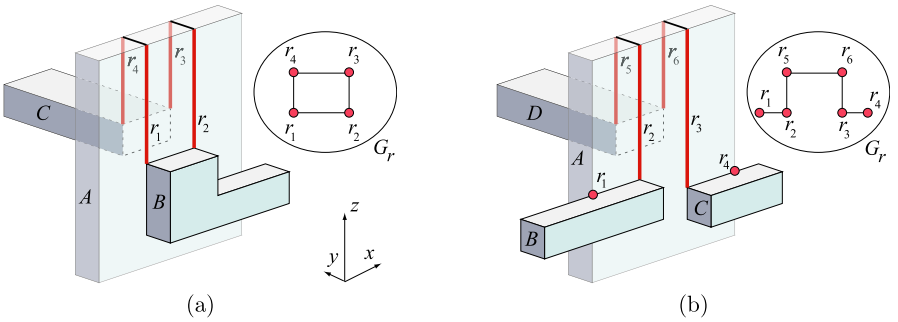

Fig. 5 Building G_r. **a** G_r is a 4-cycle; $\{r_1, r_2\}$ and $\{r_3, r_4\}$ are x-arcs and the others are y-arcs. **b** G_r is a path; $\{r_2, r_5\}$ and $\{r_3, r_6\}$ are y-arcs and the others are x-arcs

(i) The nodes for each pair of rays issuing from the top of a band B are adjacent in G_r. Call such arcs *x-arcs*; geometrically they can be viewed as parallel to the x-axis.
(ii) The nodes for two rays incident to opposite sides of the rim of a band A, connected by a y-segment on the band, are adjacent in G_r. Call such arcs *y-arcs*; geometrically they can be viewed as parallel to the y-axis.

Figure 5 shows two simple examples of G_r involving nodes on opposite sides of one band A. Before proceeding, we list the consequences of the two types of arcs in G_r. Assuming that we can 2-color G_r {red, blue}, and we select the base of (say) the red rays as pivots, then: (i) exactly one pivot is selected for each band, and (ii) no two pivot rays are in conflict across a band. So our goal now is to show that G_r is 2-colorable. Because a graph is 2-colorable if and only if it is bipartite, and a graph is bipartite if and only if every cycle is of even length, we aim to prove that every cycle in G_r is of even length. We start by listing a few relevant properties of G_r:

1. Every node $r \in G_r$ has exactly one incident x-arc. The rays are generated in pairs, and the pairs are connected by an x-arc. As no such ray is shared between two bands, at most one x-arc is incident to any r.
2. Nodes have at most degree 3, with the following structure: degree-1 nodes have an incident x-arc; degree-2 nodes have both an incident x- and y-arc; and degree-3 nodes have an incident x-arc and two incident y-arcs.
3. Each x-arc spans exactly one pair of adjacent y-gridlines, and each y-arc spans exactly one band rim-to-rim. The former is by the definition of ray pairs, which issue from adjacent gridpoints, and the latter follows from the grid partitioning of the object into bands.

Our next step requires embedding G_r in an xy-plane Π. Toward that end, we coordinatize the nodes and arcs of G_r as follows. A node $r \in G_r$ is a z-ray, and is assigned the (x, y) coordinates of the ray. Note that this means collinear rays get mapped to the same point; however, we treat them as distinct. The x-arcs are then parallel to the x-axis, and the y-arcs are parallel to the y-axis. In essence, this coordinatization is a view from $z = +\infty$.

Figure 6 shows a more complex example illustrating this viewpoint. The object is composed of 7 bands B_i, one of which (B_3) is a dent. There are 12 ray nodes,

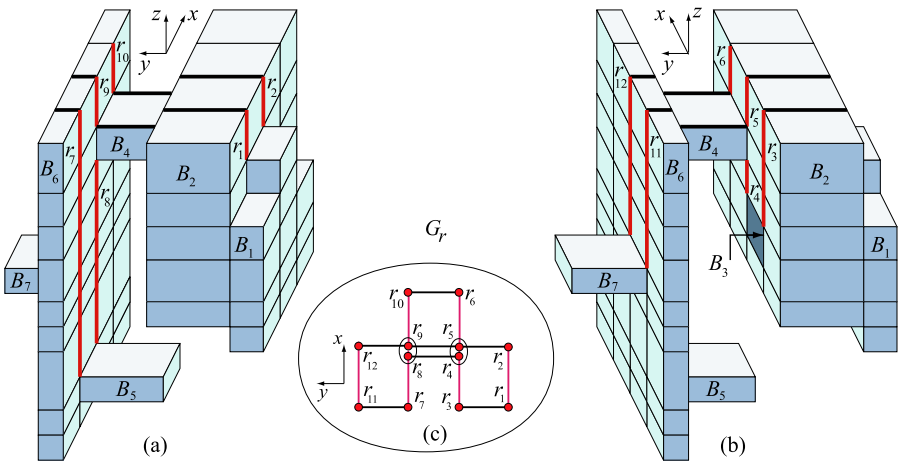

Fig. 6 **a**, **b** Two side views of an object; z-rays and y-arcs are marked with thick lines. **c** G_r coordinatized into xy-plane Π; (r_5, r_6, r_{10}, r_9) is a 4-cycle; $(r_1, r_3, r_4, r_8, r_7, r_{11}, r_{12}, r_9, r_5, r_2)$ is a 10-cycle

two pairs of which lie on the same z-vertical line, namely (r_4, r_5) and (r_8, r_9). Note that there are y-arcs crossing both the top of and the bottom[2] of B_4. The graph G_r has a 4-cycle and a 10-cycle, both detailed in the caption (as well as a 12-cycle not detailed).

Lemma 3 *Every cycle in G_r is of even length.*

Proof Let C be a cycle in G_r. The coordinatization described above maps C to a (perhaps self-crossing) closed path in the xy-plane Π, a path which may visit the same (x, y) point more than once, and/or traverse the same edge in Π more than once. Any such closed path on a grid must have even length, for the following reason.

First, by Property (3) above, each edge of the path in Π connects adjacent grid lines: an edge never "jumps over" one or more grid lines. Second, any such closed lattice path changes parity with each step, in the following sense. Number the x- and y-gridlines with integers $0, 1, 2, \ldots$ left to right and bottom to top, respectively. Define the parity of a gridpoint of Π to be the sum of its x- and y-gridline coordinates, mod 2. Then each step of the path, necessarily in one of the four compass directions, changes parity, as it changes only one of x or y. Returning to the start point to close the path must return to the starting coordinates, and so to the same parity. Thus, there must be an even number of parity changes along any closed path. Therefore, C has an even number of edges. □

We have now established this:

Theorem 4 *G_r is 2-colorable.*

[2] A dent is included in this example precisely to introduce such a bottom y-arc into G_r.

