Universal Hinge Patterns for Folding Strips Efficiently into Any Grid Polyhedron

Nadia M. Benbernou[1] *, Erik D. Demaine[2],
Martin L. Demaine[2], and Anna Lubiw[3]

[1] Google Inc., nbenbern@gmail.com.
[2] MIT Computer Science and Artificial Intelligence Laboratory,
32 Vassar St., Cambridge, MA 02139, USA, {edemaine,mdemaine}@mit.edu
[3] David R. Cheriton School of Computer Science, University of Waterloo,
Waterloo, Ontario N2L 3G1, Canada, alubiw@uwaterloo.ca

Abstract. We present two universal hinge patterns that enable a strip of material to fold into any connected surface made up of unit squares on the 3D cube grid—for example, the surface of any polycube. The folding is efficient: for target surfaces topologically equivalent to a sphere, the strip needs to have only twice the target surface area, and the folding stacks at most two layers of material anywhere. These geometric results offer a new way to build programmable matter that is substantially more efficient than what is possible with a square $N \times N$ sheet of material, which can fold into all polycubes only of surface area $O(N)$ and may stack $\Theta(N^2)$ layers at one point. We also show how our strip foldings can be executed by a rigid motion without collisions (albeit assuming zero thickness), which is not possible in general with 2D sheet folding.

To achieve these results, we develop new approximation algorithms for milling the surface of a grid polyhedron, which simultaneously give a 2-approximation in tour length and an 8/3-approximation in the number of turns. Both length and turns consume area when folding a strip, so we build on past approximation algorithms for these two objectives from 2D milling.

1 Introduction

In *computational origami design*, the goal is generally to develop an algorithm that, given a desired shape or property, produces a crease pattern that folds into an origami with that shape or property. Examples include folding any shape [9], folding approximately any shape while being watertight [10], and optimally folding a shape whose projection is a desired metric tree [14,15]. In all of these results, every different shape or tree results in a completely different crease pattern; two shapes rarely share many (or even any) creases.

The idea of a *universal hinge pattern* [6] is that a finite set of *hinges* (possible creases) suffice to make exponentially many different shapes. The main result along these lines is that an $N \times N$ "box-pleat" grid suffices to make any polycube

* Work performed while at MIT.

made of $O(N)$ cubes [6]. The *box-pleat grid* is a square grid plus alternating diagonals in the squares, also known as the "tetrakis tiling". For each target polycube, a subset of the hinges in the grid serve as the crease pattern for that shape. Polycubes form a *universal* set of shapes in that they can arbitrarily closely approximate (in the Hausdorff sense) any desired volume.

The motivation for universal hinge patterns is the implementation of *programmable matter*—material whose shape can be externally programmed. One approach to programmable matter, developed by an MIT–Harvard collaboration, is a *self-folding sheet*—a sheet of material that can fold itself into several different origami designs, without manipulation by a human origamist [12,1]. For practicality, the sheet must consist of a fixed pattern of hinges, each with an embedded actuator that can be programmed to fold or not. Thus for the programmable matter to be able to form a universal set of shapes, we need a universal hinge pattern.

The box-pleated polycube result [6], however, has some practical limitations that prevent direct application to programmable matter. Specifically, using a sheet of area $\Theta(N^2)$ to fold N cubes means that all but a $\Theta(1/N)$ fraction of the surface area is wasted. Unfortunately, this reduction in surface area is necessary for a roughly square sheet, as folding a $1\times1\times N$ tube requires a sheet of diameter $\Omega(N)$. Furthermore, a polycube made from N cubes can have surface area as low as $\Theta(N^{2/3})$, resulting in further wastage of surface area in the worst case. Given the factor-$\Omega(N)$ reduction in surface area, an average of $\Omega(N)$ layers of material come together on the polycube surface. Indeed, the current approach can have up to $\Theta(N^2)$ layers coming together at a single point [6]. Real-world robotic materials have significant thickness, given the embedded actuation and electronics, meaning that only a few overlapping layers are really practical [12].

