Thai Journal of Mathematics
Volume 21 Number 4 (2023)
Pages 1011–1023

http://thaijmath.in.cmu.ac.th

ISSN 1686-0209

Discrete and Computational Geometry, Graphs, and Games

Unfolding Orthotubes with a Dual Hamiltonian Path

Erik D. Demaine[1] **and Kritkorn Karntikoon**[2]

[1] *Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, MA 02139, USA*
e-mail : edemaine@mit.edu

[2] *Department of Computer Science, Princeton University, Princeton, NJ 08544, USA*
e-mail : kritkorn@princeton.edu

Abstract An ***orthotube*** consists of orthogonal boxes (e.g., unit cubes) glued face-to-face to form a path. In 1998, Biedl et al. showed that every orthotube has a ***grid unfolding***: a cutting along edges of the boxes so that the surface unfolds into a connected planar shape without overlap. We give a new algorithmic grid unfolding of orthotubes with the additional property that the rectangular faces are attached in a single path — a Hamiltonian path on the rectangular faces of the orthotube surface.

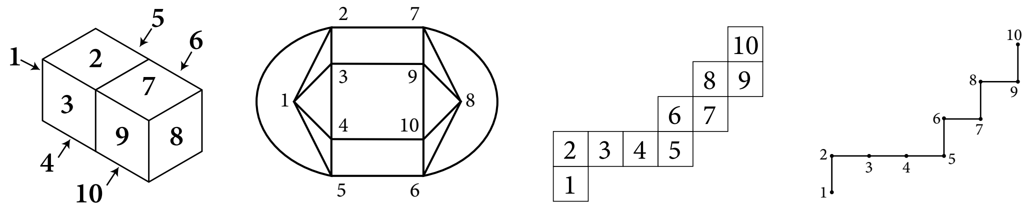
MSC: 68Q25; 52C99
Keywords: unfolding polyhedra; orthotubes; Hamiltonicity; algorithm

Submission date: 27.01.2022 / Acceptance date: 20.08.2022

1. Introduction

Does every orthogonal polyhedron have a ***grid unfolding***, that is, a cutting along edges of the induced grid (extending a plane through every face of the polyhedron) such that the remaining surface unfolds into a connected planar shape without overlap? This question remains unsolved over 20 years after this type of unfolding was introduced in 1998 [1]; see [2] for a survey and [3–5] for recent progress. This problem is in some sense the orthogonal nonconvex version of the older and more famous open problem of whether every convex polyhedron has an edge unfolding (cutting only along edges of the polyhedron) [6]. Many subclasses of orthogonal polyhedra have been successfully unfolded, though sometimes cutting along more than just grid edges [2–5, 7–14].

The first class of orthogonal polyhedra shown to have a grid unfolding is ***orthotubes*** [1], formed by gluing together a sequence of orthogonal boxes where every pair of consecutive boxes in the sequence share one face (and no other boxes share faces). Roughly speaking, this unfolding consists of a monotone dual-path of rectangular faces, with $O(1)$ rectangles attached above and below the path.



(A) orthotube (B) face adjacency graph (C) dual-Hamiltonian unfolding (D) dual-Hamiltonian path

FIGURE 1. An example of an orthotube, its face adjacency graph, dual-Hamiltonian unfolding corresponding to chain code $RSSLRLRL$, and the corresponding dual Hamiltonian path.

In this paper, we show that orthotubes have a grid unfolding with a stronger property we call ***dual-Hamiltonicity***, where the unfolded shape consists of a single dual path of rectangular faces, as shown in Figure 1. More precisely, define the ***face adjacency graph*** to have a node for each rectangular face of a box, and connect two nodes by an edge whenever the corresponding rectangular faces share an edge. Then the unfolding is given by keeping attached a chain of rectangular faces corresponding to a Hamiltonian path in the face adjacency graph. Implicitly, we take advantage of the fact that 4-connected planar graphs (which includes face adjacency graphs) have Hamiltonian cycles [15, 16]. This fact has also been studied previously in the context of vertex unfolding [17] and grid unfolding as *zipper unfolding* [18, 19].

The paper is organized as follows. First, Section 2 defines useful tools for expressing dual-Hamiltonian unfoldings. Then Section 3 describes and proves correct our algorithm to find such an unfolding for a given orthotube. Finally, in Section 4, we describe possible future extensions to our result.

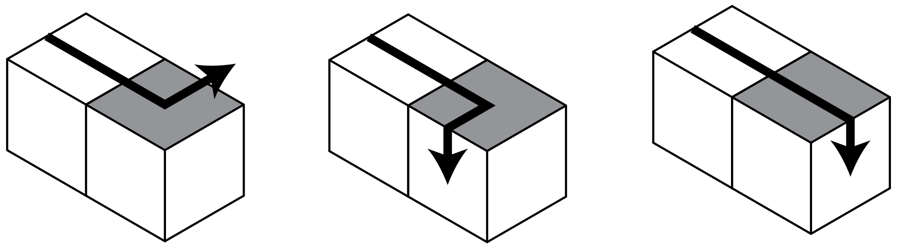
2. Chain Codes for Dual Paths

As mentioned in Section 1, our unfolding keeps attached a chain of rectangular faces corresponding to a Hamiltonian path in the face adjacency graph. In this section, we introduce a tool for describing such dual-Hamiltonian paths called "chain codes" (similar to [10, 20–22]):

Definition 2.1. A ***chain code*** is an ordered sequence of the form $a_1 a_2 \cdots a_n$, where $a_i \in \{L, R, S\}$ represents an (intrinsic) left turn, right turn, or continuing straight to move from the ith face to the $(i+1)$st face.