Note that nowhere in the above proof do we assume genus zero, so this theorem holds for polyhedra of arbitrary genus.

Band Pivoting We are finally ready to specify the pivot points. By Theorem 4, we can 2-color the nodes of G_r {red, blue}. We choose all red ray-nodes of G_r to be pivoting rays, in that their base points become pivot points. As remarked before, this selection guarantees that each band is pivoted, and no two pivots are in conflict. For the root band we choose a pivot point—the point at which the unfolding starts and ends—to be a grid point on the front rim connected by a y-segment to a blue ray. Because the rays are generated in pairs, there must be a blue ray incident to the root band. This choice guarantees that the root pivot is not in conflict with any other (necessarily red) pivot.

6.2 Unfolding Tree T_u

The next task is to define a band spanning tree T_u, based on the band graph G_b. Define G'_b, to retain just the arcs of G_b corresponding to the red ray nodes (in the above 2-coloring) in G_r. This maintains the connectivity by Lemma 2. Then take T_u to be any spanning tree of G'_b rooted at a frontmost band. The arcs in T_u and their associated rays thus determine a pivot point for each band.

With T_u finally in hand, the remainder of the $(1 \times 1 \times 1)$-algorithm follows the overall structure of the $3 \times 1 \times 1$ algorithm, with variations as mentioned before, as detailed below.

6.2.1 Selecting Connecting Paths

Having established a pivot point for each band, we are now ready to define the *forward* and *return* connecting paths for a child band in T_u. A "path" here refers to a connected sequence of gridfaces that the unfolding follows to get from one band to another. Let B be an arbitrary child of band A. If the pivot point p_B of B is at the intersection of B and A, then both forward and return connection paths for B reduce to point p_B (see Fig. 7). If B does not intersect A, then a ray r connects p_B to A (Figs. 8a and 10a). The connecting paths are the two vertical paths separated by r composed of the gridfaces sharing an edge with r (paths k_1 and k_2 in Figs. 8a and 10a). The path first encountered in the unfolding of A is used as a forward connecting path; the other path is used as a return connecting path.

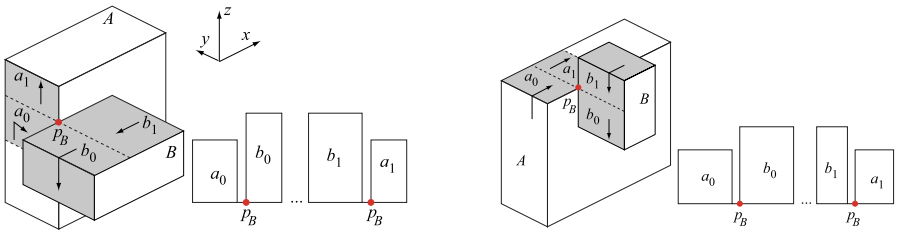

Fig. 7 Unfolding B when the ray connecting B to A degenerates to p_B

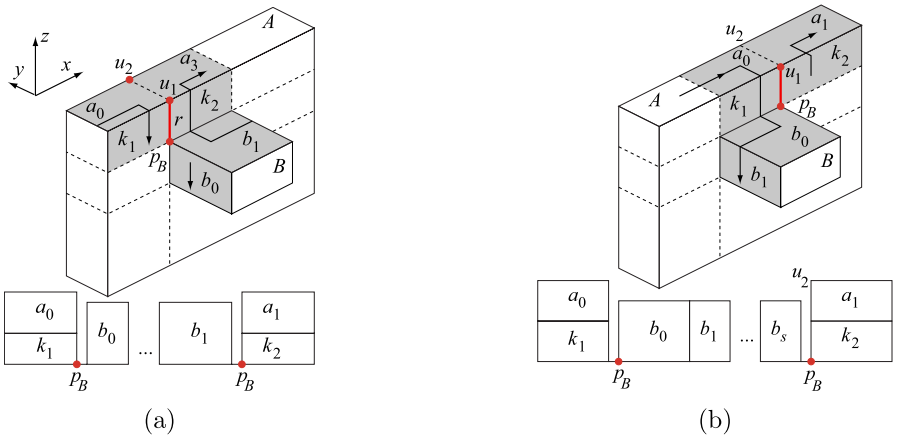

Fig. 8 Unfolding B: u_1 is not a corner vertex of A **a** p_B incident to a left gridface of B **b** p_B incident to a top gridface of b

6.2.2 Determining Unfolding Directions

A top-down traversal of T_u assigns an unfolding direction to each band in T_u as follows. The root band in T_u may unfold either clockwise or counterclockwise, but for definiteness we set the unfolding direction to clockwise. Let B be the band in T_u currently visited and let A be the parent of B. If the upward ray r incident to p_B connects B to a bottom gridpoint of A, then B unfolds in the same clockwise/counterclockwise direction as A. Otherwise, r connects B to a top or a side (for degenerate rays) gridpoint of A; in this case, B unfolds in the direction opposite to that of A. In other words, A and B unfold in the same direction if B "hangs below" A, and in opposite direction otherwise.

6.3 Unfolding Bands into a Net

Let A be a band to unfold, initially the root band. The unfolding of A starts at its pivot point p_A and proceeds in the unfolding direction (clockwise or counterclockwise) of A. Henceforth we assume without loss of generality that the unfolding of A proceeds clockwise (with respect to a viewpoint at $y = -\infty$); the counterclockwise unfolding of A is a vertical reflection of the clockwise unfolding of A. In the following we describe our method to unfold every child B of A recursively. As mentioned earlier, each band unfolds horizontally, from left to right, with recursive interruptions to unfold its children.

Without loss of generality, we assume that A and B are both protrusions (cf. Sect. 3). The possible unfoldings for a child B fall naturally into three cases. Case 1 handles the situation when B's pivot is at the intersection of A and B. Cases 2 and 3 handle situations when B's pivot is connected by a ray to A; Case 2 deals with situations in which B's connecting paths do not overlap any other connecting paths, and Case 3 addresses overlapping paths.