Our results: strip folding. In this paper, we introduce two new universal hinge patterns that avoid these inefficiencies, by using sheets of material that are long only in one dimension ("strips"). Specifically, Fig. 1 shows the two hinge patterns: the *canonical strip* is a $1 \times N$ strip with hinges at integer grid lines and same-oriented diagonals, while the *zig-zag strip* is an N-square zig-zag with hinges at just integer grid lines. We show in Section 2 that any *grid surface*—any connected surface made up of unit squares on the 3D cube grid—can be folded from either strip. The strip length only needs to be a constant factor larger than the surface area, and the number of layers is at most a constant throughout the folding. Most of our analysis concerns (genus-0) *grid polyhedra*, that is, when the surface is topologically equivalent to a sphere (a manifold without boundary, so that every edge is incident to exactly two grid squares, and without handles, unlike a torus). We show in Section 4 that a grid polyhedron of

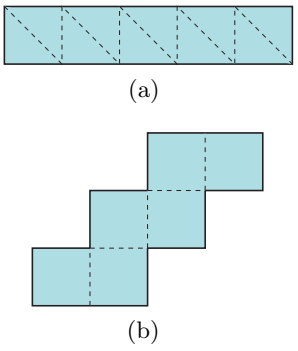


Fig. 1. Two universal hinge patterns in strips. (a) A canonical strip of length 5. (b) A zig-zag strip of length 6. The dashed lines are hinges.

surface area N can be folded from a canonical strip of length $2N$ with at most two layers everywhere, or from a zig-zag strip of length $4N$ with at most four layers everywhere.

The improved surface efficiency and reduced layering of these strip results seem more practical for programmable matter. In addition, the panels of either strip (the facets delineated by hinges) are connected acyclically into a path, making them potentially easier to control. One potential drawback is that the reduced connectivity makes for a flimsier device; this issue can be mitigated by adding tabs to the edges of the strips to make full two-dimensional contacts across seams and thereby increase strength.

We also show in Section 5 an important practical result for our strip foldings: under a small assumption about feature size, we give an algorithm for actually folding the strip into the desired shape, while keeping the panels rigid (flat) and avoiding self-intersection throughout the motion. Such a rigid folding process is important given current fabrication materials, which put flexibility only in the creases between panels [12]. An important limitation, however, is that we assume zero thickness of the material, which would need to be avoided before this method becomes practical.

Our approach is also related to the 1D chain robots of [7], but based on thin material instead of thick solid chains. Most notably, working with thin material enables us to use a few overlapping layers to make any desired surface without scaling, and still with high efficiency. Essentially, folding long thin strips of sheet material is like a fusion between 1D chains of [7] and the square sheet folding of [6,12,1].

Milling tours. At the core of our efficient strip foldings are efficient approximation algorithms for *milling* a grid polyhedron. Motivated by rapid-fabrication CNC milling/cutting tools, milling problems are typically stated in terms of a 2D region called a "pocket" and a cutting tool called a "cutter", with the goal being to find a path or tour for the cutter that covers the entire pocket. In our situation, the "pocket" is the surface of the grid polyhedron, and the "cutter" is a unit square constrained to move from one grid square of the surface to an (intrinsically) adjacent grid square.

The typical goals in milling problems are to minimize the length of the tour [3] or to minimize the number of turns in the tour [2]. Both versions are known to be strongly NP-hard, even when the pocket is an integral orthogonal polygon and the cutter is a unit square. We conjecture that the problem remains strongly NP-hard when the pocket is a grid polyhedron, but this is not obvious.

In our situation, both length and number of turns are important, as both influence the required length of a strip to cover the surface. Thus we develop one algorithm that simultaneously approximates both measures. Such results have also been achieved for 2D pockets [2]; our results are the first we know for surfaces in 3D. Specifically, we develop in Section 3 an approximation algorithm for computing a milling tour of a given grid polyhedron, achieving both a 2-approximation in length and an 8/3-approximation in number of turns.