Given a starting face f_0 and initial intrinsic direction on the surface (equivalently, the next face f_1 visited), a chain code $a_1 a_2 \cdots a_n$ defines a ***corresponding path*** $f_0, f_1, f_2, \ldots, f_{n+1}$ in the face adjacency graph; see Figure 2.

If we unfold an orthotube to keep attached the path of faces corresponding to a chain code, then we can construct the 2D geometry of the unfolding by following the intrinsic directions as follows:



(A) Turning left (L) (B) Turning right (R) (C) Going straight (S)

FIGURE 2. The three possible changes of direction for a dual path on the surface of an orthotube (forbidding U-turns). The turn occurs within the gray box, changing the entering direction from the previous box face to the exiting direction to next box face.

Definition 2.2. The ***unfolding dual*** of a chain code $a_1a_2\cdots a_n$ is the orthogonal equilateral path $p_0, p_1, \ldots, p_n$ in the xy plane that starts with the segment from $p_0 = (0,0)$ to $p_1 = (0,1)$ and where the ith vertex p_i turns left, turns right, or goes straight according to $a_i \in \{L, R, S\}$. The ***corresponding unfolding*** has a unit square centered at each vertex p_i, where the ith and $(i+1)$st squares are attached along the 90° rotation of the segment p_ip_{i+1}.

For example, if the ith segment of the unfolding dual was from $(x, y-1)$ to (x, y), then the next vertex $i+1$ in the unfolding dual is $(x-1, y)$ if $a_i = L$; $(x+1, y)$ if $a_i = R$; and $(x, y+1)$ if $a_i = S$. Figure 1d shows another example when the chain code is $RSSLRLRL$. We can characterize nonoverlap as follows:

Lemma 2.3. *If the unfolding dual of a chain code has no overlapping vertices in the xy plane, then the corresponding unfolding has no overlap.*

Proof. By the correspondence between points p_i of the unfolding dual and unit squares in the corresponding unfolding, ∎

To guarantee that our unfolding does not overlap, we prove the invariant that the unfolding dual proceeds monotonically in the $+y$ direction. We can measure this property more easily using the following notion:

Definition 2.4. For each chain code $U = a_1a_2\cdots a_n$, the ***cumulative quarter turning*** $\text{qturn}(U)$ is $\sum_{i=1}^{n} \text{qturn}(a_i)$ where $\text{qturn}(R) = +1$, $\text{qturn}(L) = -1$, and $\text{qturn}(S) = 0$.

Our desired invariant of the unfolding dual proceeding monotonically in the $+y$ direction is thus equivalent to requiring that qturn of any prefix of the chain code is in $\{-1, 0, +1\}$.

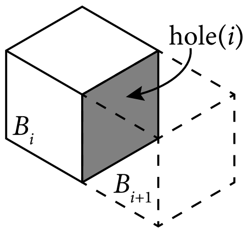
Lemma 2.5. *If a chain code U satisfies* $\text{qturn}(P) \in \{-1, 0, +1\}$ *for every prefix P of U, then its corresponding unfolding has no overlap.*

Proof. Divide U into segments $S_1S_2\cdots S_k$ where every prefix $P_j = S_1S_2\cdots S_j$ of segments has $\text{qturn}(P_j) = 0$, and no other prefix P of U has $\text{qturn}(P) = 0$. Assume by induction that the unfolding dual of P_{j-1} has no repeated points, and that the last point is strictly above all other points. (In the base case, $j-1=0$ and P_{j-1} is empty, so the hypothesis holds for the two points p_0, p_1.) If the first symbol of P_j is S, then we immediately enter a new row and segment, so the inductive hypothesis holds. If the first symbol of P_j is R, then qturn becomes $+1$, so the remaining symbols of P_j must be S followed by a final L, at which point qturn becomes 0. The corresponding points in the unfolding dual form a rightward row (above all other points), and the next point starts a new row, establishing the inductive hypothesis in this case. If the first symbol of P_j is L, then a symmetric argument holds. Therefore the unfolding dual has no repeated points, so by Lemma 2.3, the corresponding unfolding does not overlap. ■

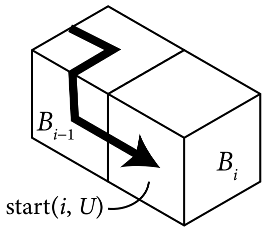
3. Algorithm

In this section, we present an algorithm to unfold an orthotube along a dual Hamiltonian path, by constructing a chain code for that path (Definition 2.1). The algorithm builds the chain code incrementally, progressively unfolding each box in the orthotube, while maintaining the invariant (mentioned in Section 2) that the unfolding's chain code so far has qturn $\in \{-1, 0, +1\}$. By Lemma 2.5, this invariant implies that the unfolding does not overlap. In fact, to simplify the induction, the algorithm creates a chain code for the unfolding of possibly several boxes at a time to maintain the stronger condition that the intermediate chain codes have qturn $= 0$. We divide into cases based on each box's relative position to the next one, two, and sometimes three or four boxes (if they exist).

