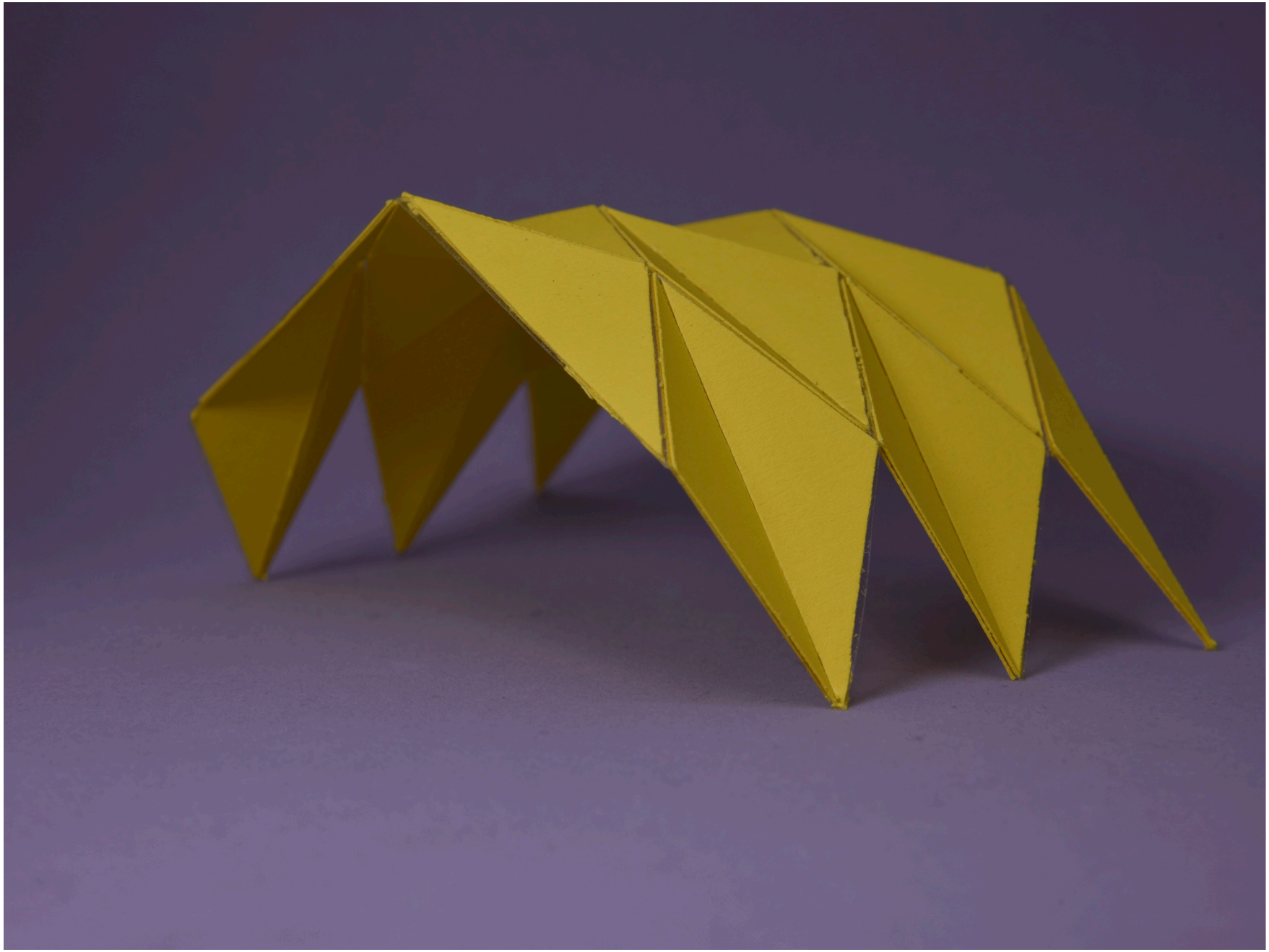
Algorithmic Design of Origami Mechanisms and Tessellations