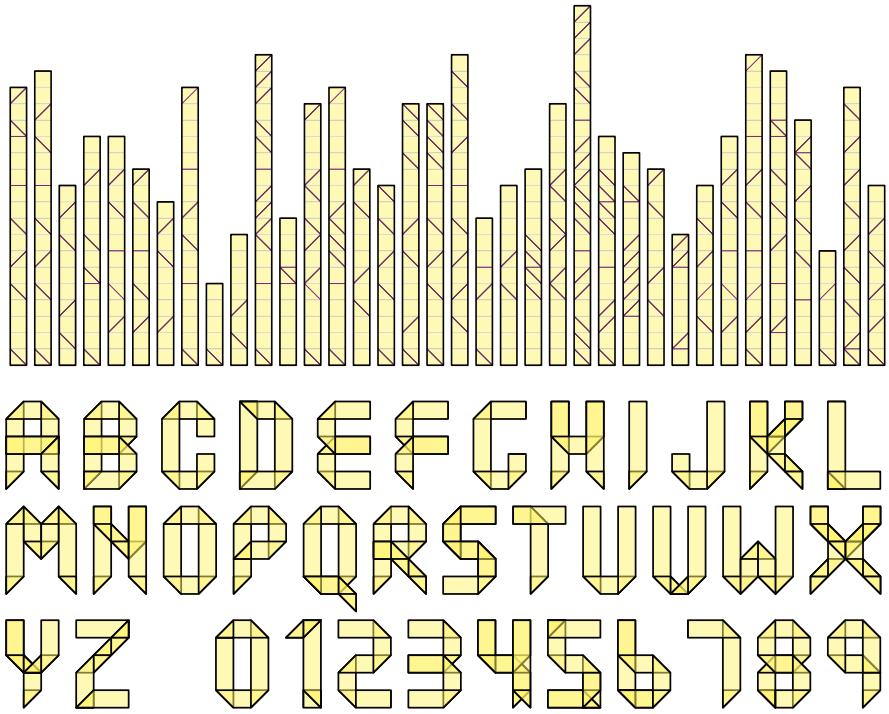


Fig. 2. Strip folding of individual letters typeface, A–Z and 0–9: unfolded font (top) and folded font (bottom), where the face incident to the bottom edge remains face-up.

Fonts. To illustrate the power of strip folding, we designed a typeface, representing each letter of the alphabet by a folding of a $1 \times x$ strip for some x, as shown in Fig. 2. The individual-letters typeface consists of two fonts: the unfolded font is a puzzle to figure out each letter, while the folded font is easy to read. These crease patterns adhere to an integer grid with orthogonal and/or diagonal creases, but are not necessarily subpatterns of the canonical hinge pattern. This extra flexibility gives us control to produce folded half-squares as desired, increasing the font's fidelity.

We have developed a web app that visualizes the font.[4] Currently in development is the ability to chain letters together into one long strip folding; Fig. 9 at the end of the paper shows one example.

The full version of this paper [5] contains details omitted from this extended abstract.

2 Universality

In this section, we prove that both the canonical strip and zig-zag strip of Fig. 1, of sufficient length, can fold into any grid surface. We begin with milling tours

[4] http://erikdemaine.org/fonts/strip/

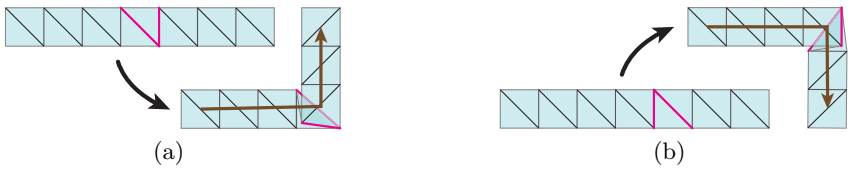


Fig. 3. (a) Left and (b) right turn with a canonical strip.

which provide an abstract plan for routing the strip, and then turn to the details of how to manipulate each type of strip.

Dual graph. Recall that a *grid surface* consists of one or more *grid squares*—that is, squares of the 3D cube grid—glued edge-to-edge to form a connected surface (ignoring vertex connections). Define the *dual graph* to have a dual vertex for each such grid square, and any two grid squares sharing an edge define a dual edge between the two corresponding dual vertices. Our assumption of the grid surface being connected is equivalent to the dual graph being connected.

Milling tours. A *milling tour* is a (not necessarily simple) spanning cycle in the dual graph, that is, a cycle that visits every dual vertex at least once (but possibly more than once). Equivalently, we can think of a milling tour as the path traced by the center of a moving square that must cover the entire surface while remaining on the surface, and return to its starting point. Milling tours always exist: for example, we can double a spanning tree of the dual graph to obtain a milling tour of length less than double the given surface area.