Case 1: Pivot $p_B \in A \cap B$. Then, whenever the unfolding of A reaches p_B, we unfold B as in Fig. 7. The unfolding uses the two band gridfaces of A incident to p_B

(a_0 and a_1 in Fig. 7). Let b_0 be the first gridface of B in counterclockwise order about p_B. In the unfolding, we rotate b_0 around p_B so that the counterclockwise unfolding of B extends horizontally to the right. The unfolding of B proceeds counterclockwise back to p_B, then the band gridface a_1 incident to p_B is oriented about p_B so that the unfolding of A continues to the right.

Note that, because the pivots of any two children of A are conflict-free, there is no competition over the use of a_0 and a_1 in the unfolding. Note also that the unfolding path does not self-cross. For example, the cyclic order of the gridfaces incident to p_B in Fig. 7a is $(a_0, A_{\text{front}}, b_0, b_1, A_{\text{back}}, a_1)$, and the unfolding path follows $(a_0, b_0, \ldots, b_1, a_1)$.

Case 2: Pivot $p_B \notin A \cap B$ and the (forward, return) connecting paths for B do not overlap other connecting paths (except at their boundaries); we will later see that connecting paths may overlap. Let us settle some notation first (cf. Fig. 8a): r is the ray connecting B to A; k_1 and k_2 are forward and return connecting paths for B (one to either side of r); u_1 is the endpoint of r that lies on A; and u_2 is the other endpoint of the y-edge of A incident to u_1. We discuss three situations:

Case 2a: u_1 is neither a reflex corner nor a bottom corner of A. In this case, whenever the unfolding of A reaches k_1, the unfolding of B proceeds according to one of three subcases, depending on the position of p_B. If p_B touches a left gridface of B, the unfolding proceeds as in Fig. 8a, and if it touches a right gridface, the unfolding proceeds as in Fig. 8b. In both cases, b_0, the first gridface of B in counterclockwise order around p_B, is rotated so that the unfolding of B extends to the right, B is recursively unfolded, and the return path k_2 is rotated about p_B so that the unfolding of A continues to the right. The final subcase occurs when p_B touches only top gridfaces of B. Then the unfolding is identical to that in Fig. 8b but with b_s a top gridface.

Case 2b: u_1 is a reflex corner of A. In this case, the unfolding of B proceeds as in Fig. 9a,b. It is the existence of the vertical strip incident to u_1 (marked t in Fig. 9) that

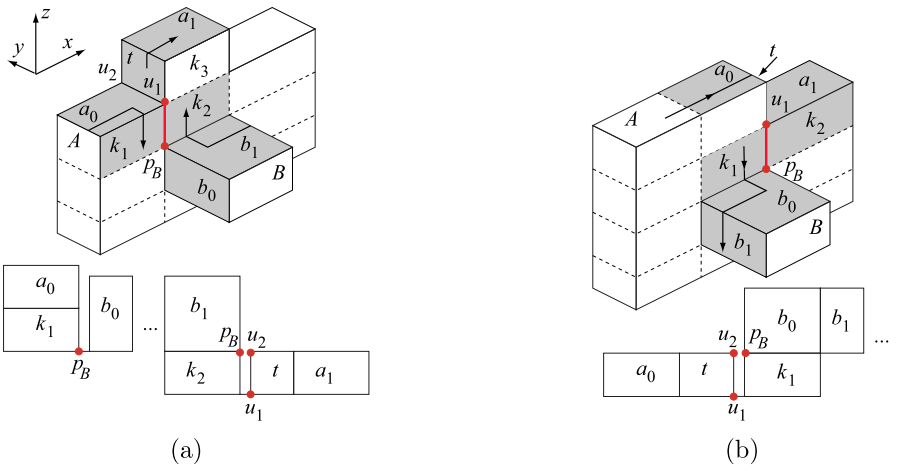

Fig. 9 Unfolding B: u_1 is a corner vertex of A. **a** t is a left strip **b** t is a right strip

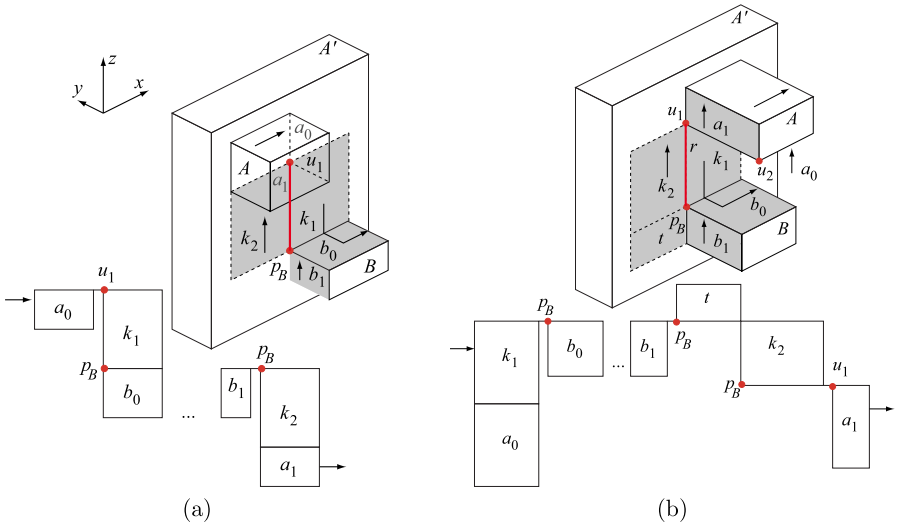

Fig. 10 Unfolding B: u_1 is a bottom corner of A **a** rightmost, and **b** leftmost gridface of A vertically aligned with leftmost gridface of B

makes handling this case different from Case 2a. Note, however, that the existence of t implies the existence of at least two gridfaces on either the return path or the forward path for B, depending on whether t is a left (Fig. 9a) or a right (Fig. 9b) strip of gridfaces. In the former case the unfolding starts as in Case 2a (Fig. 9a), and once the unfolding of B returns to p_B, it continues along the return path k_2 up to u_1, then unfolds t and orients it about u_1 so that the unfolding of A continues to the right. The gridface(s) that cover the gap above k_2 (marked k_3 in Fig. 9a) will be attached below the adjacent top gridface of A (a_1 in Fig. 9a) in the last phase of the unfolding algorithm (Sect. 6.4).

If t is a strip of right gridfaces, then we unfold t before descending along the forward path down to B, as in Fig. 9b (note the vertical symmetry with the unfolding in Fig. 9a); the unfolding of B then proceeds as in Case 2a (Fig. 8b).