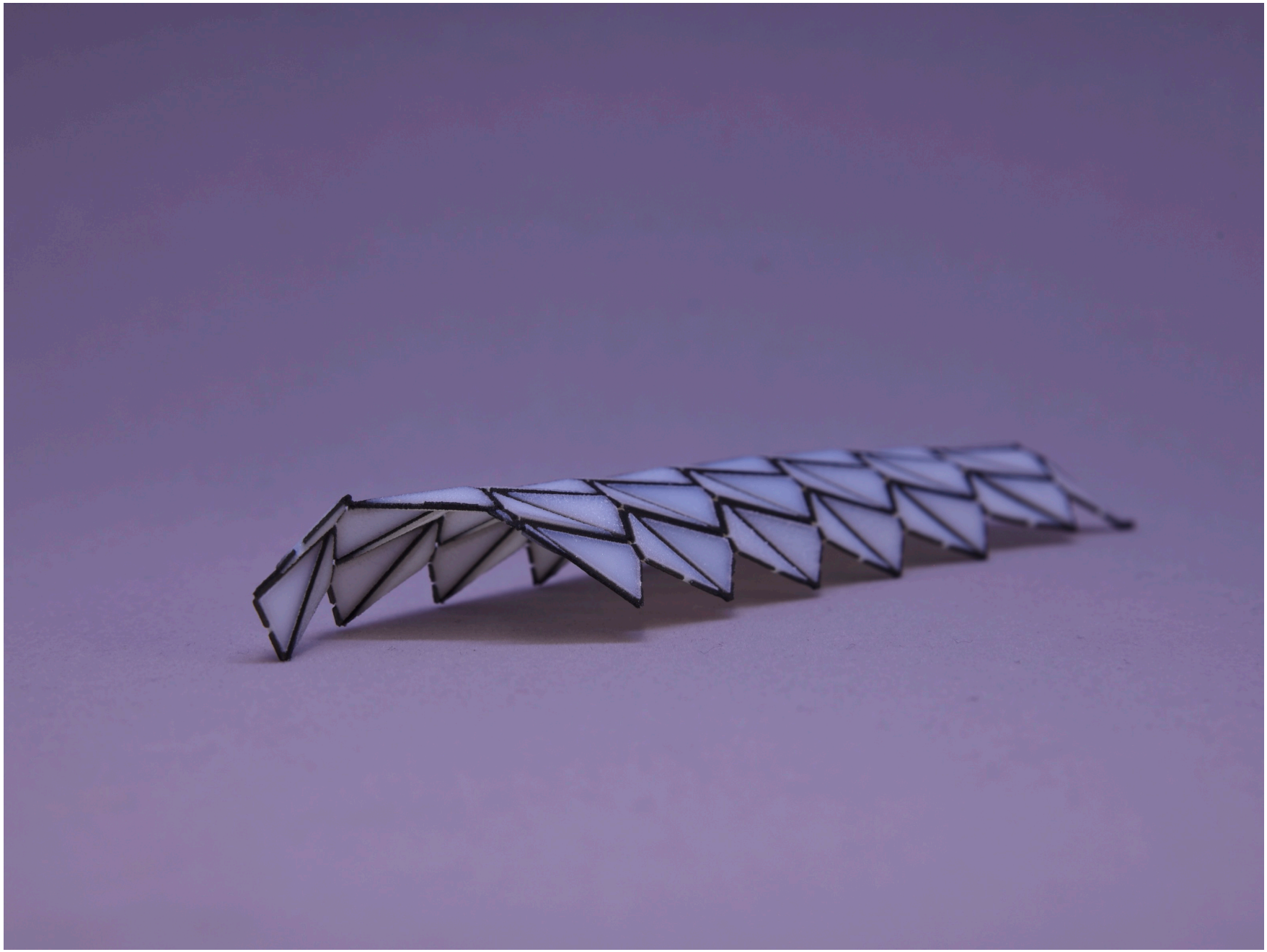
Supplementary Information

Andreas Walker[1] and Tino Stankovic[1,*]