At each grid square, we can characterize a milling tour as going straight, turning, or U-turning—intrinsically on the surface—according to which two sides of the grid square the tour enters and exits. If the sides are opposite, the tour is *straight*; if the sides are incident, the tour *turns*; and if the sides are the same, the tour *U-turns*. Intuitively, we can imagine unfolding the surface and developing the tour into the plane, and measuring the resulting planar turn angle at the center of the grid square.

Strip folding. To prove universality, it suffices to show that a canonical strip or zig-zag strip can follow any milling tour and thus make any grid polyhedron. In particular, it suffices to show how the strip can go straight, turn left, turn right, and U-turn. Then, in 3D, the strip would be further folded at each traversed edge of the grid surface, to stay on the surface. Indeed, U-turns can be viewed as folding onto the opposite side of the same surface, and thus are intrinsically equivalent to going straight; hence we can focus on going straight and making left/right turns.

Canonical strip. Fig. 3 shows how a canonical strip can turn left or right; it goes straight without any folding. Each turn adds 1 to the length of the strip, and adds 2 layers to part of the grid square where the turn is made. Therefore a milling tour of length L with t turns of a grid surface can be followed by a canonical strip of length $L + t$. Furthermore, if the milling tour visits each grid square at most c times, then the strip folding has at most $3c$ layers covering any point of the surface.

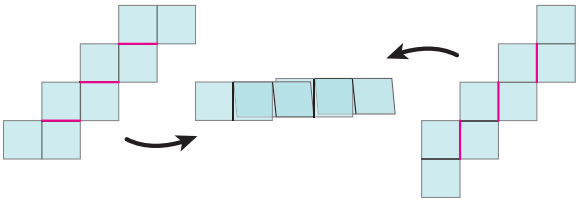


Fig. 4. Going straight with a zig-zag strip requires at most two unit squares per grid square. Left and right crease patterns show two different parities along the strip.

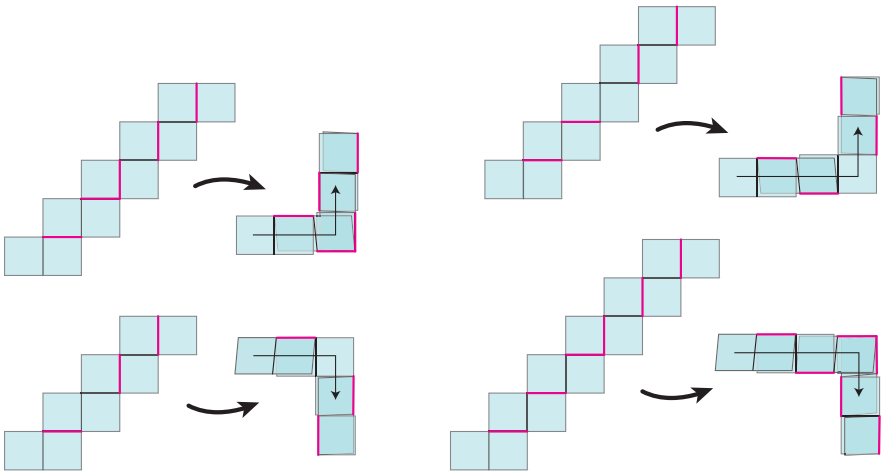


Fig. 5. Turning with a zig-zag strip has two cases because of parity. (a) Turning left at an odd position requires three grid squares, whereas turning right requires one grid square. (b) Turning left at an even position requires one grid square, whereas turning right requires three grid squares.

Zig-zag strip. Fig. 4 shows how to fold a zig-zag strip in order to go straight. In this straight portion, each square of the surface is covered by two squares of the strip. Fig. 5 shows left and right turns. Observe that turns require either one or three squares of the strip. Therefore a milling tour of length L with t turns can be followed by a zig-zag strip of length at most $2L + t$. Furthermore, if the milling tour visits each grid square at most c times, then the strip folding has at most $3c$ layers covering any point of the surface.