Case 2c: u_1 is a bottom corner of A. In this case, the unfolding proceeds as in Fig. 10a or 10b, depending on whether u_1 is a right or a left bottom corner of A. The unfolding illustrated in Fig. 10a follows the familiar unfolding pattern: orient the first gridface of B in counterclockwise order around p_B so that the unfolding of B extends to the right; once the unfolding of B returns to p_B, follow the return path back to A and unfold the gridface of A clockwise to the right of u_1 (a_1 in Fig. 10a) so that the unfolding of A continues to the right. A similar pattern applies to the case illustrated in Fig. 10b, with one subtle difference meant to aid in unfolding front and back faces (discussed in Sect. 6.4): in unfolding bands, we aim at maintaining the vertical position of the (forward, return) connecting paths in the unfolding, so that vertical strips hanging below these connecting paths in 3D could also hang vertically in the 2D unfolding. More on this in Sect. 6.4. Observe that the z-vertical edges of k_1 and k_2 from Fig. 10a hang remain vertical in the unfolding. However, the z-vertical edges of k_2 from Fig. 10b must unfold as horizontal edges, otherwise it would not

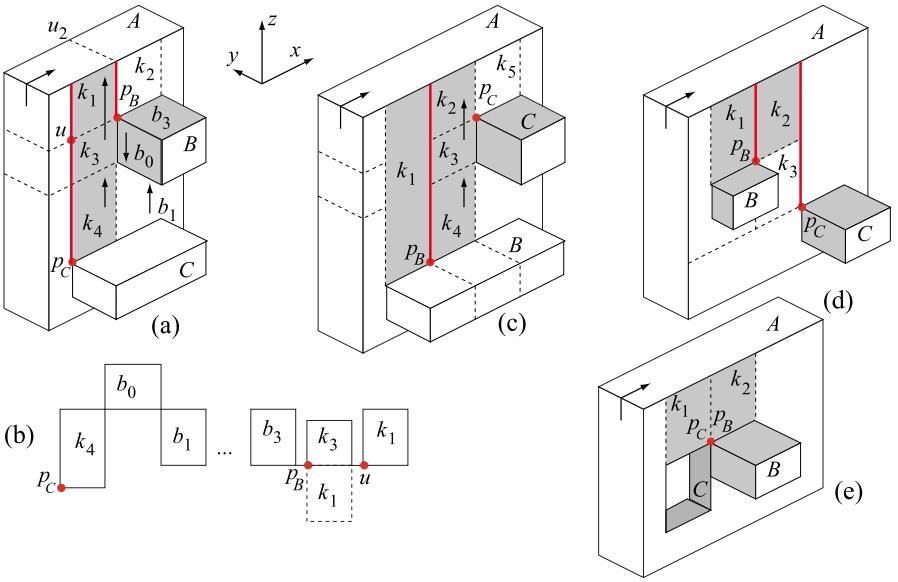

Fig. 11 **a** Return path for C includes k_4, k_3, k_1; forward path for B is k_1. **b** Unfolding for (**a**). **c** Return path for B includes k_5, k_4, k_2; forward path for C is k_2. **d** Return path for B is k_2; forward path for C includes k_2, k_3. **e** Forward (return) paths are identical for B and C

be possible to orient a_1 around u_1 so as to continue unfolding A to the right of k_2. This is the reason for employing the gridface strip marked t in the unfolding, so that z-vertical sides of t remain vertical in the unfolding, and any gridface strip hanging below t could be attached to t vertically in the unfolding.

We note that Fig. 10 illustrates only the situation in which p_B is incident to a left gridface of B, but it should not be difficult to observe that the same idea applies to any top pivot of B; the pivot position only affects the start and end unfolding position of B, and everything else remains the same.

Case 3: Pivot $p_B \notin A \cap B$ and a connecting path for B overlaps a connecting path for another descendant C of A. This case is slightly more complex, because it involves conflicts over the use of the connecting paths for B. The following three situations are possible.

Case 3a: The forward path k_1 for B overlaps the return path for another descendant C of A. This situation is illustrated in Fig. 11a. In this case, the unfolding of B starts as soon as the unfolding along the return path from C to A meets a gridface of B incident to p_B (gridface b_0 in Fig. 11a). At this point we recursively unfold B as before (see Fig. 11b), then the unfolding continues along the return path for C back to A. Figure 11b shows gridface k_1 in two positions: we let k_1 hang down only if the next gridface to unfold is a right gridface of a child of A (see also the transition from k_7 to c_5 in Fig. 12); otherwise, use k_1 in the upward position, a freedom permitted to us by rotating about vertex u.

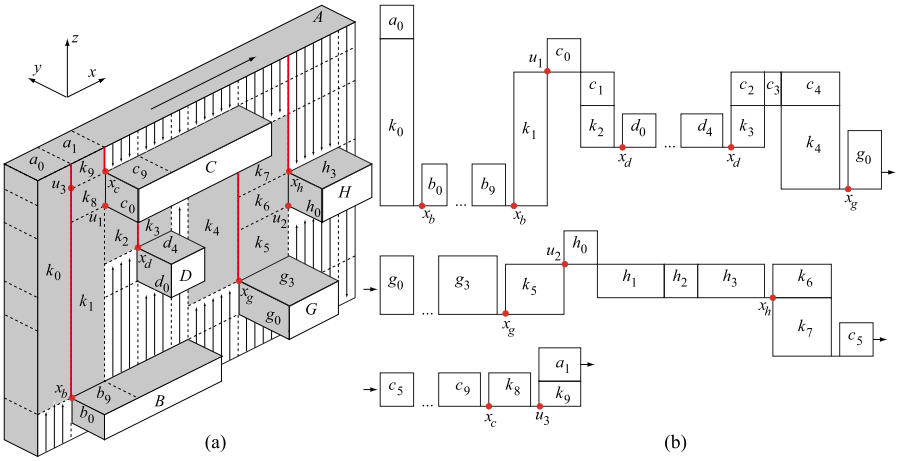

Fig. 12 **a** An example. **b** The vertex-unfolding

Case 3b: The return path for B overlaps the forward path for another descendant C of A. This situation is illustrated in Figs. 11c and 11d. The case depicted in Fig. 11c is similar to the one in Fig. 11a and is handled in the same manner. For the case depicted in Fig. 11d, notice that k_2 is on both the forward path for C and the return path for B. However, no conflict occurs here: from k_2 the unfolding continues downward along the forward path to C and unfolds C next.