More formally, suppose the given orthotube consists of boxes $B_0, B_1, \ldots, B_n$ in order. For some $k \geq 0$, we construct a chain code U_k whose corresponding unfolding is a nonoverlapping unfolding of the suborthotube $B_0, B_1, \ldots, B_k$. This chain code is thus a Hamiltonian path on the faces of boxes $B_0, B_1, \ldots, B_k$ that are actually faces of the orthotube — that is, excluding the shared face **hole($\boldsymbol{i}$)** between B_i and B_{i+1} for $0 \leq i < n$ (see Figure 3a). Note in particular that the suborthotube $B_0, B_1, \ldots, B_k$ is not a normal orthotube (for $k < n$) because B_k is missing one face, hole(k). In the chain code U_k we include a turn code for the last face of B_k visited (for $k < n$), and require that this turn would bring the path into a face of B_{k+1}; we call this requirement ***continuability***. We construct U_k inductively, using a previous U_{k-x} with qturn$(U_{k-x}) = 0$ to construct U_k with qturn$(U_k) = 0$. In the final step, we construct U_n (which may not have qturn$(U_n) = 0$), which is the chain code for an unfolding of the entire orthotube.



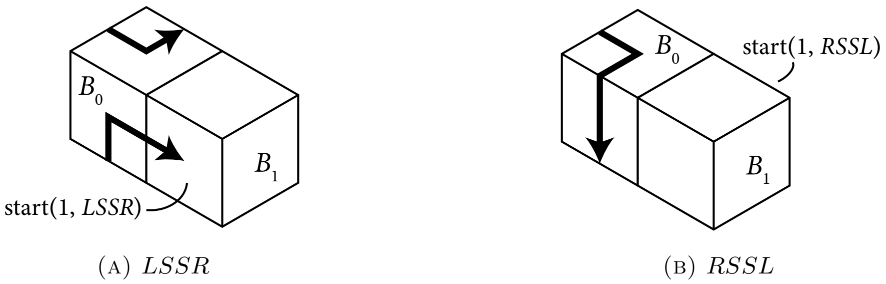
(A) Face hole(i) is the shared face of B_i and B_{i+1}, which is not a face of the orthotube.



(B) Face start(i, U) is the first face of B_i encountered by the unfolding chain code U (drawn as an arrow path).

FIGURE 3. Definitions of the faces hole(i) and start(i, U).

For the base case, we construct either $U_0 = LSSR$ or $U_0 = RSSL$; see Figure 4. In either case, $\mathrm{qturn}(U_0) = 0$. By starting this chain code at the face of B_0 opposite the shared face hole(0) with B_1, it successfully visits all faces of B_0 except the shared face, and the last turn code attempts to enter a face of B_1. Even fixing the initial direction for the chain code, at least one of the two choices for U_0 will be continuable, successfully entering a face of B_1 and not B_2.

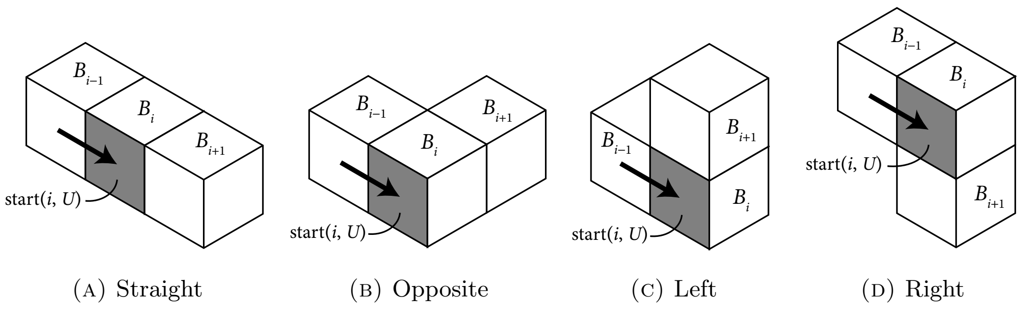


(A) *LSSR* (B) *RSSL*

FIGURE 4. Two possible ways to construct U_0 starting at the face of B_0 opposite to hole(0).

For the inductive step, we are given a continuable chain code U_{i-1} with $\mathrm{qturn}(U_{i-1}) = 0$. By continuability, the last turn code of U_{i-1} enters a face of B_i, which we call $\mathbf{start}(\boldsymbol{i}, \boldsymbol{U_{i-1}})$; see Figure 3b. Now we unfold B_i based on the relative position of the next box B_{i+1}, in order to guarantee that U_i is continuable into B_{i+1}. There are four cases for the relative position, which we denote $\boldsymbol{N(i, U_{i-1})}$, as follows; refer to Figure 5.

- $N(i, U) = S$ if $\mathrm{start}(i, U)$ can be directed to hole(i) by going straight, as in Figure 5a.
- $N(i, U) = O$ if $\mathrm{start}(i, U)$ is on the opposite side of hole(i), as in Figure 5b.
- $N(i, U) = L$ if $\mathrm{start}(i, U)$ can be directed to hole(i) by turning left, as in Figure 5c.
- $N(i, U) = R$ if $\mathrm{start}(i, U)$ can be directed to hole(i) by turning right, as in Figure 5d.