Proposition 1. *Every grid surface of area N can be folded from a canonical strip of length $4N$, with at most eight layers stacked anywhere, and from a zig-zag strip of length $6N$, with at most twelve layers stacked anywhere.*

The goal in the rest of this paper is to achieve better bounds for grid polyhedra, using more carefully chosen milling tours.

3 Milling Tour Approximation

This section presents a constant-factor approximation algorithm for milling a (genus-0) grid polyhedron P with respect to both length and turns. Specifically, our algorithm is a 2-approximation in length and an 8/3-approximation in turns. Our milling tours also have special properties that make them more amenable to strip folding.

Our approach is to reduce the milling problem to vertex cover in a tripartite graph. Then it follows that our algorithm is a 2α-approximation in turns, where α is the best approximation factor for vertex cover in tripartite graphs. The best known bounds on α are $34/33 \leq \alpha \leq 4/3$. Clementi et al. [8] proved that minimum vertex cover in tripartite graphs is not approximable within a factor smaller than $34/33 = 1.\overline{03}$ unless P = NP. Theorem 1 of [13] implies a 4/3-approximation for minimum weighted vertex cover for tripartite graphs (assuming we are given the 3-partition of the vertex set, which we know in our case). Thus we use $\alpha = 4/3$ below. An improved approximation ratio α would improve our approximation ratios, but may also affect the stated running times, which currently assume use of [13].

3.1 Bands

The basis for our approximation algorithms is the notion of "bands" for a grid polyhedron P. Let $x_{\min}$ and $x_{\max}$ respectively be the minimum and maximum x coordinates of P; define $y_{\min}, y_{\max}, z_{\min}, z_{\max}$ analogously. These minima and maxima have integer values because the vertices of P lie on the integer grid. Define the ith *x-slab* $S_x(i)$ to be the slab bounded by parallel planes $x = x_{\min}+i$ and $x = x_{\min}+i+1$, for each $i \in \{0, 1, \ldots, x_{\max} - x_{\min} - 1\}$. The intersection of P with the ith x-slab $S_x(i)$ (assuming i is in the specified range) is either a single *band* (i.e., a simple cycle of grid squares in that slab), or a collection of such bands, which we refer to as x-bands. Define y-bands and z-bands analogously.

Two bands *overlap* if there is a grid square contained in both bands. Each grid square of P is contained in precisely two bands (e.g., if a grid square's outward normal were in the $+z$-direction, then it would be contained in one x-band and one y-band). Two bands B_1 and B_2 are *adjacent* if they do not overlap, and a grid square of B_1 shares an edge with a grid square of B_2. A *band cover* for P is a collection of x-, y-, and z-bands that collectively cover the entire surface of P. The *size* of a band cover is the number of its bands.

3.2 Cover Bands

The starting point for the milling approximation algorithm is to find an approximately minimum band cover, as the minimum band cover is a lower bound on the number of turns in any milling tour:

Proposition 2. [2, Lemma 4.9] *The size of a minimum band cover of a grid polyhedron P is a lower bound on the number of turns in any milling tour of P.*

Next we describe how to find a near-optimal band cover. Consider the graph G_P with one vertex per band of a grid polyhedron P, connecting two vertices by an edge if their corresponding bands overlap. It turns out that an (approximately minimum) vertex cover in G_P will give us an (approximately minimum) band cover in P:

Proposition 3. *A vertex cover for G_P induces a band cover of the same size and vice versa.*

Because the bands fall into three classes (x-, y-, and z-), with no overlapping bands within a single class, G_P is tripartite. Hence we can use an α-approximation algorithm for vertex cover in tripartite graphs to find an α-approximate vertex cover in G_P and thus an α-approximate band cover of P.

3.3 Connected Bands

Our next goal will be to efficiently tour the bands in the cover. Given a band cover S for a grid polyhedron P, define the *band graph* G_S to be the subgraph of G_P induced by the subset of vertices corresponding to S. We will construct a tour of the bands S based on a spanning tree of G_S. Our first step is thus to show that G_S is connected (Lemma 5 below). We do so by showing that adjacent bands (as defined in Section 3.1) are in the same connected component of G_S, using the following lemma of Genc [11]:

Lemma 4. [11][5] *For any band B in a grid polyhedron P, let N_b be the bands of P overlapping B. (Equivalently, N_b is the set of neighbors of B in G_P). Then the subgraph of G_P induced by N_B is connected.*

Lemma 5. *If S is a band cover for a grid polyhedron P, then the graph G_S is connected.*

3.4 Band Tour

Now we can present our algorithm for transforming a band cover into an efficient milling tour.