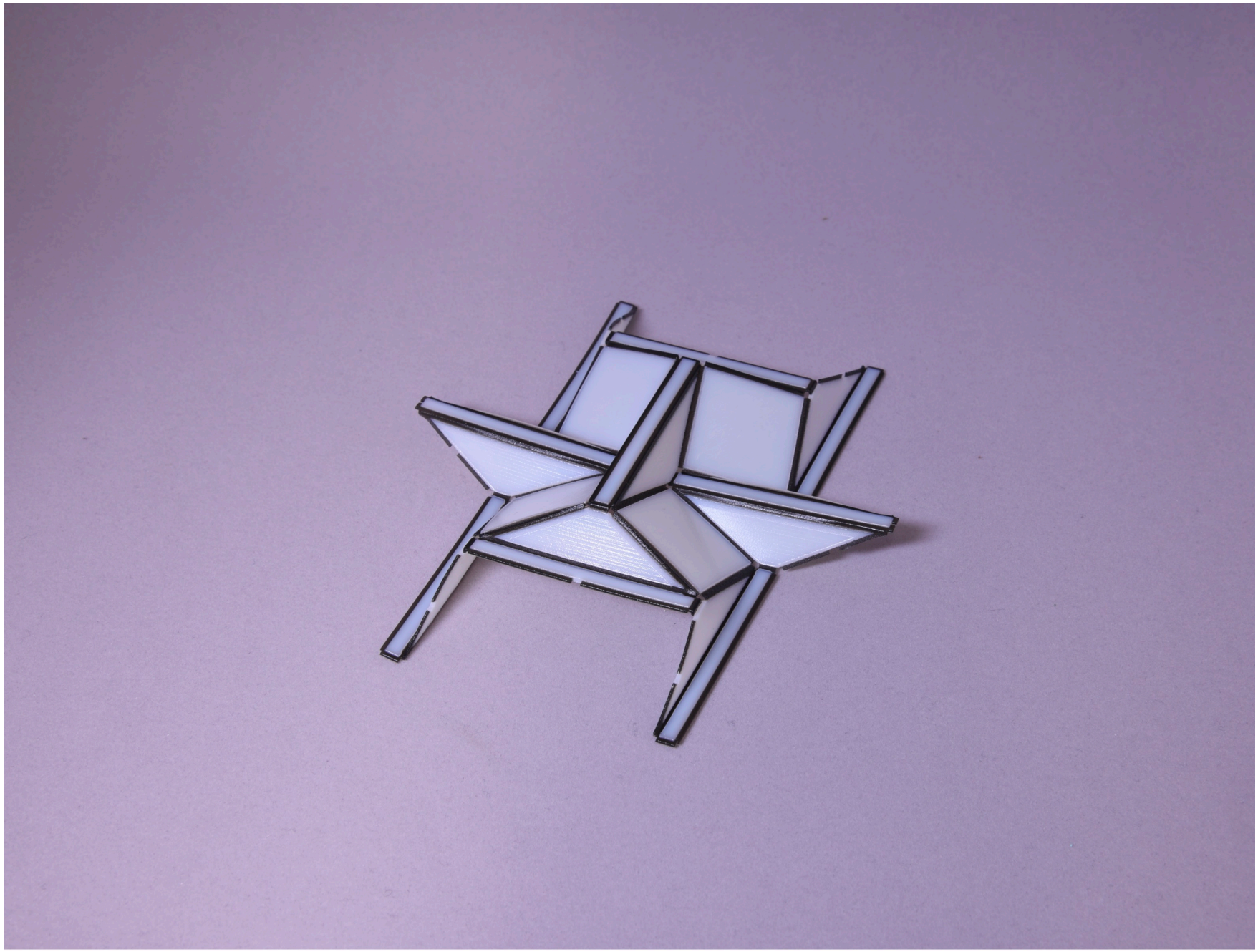
[1]Engineering Design and Computing Laboratory, Dept. of Mechanical and Process Engineering, ETH Zurich, Tannenstrasse 3, 8092, Zurich, Switzerland
[*]Corresponding author: tinos@ethz.ch