(A) Straight (B) Opposite (C) Left (D) Right

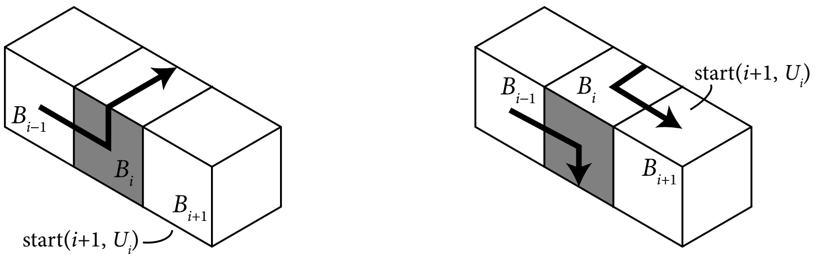
FIGURE 5. Four possible configurations of the next box B_{i+1} (equivalently, hole(i)) relative to box B_i and face $\mathrm{start}(i, U)$. Each configuration gives a different value for $N(i, U)$.

The chain code needed to unfold box B_i in each case is different. For example, if $N(i,U) = S$ as shown in Figure 5a, then we might unfold by $RSSL$ or $LSSR$; while, if $N(i,U) = O$ as shown in Figure 5b, then we might unfold by $RLSR$ or $LRSL$ instead. But only some of these chain codes may be continuable, depending on the relative configuration $N(i+2,U)$ of box B_{i+2}.

In the remainder of this section, we provide a chain code to append to U_{i-1} that has qturn $= 0$, determined by $N(i, U_{i-1})$ and its continuability. In some cases, such as when $N(i, U_{i-1}) = S$, we can find a continuable chain code to unfold the box B_i, extend the code U_{i-1} to U_i with $\text{qturn}(U_i) = 0$, and continue to the induction step $i+1$. However, in most cases, the continuable chain code involves unfolding multiple boxes at a time in order to restore qturn to 0, resulting in skipped step(s). We also need to ensure that the provided chain code visits every face and the qturn of any prefix of the chain code is in $\{-1, 0, +1\}$. These facts are easy to check and will be omitted here.

Case 1. $N(i, U_{i-1}) = S$.

There are two ways to unfold B_i, $LSSR$ and $RSSL$; see Figure 6. Because both chain codes have qturn $= 0$, then no matter which of these chain codes we append, we still have $\text{qturn}(U_i) = 0$. Because $LSSR$ and $RSSL$ lead to different faces, at least one of them makes U_i continuable.



(A) $LSSR$ (B) $RSSL$

FIGURE 6. Two possible ways to unfold B_i when $N(i,U) = S$. The gray face is $\text{start}(i, U_{i-1})$.

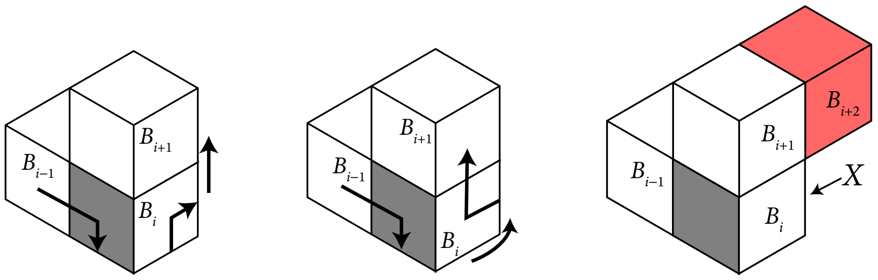
Case 2. $N(i, U_{i-1}) = L$.

We consider two ways to unfold B_i, $RLRL$ and $RSLR$; see Figure 7a and 7b. We prefer adding $RLRL$ because its qturn is 0. Thus, if $U_{i-1} + RLRL$ is continuable, then we assign $U_i = U_{i-1} + RLRL$, and this satisfies our invariants.

In the subcase when $U_{i-1} + RLRL$ is not continuable, we need to unfold B_i with $RSLR$, i.e., set $U_i = U_{i-1} + RSLR$. Thus $\text{qturn}(U_i) = 1$, so we need to unfold the next box in order to restore qturn to 0.

Because $U_{i-1} + RLRL$ is not continuable, the face X of box B_i opposite $\text{start}(i, U_{i-1})$ (as shown in Figure 7c) must not be adjacent to box B_{i+1}, so it must be adjacent to box B_{i+2}. Thus, after unfolding B_i with $RSLR$, we have $N(i+1, U_i) = R$. We consider unfolding B_{i+1} with either $LRLR$ or $LSRL$ (the reflections of the unfoldings we considered for B_i, to keep qturn $\in \{-1, 0, +1\}$), but now we prefer $LSRL$ because it restores $\text{qturn}(U_{i+1})$ to 0.

We use + to denote concatenation of chain codes.