Theorem 6. *Let P be a grid polyhedron with N grid squares. In $O(N^2 \log N)$ time, we can find a milling tour of P that is a 2-approximation in length and an $8/3$-approximation (or more generally, a 2α-approximation) in turns.*

We now state some additional properties of any milling tour produced by the approximation algorithm of Theorem 6, which will be useful for later applications to strip folding in Section 4.

[5] Genc [11] uses somewhat different terminology to state this lemma: "straight cycles in the dual graph" are our bands, and "crossing" is our overlapping. The induced subgraph is also defined directly, instead of as an induced subgraph of G_P.

Proposition 7. *Let P be a grid polyhedron, and consider a milling tour of P obtained from the approximation algorithm of Theorem 6. Then the following properties hold:*

1. *A grid square of P is either visited once, in which case it is visited by a straight part of the tour; or it is visited twice, by two straight junctions or by two turn junctions.*
2. *In the case of a turn junction, the length of the milling tour between the two visits to the grid square (counting only one of the two visits to the grid square in the length measurement) is even.*
3. *The tour can be modified to alternate between left and right turns (without changing its length or the number of turns).*

3.5 Polynomial Time

The algorithm described above is polynomial in the surface area N of the grid polyhedron, or equivalently, polynomial in the number of unit cubes making up the polyomino solid. For our application to strip folding, this is polynomial in the length of the strip, and thus sufficient for most purposes. On the other hand, polyhedra are often encoded as a collection of n vertices and faces, with faces possibly much larger than a unit square. In this case, the algorithm runs in *pseudopolynomial* time.

Although we do not detail the approach here, our algorithm can be modified to run in polynomial time. To achieve this result, we can no longer afford to deal with unit bands directly, because their number is polynomially related to the number N of grid squares, not the number n of vertices. To achieve polynomial time in n, we concisely encode the output milling tour using "fat" bands rather than unit bands, which can then be easily decoded into a tour of unit bands. By making each band as wide as possible, their number is polynomially related to n instead of N. Details of an $O(n^3 \log n)$-time milling approximation algorithm (with the same approximation bounds as above) can be found in [4].

4 Strip Foldings of Grid Polyhedra

In this section, we show how we can use the milling tours from Section 3 to fold a canonical strip or zig-zag strip efficiently into a given (genus-0) grid polyhedron. For both strip types, define the *length* of a strip to be the number of grid squares it contains; refer to Fig. 1. For a strip folding of a polyhedron P, define the *number of layers covering a point q* on P to be the number of interior points of the strip that map to q in the folding, excluding *crease points*, that is, points lying on a hinge that gets folded by a nonzero angle. (This definition may undercount the number of layers along one-dimensional creases, but counts correctly at the remaining two-dimensional subset of P.) We will give bounds on the length of the strip and also on the maximum number of layers of the strip covering any point of the polyhedron.

The main idea for canonical strips is that Properties (1) and (3) of Proposition 7 allow us to make turns as shown in Fig. 6, so that we do not waste an extra square of the strip per turn.

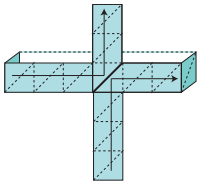


Fig. 6. A turn junction for a canonical strip.

Theorem 8. *Let P be a grid polyhedron, and let N be the number of grid squares of P. Then P can be covered by a folding of a canonical strip of length $2N$, and with at most two layers covering any point of P.*

For zig-zag strips, we instead use Properties (1) and (2) of Proposition 7:

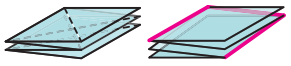
Theorem 9. *Let P be a grid polyhedron, and let N be the number of grid squares of P. Then P can be covered by a folding of a zig-zag strip of length $4N$, and with at most four layers covering any point of P.*

By coloring the two sides of the zig-zag strip differently, we can also bicolor the surface of P in any pattern we wish, as long as each grid square is assigned a uniform color. We do not prove this result formally here, but mention that the bounds in length would become somewhat worse, and the rigid motions presented in Section 5 do not work in this setting.