Case 3c: The forward path k_1 for B overlaps the forward path for another descendant C of A. This situation occurs when either B or another band C incident to B is a dent, as illustrated in Fig. 11e. Again, no conflict occurs here: the recursive unfolding of C, which returns to $p_C = p_B$, is followed by the recursive unfolding of B, which returns to p_B, then the unfolding continues along the return path for B (C) back to A. We note that the forward paths for B and C overlap if and only if their reverse paths overlap, so this case also handles the situation in which the reverse paths overlap.

Figure 12 shows a more complex example that emphasizes these subtle unfolding issues. Note that the return path k_1, k_8, k_9 for B overlaps the forward path k_9 for C; and the return path k_5, k_6 and k_7 for G overlaps the forward path for H, which includes k_7. The unfolding produced by the method described in this section is depicted in Fig. 12b.

6.4 Attaching Front and Back Faces to the Net

Front and back faces of a slab are "hung" from bands following the basic idea of the illumination model discussed in Sect. 5.3. There are three differences, however, caused by the employment of some front and back gridfaces for the connecting paths, which can block illumination from the bands.

1. We illuminate both upward and downward from each band: each x-edge illuminates the vertical front/back face it attaches to. This alone already suffices to han-

dle the example in Fig. 12: all front and back faces are illuminated downward from the top of A, upward from the bottom of A, and upward from the top of B.

2. Some gridfaces still might not be illuminated by any bands, because they are obscured both above and below by paths in connecting gridfaces. Therefore we incorporate the connecting gridfaces into the band for the purposes of illumination. For example, in Fig. 10a, k_2 illuminates downward and k_1 illuminates upward. The reason this strategy works is that, with one exception, each vertical connecting strip remains vertical in the unfolding, and so illuminated strips can be hung safely without overlap. Note that although k_2 illuminates downward, it is rotated about p_B so that what was down in 3D becomes up in the unfolding. So the faces illuminated downward from k_2 get "hung upward."
3. The one exception is the return connecting path k_2 in Fig. 10b. This paths unfolds "on its side," i.e., what is vertical in 3D becomes horizontal in 2D. Note, however, that the gridface t below such a path (a gridface always present), is oriented vertically. We thus consider t to be part of the connecting path for illumination purposes, permitting the strip below to be hung under t.

Because our cases are exhaustive, all gridfaces of (say) the front face of A are either illuminated by A, or by some descendant of A on the front face, augmented by the connecting paths as just described. (In fact every gridface is illuminated twice, from above and below.) Hanging the strips then completes the unfolding.

6.5 Algorithm Complexity

Because there are so few unfolding algorithms, that there is *some* algorithm for a class of objects is more important than the speed of the algorithm. Nevertheless, we offer an analysis of the complexity of our algorithm. Let n be the number of corner vertices of the polyhedron, and $N = O(n^2)$ be the number of gridpoints. The vertex grid can be easily constructed in $O(N)$ time, leaving a planar surface map consisting of $O(N)$ gridpoints, gridedges, and gridfaces. The computation of connecting rays (Sect. 6.2) requires determining the components of $A \cap P_i^+$ and $A \cap P_i^-$, for each band A and incident plane Y_i. These can be easily read from the planar map by running through the n vertices of each of the $O(n)$ bands and determining, for each vertex, whether it belongs to P_i^+ or P_i^-. Each of the $O(n)$ band components shoots a vertical ray from one corner vertex, in a 2D environment (the plane Y_i) of n noncrossing orthogonal segments. Determining which band a ray hits involves a ray-shooting query. Although an implementation would employ an efficient data structure, perhaps BSP trees [9], for complexity purposes the naive $O(n)$ query cost suffices to lead to $O(n^2)$ time to construct G_r. Selecting pivots (Sect. 6.1) involves 2-coloring G_r in $O(n)$ time, and computing the unfolding tree T_u in a breadth-first traversal of G_r, which takes $O(n)$ time. Unfolding bands (Sect. 6.3) involves a depth-first traversal of T_u in $O(n)$ time, and laying out the $O(N)$ gridfaces in $O(N)$ time. Thus, the algorithm can be implemented to run in $O(N) = O(n^2)$ time.

7 Further Work

Extending these algorithms to arbitrary genus orthogonal polyhedra remains an interesting open problem. Holes that extend only in the x and z directions within a slab

seem unproblematic, as they simply disconnect the slab into several components. Holes that penetrate several slabs (i.e., extend in the y direction) present new challenges, as they may obstruct vertical band visibility necessary to establish that the band graph is connected. One idea to handle such holes is to place a virtual xz-face midway through the hole, and treat each half-hole as a dent (protrusion).

Acknowledgements We thank the anonymous referees for their careful reading and insightful comments.

Appendix: Proof of Lemma 2 (Connectedness of G_b)

For a band A, let $r_i(A)$ be the closed region of Y_i whose boundary is the rim of A, i.e., $A \cap Y_i$. Two subsets of $P_i = \partial O \cap Y_i \subset Y_i$ are *path-connected*, or just *connected*, if there are points in each that are connected by a path that lies in P_i. We first develop notation to describe the relevant portions of $r_i(A)$ that are connected to each band A. Recall from Sect. 2 that P_i^+ is composed of back faces and P_i^- of front faces.

We decompose the set of points in P_i into sets $c_i(A)$ for all bands A that meet P. The sets $c_i(A)$ will have disjoint interiors, overlapping only on their boundaries. Initially assign $c_i(A) = A \cap P_i$; we now augment these sets. Let p be an arbitrary point in P_i. We consider four cases, which ultimately reduce to a single case. First let p be on a front face, i.e., on P_i^-. Then p is either on a protrusion that lies behind Y_i (Fig. 13a), or on a dent in front of Y_i (Fig. 13b). Symmetric cases occur when p is on a back face (on P_i^+), either on a protrusion in front of Y_i (Fig. 13c), or a dent behind Y_i (Fig. 13d). Let p be on a front face of a band A (encompassing the first two cases). If p is path-connected to A, we add p to $c_i(A)$. Otherwise, p must be in $r_i(B)$ of a unique dent band B, which is itself in a protrusion B', both in front of Y_i. In this case, we add p to $c_i(B)$. For example, in Fig. 17a, p lies on the front face of A and is path-connected to A, and therefore $p \in c_i(A)$ (even though it is also path-connected to the surrounding dent B). In Fig. 18a, however, p lies on the front face of A' but is not path-connected to A', and therefore p is instead in the set $c_i(B)$ for the surrounding dent B. Figure 16b illustrates the symmetric case where p is on the back face of protrusion B', and because p is path-connected to B', $p \in c_i(B')$ (even though p is also path-connected to B).