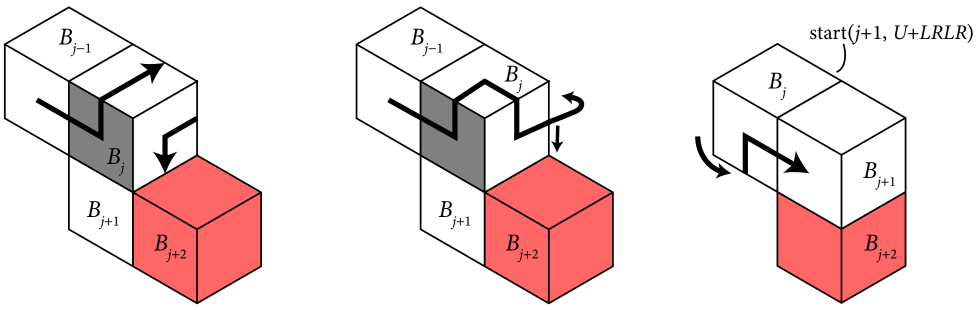
(A) *RLRL* (B) *RSLR* (C) The position of B_{i+2} (red box) when pattern *RLRL* is not continuable.

Figure 7. Two possible ways to unfold B_i when $N(i, U) = L$, and an analysis of the latter case.

If $U_i + LSRL$ is continuable, then we assign $U_{i+1} = U_i + LSRL$ and satisfy the invariants. Otherwise, we set $U_{i+1} = U_i + LRLR$ and need to continue unfolding, as $\mathrm{qturn}(U_{i+1}) = 1$ still. The following lemma guarantees that we remain in the same relative configuration in this subsubcase:

Lemma 3.1. *If we have an integer j and chain code U such that $N(j, U) = R$ and $U + LSRL$ is not continuable, then $N(j + 1, U + LRLR) = R$.*

Proof. Refer to Figure 8b, which in particular shows B_j and B_{j+1} with $N(j, U) = R$. Given that $U + LSRL$ is not continuable, following $LSRL$ in B_j does not lead us to the box B_{j+1}, so that face of B_{j+1} must be hole$(j + 1)$. Focusing on boxes B_j to B_{j+2}, as in Figure 8c, we see that $N(j + 1, U + LRLR) = R$ as desired. ∎



(A) A configuration from B_{j-1} to B_{j+2} where $N(j, U) = R$ and $U + LSRL$ is not continuable. (B) $U + LRLR$ is continuable. (C) The same configuration, focusing on B_j to B_{j+2}.

Figure 8. Illustration of the proof of Lemma 3.1.

Because we remain in the same relative configuration by Lemma 3.1, we can repeat the *LRLR* unfolding until we reach the first positive integer k such that $U_{i-1+k} + LSRL$

is continuable. Then we set

$$U_{i+k} = U_{i-1} + RSLR + (k-1) \times LRLR + LSRL$$

(where $\times$ denotes repetition of a chain code). Thus qturn$(U_{i+k}) = 0$ as desired, and we satisfy the invariants.

Case 3. $N(i, U_{i-1}) = R$.

We generate a chain code in the same way as in Case 2 where $N(i, U_{i-1}) = L$, but swapping the roles of L and R in both U and N (effectively reflecting all diagrams).

Case 4. $N(i, U_{i-1}) = O$.

There are two ways to unfold B_i, $RLSR$ and $LRSL$, with respective qturn values of $+1$ and -1. Hence, again, we need to unfold the next box in order to restore qturn to 0. We do so exhaustively for all possible relative positions of B_{i+2}, as shown in Figure 9.

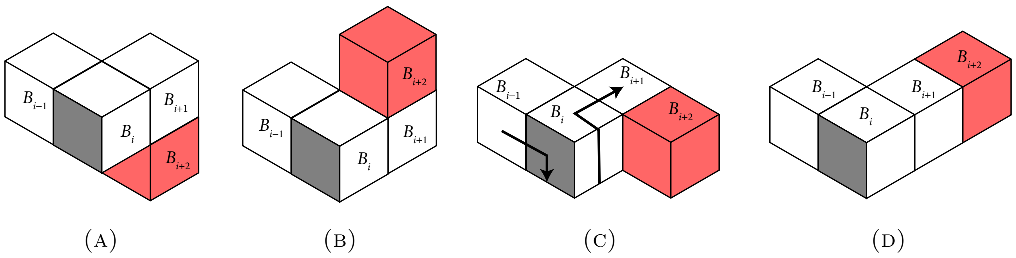
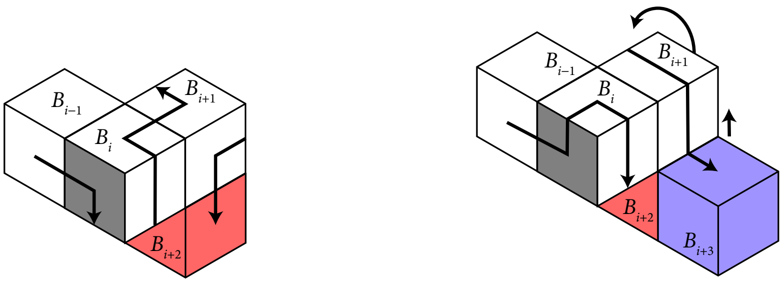


FIGURE 9. Four possible positions of B_{i+2} (red box) when $N(i, U) = O$.

Case 4.1. $U_{i-1} + LRSL$ is not continuable, as shown in Figure 9a.

In this subcase, we need to further consider the position of B_{i+1} (if it exists). Observe that $U_{i-1} + RLSR$ is continuable. If $U_{i-1} + RLSRLRSL$ is continuable, then we assign $U_{i+1} = U_{i-1} + RLSRLRSL$, which restores qturn to 0.