5 Rigid Motion Avoiding Collision

So far we have focused on just specifying a final folded state for the strip, and ignored the issue of how to continuously move the strip into that folded state. In this section, we show how to achieve this using a *rigid folding motion*, that is, a continuous motion that keeps all polygonal faces between hinges rigid/planar, bending only at the hinges, and avoids collisions throughout the motion. Rigid folding motions are important for applications to real-world programmable matter made out of stiff material except at the hinges. Our approach may still suffer from practical issues, as it requires a large (temporary) accumulation of many layers in an accordion form.

We prove that, if the grid polyhedron P has feature size at least 2, then we can construct a rigid motion of the strip folding without collision. (By *feature size at least* 2, we mean that every exterior voxel of P is contained in some empty $2 \times 2 \times 2$ box.) If the grid polyhedron we wish to fold has feature size 1, then one solution is to scale the polyhedron by a factor of 2, and then the results here apply.



(a) (b)

Fig. 7. Accordion for (a) canonical strip and (b) zig-zag strip, with hinges drawn thick for increased visibility.

Our approach is to keep the yet-unused portion of the strip folded up into an *accordion* and then to unfold only what is needed for the current move: straight, left, or right. Fig. 7 shows the accordion state for the canonical strip and for the zig-zag strip. We will perform the strip folding in such a way that the strip never penetrates the interior of the polyhedron P, and it never weaves under

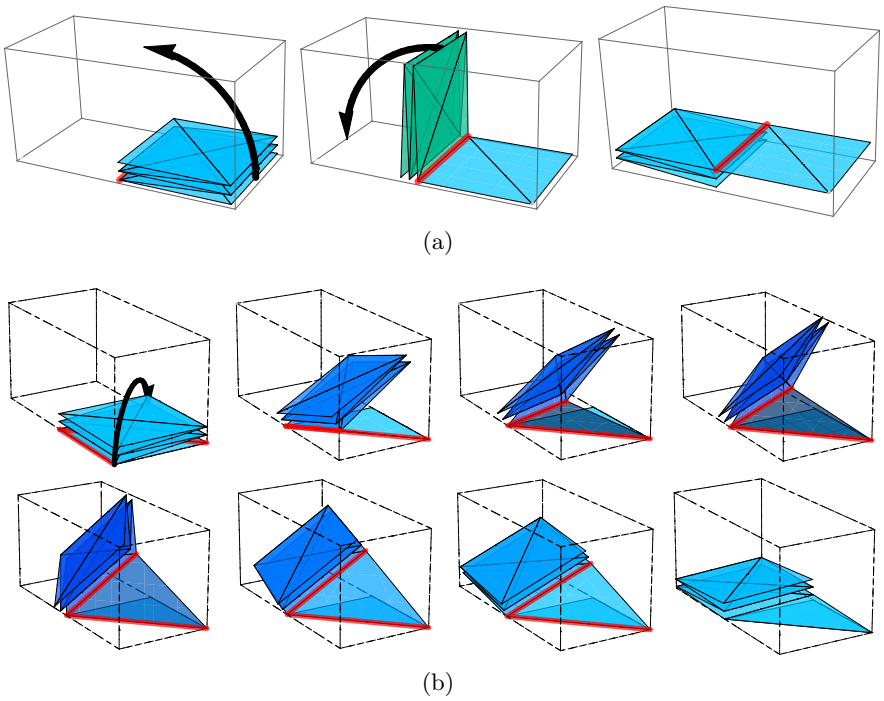


Fig. 8. Canonical strip, face-up cases. (a) Straight. (b) Turn where e_{23} is flat.

previous portions of the strip. Thus, we could wrap the strip around P's surface even if P's interior were already a filled solid. This restriction helps us think about folding the strip locally, because some of P's surface may have already been folded (and it thus should not be penetrated) by earlier parts of the strip.