The above definition of $c_i(A)$ ensures that $\cup c_i(A) = P_i$, where the union is over all A that meet P_i. Moreover the c_i sets have disjoint interiors. We now concentrate

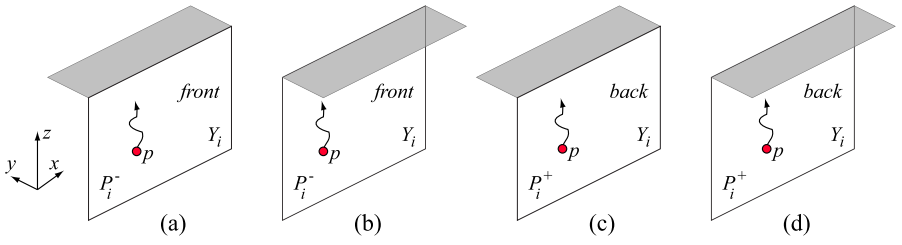

Fig. 13 Four cases: p is located on: **a** front face of protrusion behind Y_i, **b** front face of dent in front of Y_i, **c** back face of protrusion in front of Y_i, **d** back face of dent behind Y_i

on the boundaries of the c_i sets, and raise the observation we need to a lemma for later reference:

Lemma 5 *For protrusion A and dent B on opposite sides of Y_i such that $c_i(A) \cap c_i(B)$ is nonempty, it must be that $A \cap B$ is nonempty, i.e., the band rims share one or more points.*

Proof Suppose to the contrary that $A \cap B$ is empty. Then either $r_i(A)$ and $r_i(B)$ are disjoint, in which case $c_i(A) \cap c_i(B)$ is empty, a contradiction, or $r_i(A) \supset r_i(B)$, in which case B is a cavity in object O, violating our genus-zero assumption. □

This lemma justifies the following definition:

$$c_i(A, B) = \begin{cases} A \cap B, & \text{if } A \cap B \neq \emptyset, \text{ and at least one of } A \text{ and } B \text{ is a dent,} \\ c_i(A) \cap c_i(B) & \text{otherwise.} \end{cases}$$

This definition is intended to identify gridpoints on either A or B from which rays are issued by the ray-pair generation algorithm (Sect. 6.1.1). The reason for treating intersecting dents and protrusions differently is a subtle one, and is captured by Fig. 16b: B is a dent behind Y_i and B' is a protrusion in front of Y_i; $c_i(B')$ is the piece of the back face of B' enclosed by B; u is a highest gridpoint in $B \cap B'$, while w is a highest gridpoint in $c_i(B) \cap c_i(B')$; u is a potential ray basepoint, while w is not. The above definition eliminates points such as w from the set $c_i(A, B)$.

Our connectivity proof for G_b proceeds as follows. Let $P_i^1, P_i^2, \ldots$ denote the connected components of P_i, with $P_i = P_i^1 \cup P_i^2 \cup \cdots$. The bands incident to each of these are connected by rays (as discussed in Sect. 6.1.2) that lie in planes other than Y_i (see Fig. 14 for an example). We first argue that, to prove that G_b is ray-connected, it suffices to prove that each P_i^m is ray-connected. Remove from O all the slabs $S_1, S_2, \ldots$ incident to Y_0. Establish that the bands in the resulting object O' are ray-connected, via induction. The inductive hypothesis implies that the bands in each connected component of O' are ray-connected. Now put back the slabs. Each S_m corresponds to a component P_i^m. We will prove that all bands incident to P_i^m are ray-connected to one another. This along with the fact that O itself is connected implies that all bands are ray-connected. Henceforth we concentrate on one such connected component P_i^m, call it $Q \subset Y_i$ for succinctness. Let χ be the collection of all bands that intersect Q. Then $\bigcup_{A \in \chi} c_i(A) = Q$. The idea of the connectedness

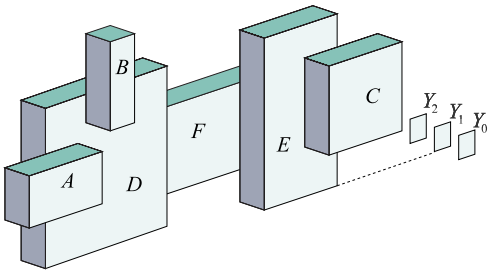

Fig. 14 P_1 contains two connected components, one incident to A, B, D, and one incident to C, E; pairs of bands incident to different components are connected by rays that lie in Y_2

proof is that the bands get connected in upward chains, and ultimately to each other through "common ancestor" higher bands. We choose to prove it by contradiction, arguing that a highest disconnected component cannot exist.

Lemma 6 *All bands in χ are ray-connected. Furthermore, if one arbitrary ray in each ray-pair is discarded, χ remains ray-connected.*

Proof For the purpose of contradiction, assume that not all bands in χ are ray-connected. Let $\chi_1, \chi_2, \ldots$ be the maximal subsets of χ that are ray-connected. Let $Q_j = \bigcup_{A \in \chi_j} c_i(A)$. Then $Q = \bigcup_j Q_j$. Since Q is connected, the subsets Q_j are not disjoint, in that for every Q_j there is an Q_k such that $Q_j \cap Q_k$ is nonempty. This along with Lemma 5 implies that

$$Q_{jk} = \bigcup_{A \in \chi_j} \bigcup_{B \in \chi_k} c_i(A, B)$$

is also nonempty. Let j and k be such that Q_{jk} contains a *highest* x-gridedge (gridpoint, if Q_{jk} contains only isolated points) among all Q_{jk}. Let u be the leftmost highest gridpoint in Q_{jk}. Let $A \in \chi_j$ and $B \in \chi_k$ be such that $u \in c_i(A, B)$.

We have thus identified two bands A and B, ray-disconnected because they lie in different components of Q, which contribute this highest gridpoint u in the "highest" intersection Q_{jk}. We now examine in turn the four protrusion/dent possibilities for these two bands.