On the other hand, if $U_{i-1} + RLSRLRSL$ is not continuable, then the position of B_{i+3} is as shown in Figure 10. We then assign $U_{i+2} = U_{i-1} + LRSLRLRSRLSS$ which restores qturn to 0.



(A) $RLSRLRSL$

(B) The extension of the configuration in Figure 9a when $U_{i-1} + RLSRLRSL$ is not continuable. Box B_{i+2} is red and B_{i+3} is blue.

FIGURE 10. Two ways to generate the chain code for the configuration in Figure 9a.

Case 4.2. $U_{i-1} + RLSR$ is not continuable, as shown in Figure 9b.

We generate a chain code as in the previous subcase by swapping the roles of L and R (effectively reflecting all diagrams).

Case 4.3. B_{i+2} is "coplanar" with B_{i-1}, B_i, B_{i+1}, as shown in Figure 9c.

Because $U_{i-1} + RLSR$ is continuable, we can assign $U_i = U_{i-1} + RLSR$. Then we have $N(i+1, U_i) = R$ and $\operatorname{qturn}(U_i) = +1$. If $U_i + LSRL$ is continuable, then we assign

$$U_{i+1} = U_i + LSRL = U_{i-1} + RLSRLSRL,$$

which restores qturn to 0.

On the other hand, if $U_i + LSRL$ is not continuable, then we can use Lemma 3.1 to repeatedly unfold boxes with $LRLR$ until we reach a box where $LSRL$ is continuable, similar to Case 2. Thus with the same method we obtain a positive integer k and chain code U_{i+k} with $\operatorname{qturn}(U_{i+k}) = 0$.

Case 4.4. B_{i+2} and B_i are on opposite sides of B_{i+1}, as shown in Figure 9d.

In this subcase, we exhaustively examine the position of B_{i+3} and provide the extended chain code accordingly. There are five possible relative positions for B_{i+3} as follows; refer to Figure 11.

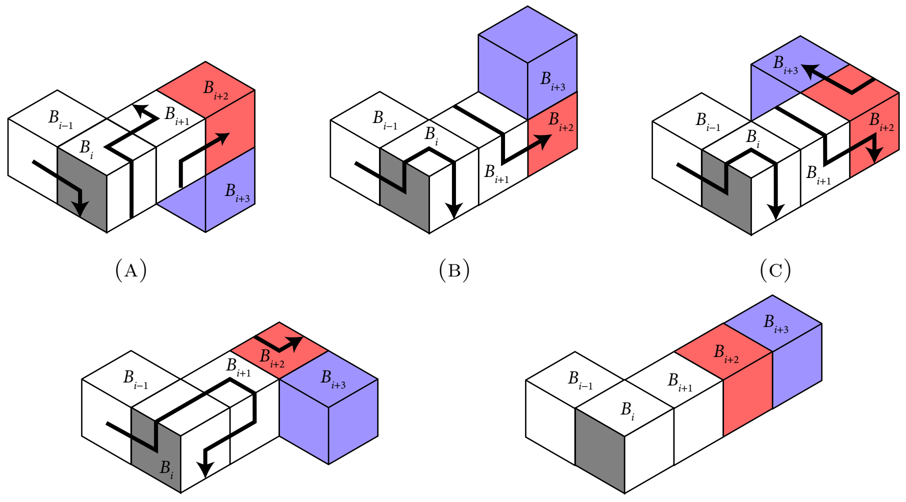


FIGURE 11. Five possible extensions of configuration shown in Figure 9d. Box B_{i+2} is red and B_{i+3} is blue.

(a) Shown in Figure 11a. We assign $U_{i+1} = U_{i-1} + RLSRLSSR$. Hence, we have $\operatorname{qturn}(U_{i+1}) = +1$ and $N(i+2, U_{i+1}) = R$. We can follow the same process as in Case 4.3 to get a chain code for some U_{i+k} with $\operatorname{qturn}(U_{i+k}) = 0$.

(b) Shown in Figure 11b. We use the same method as the previous case but swap L and R.

(c) Shown in Figure 11c. If $U_{i-1} + LRSLRSSLRLSR$ is continuable, we assign that to U_{i+2}. Otherwise, we have the configuration shown in Figure 12, and we assign $U_{i+2} = U_{i-1} + RLSRLSSRLRSL$ instead.

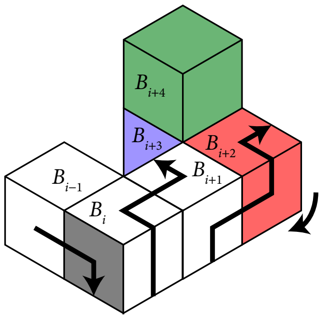


FIGURE 12. The extension of configuration in Figure 11c when $U_{i-1} + LRSLRSSLRLSR$ is not continuable. Box B_{i+4} is green.

(d) Shown in Figure 11d. If $U_{i-1} + LSRRLLRLRLSR$ is continuable, then we assign that to U_{i+2}. Otherwise, reflecting the extension, $U_{i-1} + RSLLRRLRLRSL$ must be continuable, and we assign that to U_{i+2}. In either case, we satisfy the invariants.

(e) In the last case, we have four boxes $B_i, B_{i+1}, B_{i+2}, B_{i+3}$ in a straight line; refer to Figure 11e.