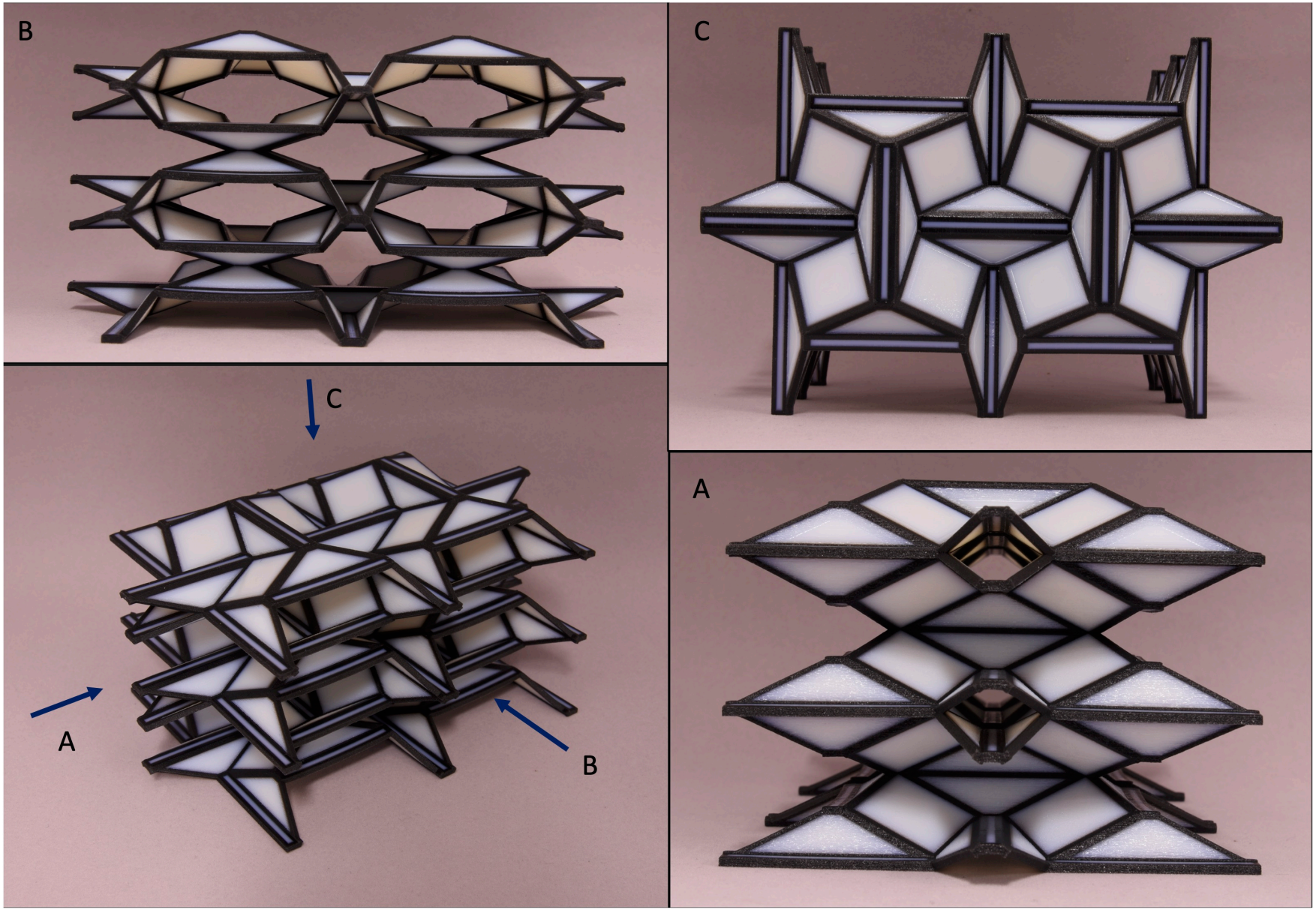
Supplementary Figure 1: Prototype of Yoshimuras pattern (Fig. 3d) in folded configuration.



Supplementary Figure 2: Prototype of Huffmans grid (Fig. 3e) in folded configuration.



Supplementary Figure 3: Prototype of square twist pattern (Fig. 3f) in folded configuration.

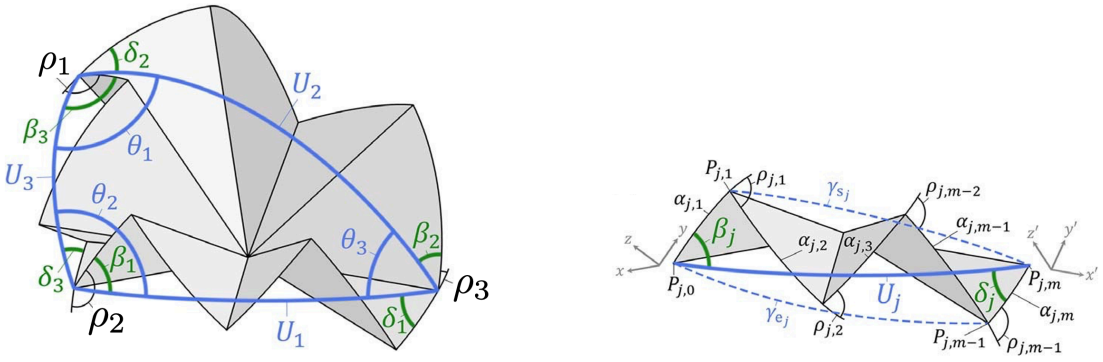


Supplementary Figure 4: Prototype of stacked square twist pattern (Fig. 4h). The prototype corresponds only to a part (approx. 1/4) of the structure in Fig. 4h.

Supplementary Note 1: Graph-Based Representation of Origami Crease Patterns

Here, we provide a detailed description on how crease patterns are represented using directed acyclic graphs and give an overview of the implementation. The concepts presented here are based on Zimmermann et al.[1].

Principle of Three Units (PTU) Here, we briefly reintroduce the Principle of Three Units as described in detail by Zimmermann et al. [1]. The principle shows how grouping the sectors around a vertex in three units can be used to determine the foldability of a generic vertex and calculate remaining dihedral angles.



(a) A folded origami vertex.

(b) A unit of the vertex in Supp. Fig. 5a.

Supplementary Figure 5: Geometry of a folded vertex and of a single unit. Reprinted from [1] with permission from ASME.