Case 1. A and B are both protrusions on opposite sides of Y_i. Assume without loss of generality that A is behind Y_i, B is in front of Y_i, and u is on B (as depicted in Fig. 15). We discuss two subcases:

a. u is on a top edge of A or B; choose B without loss of generality (Figs. 15a, b). Then our ray-pair algorithm generates a ray-pair (r, r'), with r incident to u and r' incident to the gridpoint u' clockwise from u. Consider r (the analysis is similar for r'). If r hits A, then in fact A and B are ray-connected, contradicting the fact that A and B belong to different ray-connected components of χ. So let us assume

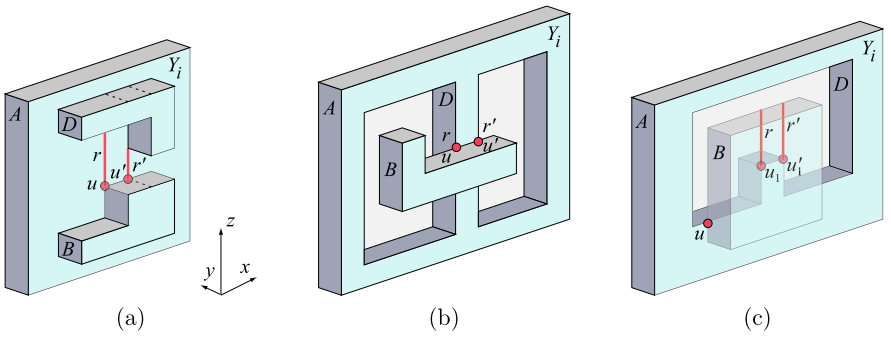

Fig. 15 Case 1: A and B are both protrusions on opposite sides of Y_i **a** D is a protrusion **b** D is a dent with a vertical side incident to u **c** D is a dent with a bottom edge incident to u

that r hits another band $D \in \chi_\ell$. Figure 15a, b illustrates the situation when D is a protrusion (dent). If $\ell = j$, then D and A are ray-connected in χ_j, and since B and D are ray-connected, it follows that B and A are ray-connected, a contradiction. On the other hand, if $\ell \neq j$, then $c_i(A, D)$ (and implicitly $Q_{j\ell}$) has a gridpoint higher than u, contradicting our choice of j, k and u.

b. u is not on a top edge of A or B, and so must be on a vertical (left, right) edge of A or B; again we choose B without loss of generality (Fig. 15c). Then u must be at the intersection between a dent D in protrusion A, and B. Because $A \cap D$ is empty, we fall into the second case of the definition of $c_i(A, D)$, which is therefore $c_i(A) \cap c_i(D)$. In this case, the same arguments as in Case a show that D and A are ray-connected, meaning that $D \in \chi_j$. Let u_1 be the leftmost among the highest gridpoints of $D \cap c_i(B)$. Then our ray-pair algorithm generates a ray-pair (r, r') from u_1 and its right neighbor u'_1. Consider r (the analysis is similar for r'). If r hits B, then B is ray-connected to D, which is ray-connected to A, a contradiction. If r hits a band E other than D, then it must be that $E \in \chi_k$, the same component containing B. Otherwise B and E would yield an intersection point higher than u, contradicting our choice of A and B. This means that B is ray-connected to E, which is ray-connected to D, which is ray-connected to A, a contradiction.

Case 2. A is a protrusion and B is a dent, both on a same side of Y_i. The case when A and B are both in front of Y_i (illustrated in Fig. 16a) is identical to Case 1 above, once one conceptually pops out B into a protrusion. We now discuss the case when A and B are both behind Y_i.

Assume first that $c_i(A, B)$ contains no top edges of B, as depicted in Fig. 16b. Let B' be a protrusion in front of Y_i covering the top of B. Then $c_i(A, B')$ and $c_i(B', B)$ each contains a gridpoint higher than u (see point $w' \in c_i(A, B')$ and $w \in c_i(B, B')$ in Fig. 16). The following two contradictory observations settle this case:

a. It must be that $B' \notin \chi_k$; otherwise Q_{jk} would contain a gridpoint in $c_i(A, B')$ higher than u.
b. If $B' \in \chi_\ell$, then it must be that $\ell = k$; otherwise $Q_{\ell k}$ would contain a gridpoint in $c_i(B', B)$ higher than u.

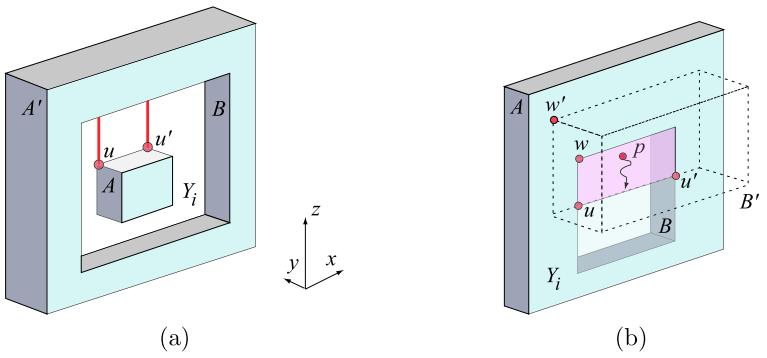

Fig. 16 Case 2: A is a protrusion and B is a dent **a** in front of Y_i **b** behind Y_i

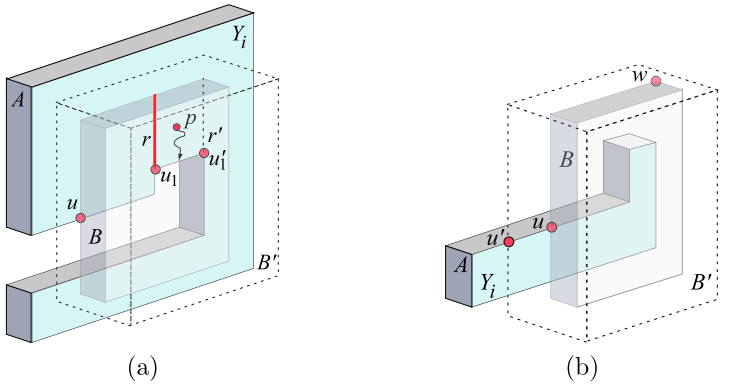

Fig. 17 Case 3: A is a protrusion behind Y_i; B is a dent in B', both in front of Y_i

If $c_i(A, B)$ contains at least one top gridedge of B, then arguments similar to the ones used for the case illustrated in Fig. 15a (conceptually popping B to become a protrusion) settle this case as well.