Observe that as long as the boxes remain in a straight line, unfolding each box will preserve the qturn value because the only possible ways to unfold are $LSSR$ and $RSSL$. Hence, to restore qturn to 0, we need to skip until we find the first box which is not lined up. Let $k \geq 4$ be the smallest integer such that B_{i+k} is not lined up with B_i, B_{i+1}, B_{i+2}.

The idea is to unfold $B_i, B_{i+1}, \ldots$ until B_{i+k-3} or B_{i+k-2}, and then apply the same methods used previously in Case 4 to generate chain code for the remaining boxes. The essence of why this works is that the chain code generated in Case 4 (except the forth subcase of Case 4.4) always fully unfolds box B_i before start unfolding B_{i+1}. Hence, if we can simulate the same $\text{qturn}(U_i)$ and $\text{start}(i+1, U_i)$, then we can treat B_{i+k} as B_{i+2} in Cases 4.1–4.3 or as B_{i+3} in Case 4.4 and follow the same method.

We claim that, for any $j > 0$, it is possible to unfold boxes B_i to B_{i+2j} in such a way that $\text{qturn}(U_{i+2j})$ can be either $+1$ or -1 and $\text{start}(i + 2j + 1, U_{i+2j})$ can be either top or bottom of the figure when looking from the perspective shown in Figure 13. For $j = 1$, the claim is true from $\{U_{i-1} + RLSRLSSRLSSR, U_{i-1} + RSLLRRLRLSSR, U_{i-1} + LRSLRSSLRSSL, U_{i-1} + LSRRLLRLRSSL\}$, as shown in Figure 13. To extend the result to any j, we need to unfold every two consecutive boxes with $RSSLRSSL$ if $\text{qturn}(U_{i+1}) = -1$ and $LSSRLSSR$ if $\text{qturn}(U_{i+1}) = +1$. This proves the claim.

Next, we provide an algorithm to generate the chain code to unfold $B_i, \ldots, B_{i+k}$.

For even k, we first find the Case 4.1–4.3 where their B_i, B_{i+1} and B_{i+2} are rearranged in the same way as boxes B_{i+k-2}, B_{i+k-1}, and B_{i+k}. (We also consider the upside-down version of Case 4.3, so that it covers the case where B_{i+k} is "coplanar" with B_{i+k-1}, B_{i+k-2} and B_{i+k-3} but has a different configuration from Case 4.3.) Then, we generate the chain code according to the claim so we get the desired $\text{qturn}(U_{i+k-2})$ and $\text{start}(i + k - 1, U_{i+k-2})$.

For odd k, we follow a similar process as the even case, but with B_{i+k-3} instead of B_{i+k-2}, and follow the same method for first three subcases of Case 4.4. For the fourth case where the configuration is aligned with Figure 11d, we use the flipped chain code of Figure 11c instead because it fully unfolds B_i first.

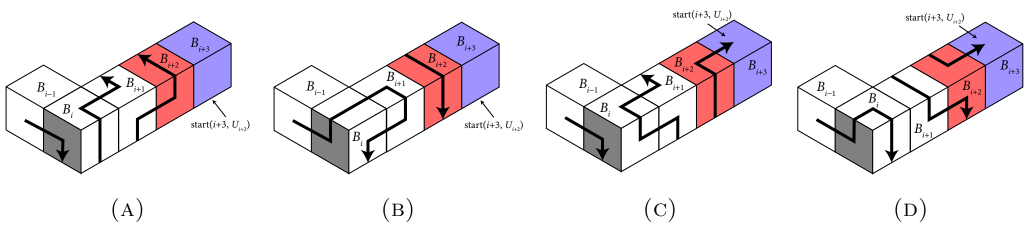


Figure 13. Four ways to unfold B_i to B_{i+2} so we can have desired start$(i+3, U_{i+2})$ and qturn(U_{i+2}).

Combining these cases, we have provided a chain code for every possible position of B_{i+1} (and sometimes $B_{i+2}, B_{i+3}, \dots$). We ensure that the invariant for the induction is true: the qturn value of the chain code is always restored to 0 (except when we reach B_n first).

To construct the final unfolding U_n, we can temporarily create a new box B_{n+1} attached to B_n. By induction, we can find a chain code U_n to unfold $B_0, B_1, \dots, B_n$ and end at start$(n+1, U_n)$. Then when we remove B_{n+1}, the chain code U_n will end at hole(n) instead, which results in an unfolding of the original orthotube. Because the qturn of any prefix of the chain code is in $\{-1, 0, +1\}$, by Lemma 2.5, the corresponding unfolding of U_n does not overlap (and is dual-Hamiltonian) as desired.

4. Conclusion

In this paper, we proposed a new algorithm to unfold orthotubes such that the unfolding path is a Hamiltonian path of the orthotube's face adjacency graph. In other words, we can unfold an orthotube by traveling through all faces on the orthotube's surface without having to visit the same face twice.

An intriguing harder goal is to find a Hamiltonian *cycle* through the face adjacency graph, such that breaking that cycle into a path results in an unfolding without overlap. This would require the unfolding to effectively traverse the length of the orthotube twice (down and back up). Potentially, such an approach could be more amenable for extension to unfolding ***orthotrees*** (boxes glued to form a tree instead of a path), by recursing on subtrees and combining the cycles together.