It suffices to show, regardless of the local geometry of the polyhedron at the grid square where the milling tour either goes straight or turns, and regardless of whether the accordion faces up or down relative to the grid square it is covering, that we can maneuver the accordion in a way that allows us to unroll as many squares as necessary to perform the milling-tour move. Fig. 8 shows two key cases for unrolling part of the accordion of a canonical strip. See [5] for details.

Acknowledgments. We thank ByoungKwon An and Daniela Rus for several helpful discussions about programmable matter that motivated this work.

References

1. B. An, N. Benbernou, E. D. Demaine, and D. Rus. Planning to fold multiple objects from a single self-folding sheet. *Robotica*, 29(1):87–102, 2011. Special issue on Robotic Self-X Systems.

2. E. M. Arkin, M. A. Bender, E. D. Demaine, S. P. Fekete, J. S. B. Mitchell, and S. Sethia. Optimal covering tours with turn costs. *SIAM Journal on Computing*, 35(3):531–566, 2005.
3. E. M. Arkin, S. P. Fekete, and J. S. B. Mitchell. Approximation algorithms for lawn mowing and milling. *Computational Geometry: Theory and Applications*, 17(1–2):25–50, 2000.
4. N. M. Benbernou. *Geometric Algorithms for Reconfigurable Structures*. PhD thesis, Massachusetts Institute of Technology, September 2011.
5. N. M. Benbernou, E. D. Demaine, M. L. Demaine, and A. Lubiw. Universal hinge patterns for folding strips efficiently into any grid polyhedron. arXiv:1611.03187, 2016. https://arXiv.org/abs/1611.03187.
6. N. M. Benbernou, E. D. Demaine, M. L. Demaine, and A. Ovadya. Universal hinge patterns to fold orthogonal shapes. In *Origami^5: Proceedings of the 5th International Conference on Origami in Science, Mathematics and Education*, pages 405–420. A K Peters, Singapore, 2010.
7. K. C. Cheung, E. D. Demaine, J. Bachrach, and S. Griffith. Programmable assembly with universally foldable strings (moteins). *IEEE Transactions on Robotics*, 27(4):718–729, 2011.
8. A. E. F. Clementi, P. Crescenzi, and G. Rossi. On the complexity of approximating colored-graph problems. In *Proceedings of the 5th Annual International Conference on Computing and Combinatorics*, volume 1627 of *Lecture Notes in Computer Science*, pages 281–290, 1999.
9. E. D. Demaine, M. L. Demaine, and J. S. B. Mitchell. Folding flat silhouettes and wrapping polyhedral packages: New results in computational origami. *Computational Geometry: Theory and Applications*, 16(1):3–21, 2000.
10. E. D. Demaine and T. Tachi. Origamizer: A practical algorithm for folding any polyhedron. Manuscript, 2017.
11. B. Genc. *Reconstruction of Orthogonal Polyhedra*. PhD thesis, University of Waterloo, 2008.
12. E. Hawkes, B. An, N. M. Benbernou, H. Tanaka, S. Kim, E. D. Demaine, D. Rus, and R. J. Wood. Programmable matter by folding. *Proceedings of the National Academy of Sciences of the United States of America*, 107(28):12441–12445, 2010.
13. D. S. Hochbaum. Efficient bounds for the stable set, vertex cover and set packing problems. *Discrete Applied Mathematics*, 6(3):243–254, 1983.
14. R. J. Lang. A computational algorithm for origami design. In *Proceedings of the 12th Annual ACM Symposium on Computational Geometry*, pages 98–105, Philadelphia, PA, May 1996.
15. R. J. Lang and E. D. Demaine. Facet ordering and crease assignment in uniaxial bases. In *Origami^4: Proceedings of the 4th International Conference on Origami in Science, Mathematics, and Education*, pages 189–205, Pasadena, California, September 2006. A K Peters.

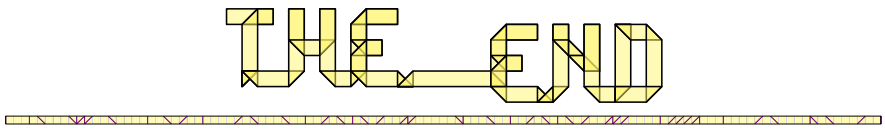


Fig. 9. An example of joining together a few letters from our typeface in Fig. 2. Unfolding (bottom) not to scale with folding (top).