Case 3. A is a protrusion and B is a dent on opposite sides of Y_i (see Fig. 17). Let B' be the protrusion in front of Y_i enclosing B. We discuss three subcases:

a. $c_i(A)$ contains a top edge of B (see Fig. 17a). This means that $c_i(A) \cap r_i(B)$ is nonempty, and the ray-pair algorithm shoots a ray-pair (r, r') upward from the endpoints of a highest gridedge $\{u_1, u_1'\}$ of $A \cap r_i(B)$. Consider ray r (the analysis is similar for r'). If r hits B, then A and B are in fact ray-connected, a contradiction. If r hits a band D other than B, then arguments similar to the ones for the case illustrated in Fig. 15a (Case 1) lead to a contradiction.
b. $c_i(A)$ contains a bottom edge of B. This case is symmetrical to the one above in that a ray upward from a gridpoint of $B \cap r_i(A)$ hits A, thus ray-connecting A and B.
c. $c_i(A)$ contains neither a top nor a bottom edge of B (see Fig. 17b). Arguments similar to the ones used in Case 1 (protrusions on opposite sides of Y_i) show that A and B' are ray-connected. That B and B' are ray-connected follows immediately from the fact that $c_i(B, B')$ has a gridpoint higher than u (w in Fig. 17b). These together imply that A and B are ray-connected, a contradiction.

Case 4. A and B are both dents: A is a dent behind Y_i enclosed within protrusion A', and B is a dent in front of Y_i enclosed within protrusion B' (see Fig. 18). The genus-zero assumption implies that $r_i(A) \cap r_i(B)$ is a polygonal region of positive area. Since $u \in c_i(A) \cap c_i(B)$, we have that $u \in r_i(A) \cap r_i(B)$. Let β be the boundary segment of $r_i(A) \cap r_i(B)$ incident to u. We discuss two subcases:

a. $\beta \subset P_i^-$, meaning that $\beta \subset A$ (see Fig. 18a). An analysis similar to the one for the case illustrated in Fig. 17a (Case 3) shows that A and B are ray-connected, a contradiction.

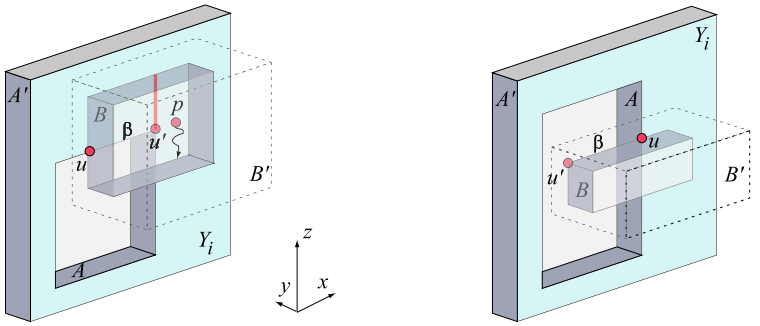

Fig. 18 Case 4: A is a dent behind Y_i, enclosed within protrusion A'. B is a dent in front of Y_i, enclosed within protrusion B'

b. $\beta \subset P_i^+$, meaning that $\beta \subset B$ (see Fig. 18b). We show that A and A' are ray-connected, B and B' are ray-connected, and A' and B' are ray-connected. This implies that A and B are ray-connected, a contradiction. First note that the ray-pair algorithm shoots a ray-pair (r, r') upward from a highest gridedge on β. An analysis similar to the one for the case illustrated in Fig. 15a (conceptually popping B to become a protrusion) shows that r and r' must hit B', thus ray-connecting B and B'. That A and A' are ray-connected follows immediately from the fact that $c_i(A, A')$ has a gridpoint higher than u, and similarly for A' and B'.

Having exhausted all possible cases, the connectivity claim of the lemma is established. Because the proof for each of these cases goes through by considering either the first or second ray of a ray-pair, retaining either ray suffices to preserve connectivity. Thus the second claim of the lemma is established as well. □

References

1. Biedl, T., Demaine, E., Demaine, M., Lubiw, A., O'Rourke, J., Overmars, M., Robbins, S., Whitesides, S.: Unfolding some classes of orthogonal polyhedra. In: Proc. 10th Canad. Conf. Comput. Geom., pp. 70–71, 1998
2. Damian, M., Flatland, R., O'Rourke, J.: Grid vertex-unfolding orthogonal polyhedra. In: 23rd Symp. Theoretical Aspects Comput. Sci., 2006. Lecture Notes Comput. Sci., vol. 3884, pp. 264–276. Springer, Berlin (2006)
3. Demaine, E.D., O'Rourke, J.: A survey of folding and unfolding in computational geometry. In: Goodman, J.E., Pach, J., Welzl, E. (eds.) Combinatorial and Computational Geometry. Cambridge University Press, Cambridge (2005)
4. Demaine, E.D., O'Rourke, J.: Open problems from CCCG 2004. In: Proc. 17th Canad. Conf. on Comput. Geom., pp. 303–306, 2005
5. Demaine, E.D., O'Rourke, J.: Geometric Folding Algorithms: Linkages, Origami, Polyhedra. Cambridge University Press, Cambridge (2007). http://www.gfalop.org
6. Demaine, E.D., Eppstein, D., Erickson, J., Hart, G.W., O'Rourke, J.: Vertex-unfoldings of simplicial manifolds. In: Bezdek, A. (ed.) Discrete Geometry, pp. 215–228. Dekker, New York (2003)
7. Demaine, E.D., Iacono, J., Langerman, S.: Grid vertex-unfolding of orthostacks. In: Japan Conf. Discrete Comput. Geom. 2004. Lecture Notes Comput. Sci., vol. 3742, pp. 76–82. Springer, Berlin (2005). Int. J. Comput. Geom. Appl. (to appear)

8. O'Rourke, J.: Folding and unfolding in computational geometry. In: Discrete Comput. Geom., Japan Conf. Discrete Comput. Geom., 1998. Lecture Notes Comput. Sci., vol. 1763, pp. 258–266. Springer, Berlin (2000).
9. Paterson, M.S., Yao, F.F.: Optimal binary space partitions for orthogonal objects. J. Algorithms **13**, 99–113 (1992)
10. Schwartz, E.L., Shaw, A., Wolfson, E.: A numerical solution to the generalized map-maker's problem: flattening nonconvex polyhedral surfaces. IEEE Trans. Pattern Anal. Mach. Intell. **11**(9), 1005–1008 (1989)
11. Tarini, M., Hormann, K., Cignoni, P., Montani, C.: Polycube-maps. ACM Trans. Graph. **23**(3), 853–860 (2004)
12. Wang, C.-H.: Manufacturability-driven decomposition of sheet metal products. PhD thesis, Carnegie Mellon University, The Robotics Institute (1997)