References

[1] T. Biedl, E. Demaine, M. Demaine, A. Lubiw, M. Overmars, J. O'Rourke, S. Robbins, and S. Whitesides, "Unfolding some classes of orthogonal polyhedra," in *Proceedings of the 10th Canadian Conference on Computational Geometry*, pp. 70–71, 1998.

[2] J. O'Rourke, "Unfolding orthogonal polyhedra," in *Surveys on Discrete and Computational Geometry: Twenty Years Later* (J. E. Goodman, J. Pach, and R. Pollack, eds.), pp. 231–255, American Mathematical Society, 2008.

[3] M. Damian, E. Demaine, R. Flatland, and J. O'Rourke, "Unfolding genus-2 orthogonal polyhedra with linear refinement," *Graphs and Combinatorics*, vol. 33, no. 5, pp. 1357–1379, 2017.

[4] M. Damian and R. Y. Flatland, "Unfolding low-degree orthotrees with constant refinement," in *Proceedings of the 30th Canadian Conference on Computational Geometry*, (Winnipeg, Canada), pp. 189–208, 2018.

[5] M. Damian and R. Flatland, "Unfolding polycube trees with constant refinement," *Computational Geometry: Theory and Applications*, vol. 98, p. 101793, 2021.

[6] E. D. Demaine and J. O'Rourke, *Geometric Folding Algorithms: Linkages, Origami, Polyhedra*. Cambridge University Press, 2007.

[7] M. W. Bern, E. D. Demaine, D. Eppstein, E. H. Kuo, A. Mantler, and J. Snoeyink, "Ununfoldable polyhedra with convex faces," *Computational Geometry: Theory and Applications*, vol. 24, pp. 51–62, 2003.

[8] M. Damian and H. Meijer, "Edge-unfolding orthostacks with orthogonally convex slabs," in *Abstracts from the 14th Annual Fall Workshop on Computational Geometry*, pp. 20–21, 2004.

[9] M. Damian, R. Flatland, H. Meijer, and J. O'Rourke, "Unfolding well-separated orthotrees," in *Abstracts from the 15th Annual Fall Workshop Computational Geometry*, pp. 23–25, 2005.

[10] M. Damian, R. Flatland, and J. O'Rourke, "Epsilon-unfolding orthogonal polyhedra," *Graphs and Combinatorics*, vol. 23, no. 1, pp. 179–194, 2007.

[11] M. Damian, E. D. Demaine, and R. Flatland, "Unfolding orthogonal polyhedra with quadratic refinement: the delta-unfolding algorithm," *Graphs and Combinatorics*, vol. 30, no. 1, pp. 125–140, 2014.

[12] M.-H. Liou, S.-H. Poon, and Y.-J. Wei, "On edge-unfolding one-layer lattice polyhedra with cubic holes," in *Proceedings of the 20th International Conference on Computing and Combinatorics*, vol. 8591 of *Lecture Notes in Computer Science*, (Atlanta), August 2014.

[13] Y. Chang and H. Yen, "Unfolding orthogonal polyhedra with linear refinement," in *Proceedings of the 26th International Symposium on Algorithms and Computation* (K. M. Elbassioni and K. Makino, eds.), vol. 9472 of *Lecture Notes in Computer Science*, (Nagoya, Japan), pp. 415–425, December 2015.

[14] K.-Y. Ho, Y.-J. Chang, and H.-C. Yen, "Unfolding some classes of orthogonal polyhedra of arbitrary genus," *Journal of Combinatorial Optimization*, vol. 37, pp. 482–500, February 2019.

[15] C. Thomassen, "A theorem on paths in planar graphs," *Journal of Graph Theory*, vol. 7, no. 2, pp. 169–176, 1983.

[16] N. Chiba and T. Nishizeki, "The Hamiltonian cycle problem is linear-time solvable for 4-connected planar graphs," *Journal of Algorithms*, vol. 10, no. 2, pp. 187–211, 1989.

[17] L. Garcia, A. Gutierrez, I. Ruiz, and A. Winslow, "Vertex unfoldings of orthogonal polyhedra: Positive, negative, and inconclusive results.," in *CCCG*, pp. 217–222, 2018.

[18] A. Lubiw, E. D. Demaine, M. L. Demaine, A. Shallit, and J. Shallit, “Zipper unfoldings of polyhedral complexes,” in *Proceedings of the 22nd Annual Canadian Conference on Computational Geometry, Winnipeg, Manitoba, Canada, August 9-11, 2010*, pp. 219–222, 2010.

[19] E. D. Demaine, M. L. Demaine, and R. Uehara, “Zipper unfoldability of domes and prismoids,” in *Proceedings of the 25th Canadian Conference on Computational Geometry, CCCG 2013, Waterloo, Ontario, Canada, August 8-10, 2013*, Carleton University, Ottawa, Canada, 2013.

[20] Z. Abel and E. D. Demaine, “Edge-unfolding orthogonal polyhedra is strongly NP-complete,” August 2011.

[21] E. Lemus, E. Bribiesca, and E. Garduo, “Representation of enclosing surfaces from simple voxelized objects by means of a chain code,” *Pattern Recognition*, vol. 47, no. 4, pp. 1721–1730, 2014.

[22] E. Lemus, E. Bribiesca, and E. Garduo, “Surface trees representation of boundary surfaces using a tree descriptor,” *Journal of Visual Communication and Image Representation*, vol. 31, pp. 101–111, 2015.