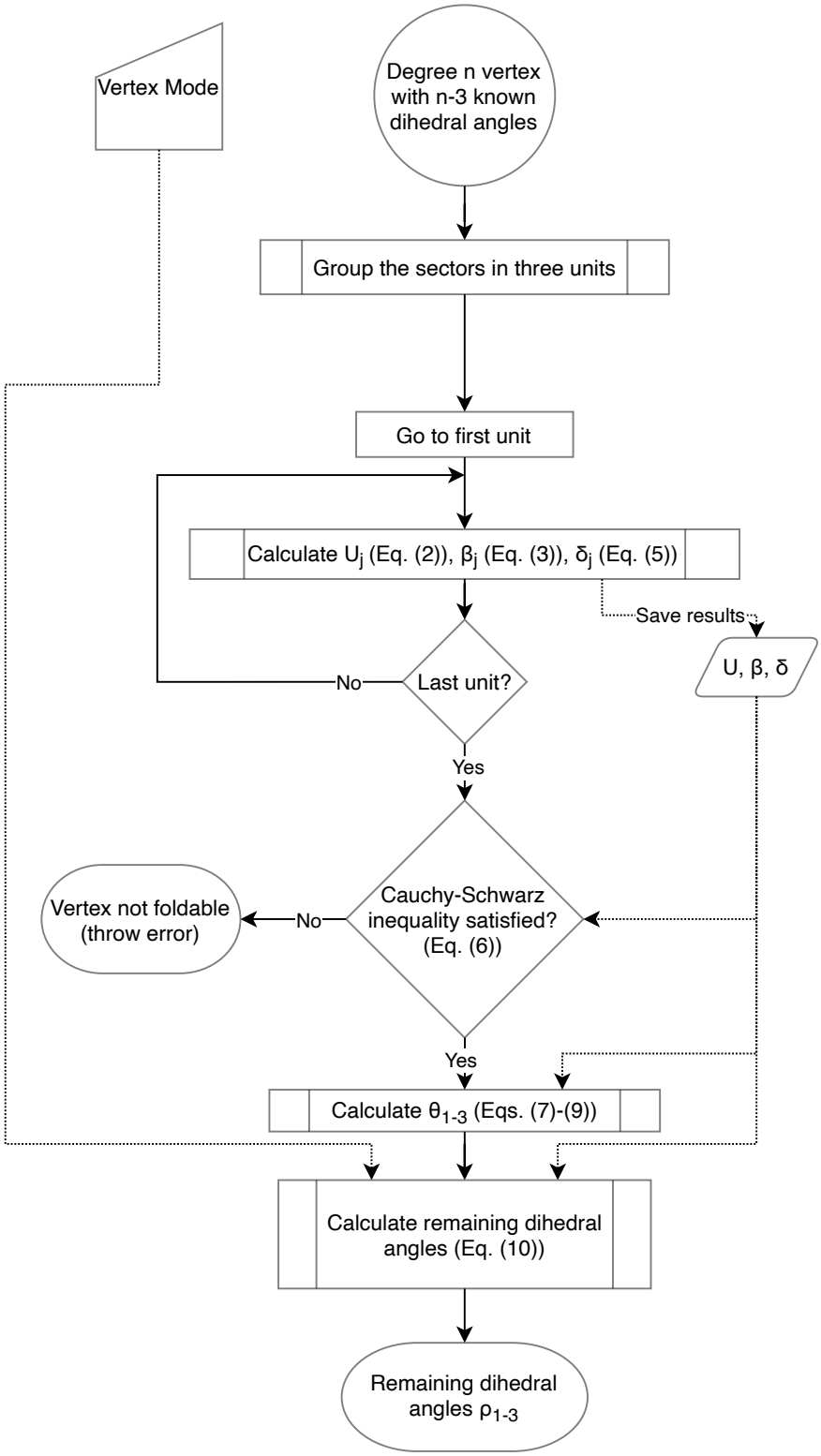
To show that, we consider a generic vertex as shown in Supp. Fig. 5a. We consider all sector angles α and the dihedral angles ρ known except for those marked with ρ_{1-3}. To calculate these three angles, we split the sectors around the vertex into three sets ("units"), such that each crease with unknown dihedral angle lies between two units. If there is a sector lying between two creases with unknown dihedral angles, this sector forms a unit of its own. To calculate the angles ρ_{1-3}, we first calculate the angles U_j, β_j and δ_j for each unit, based on which ρ_{1-3} are calculated in a next step.

An example for a unit is shown in Supp. Fig. 5b. To analyze the unit, we define a reference system where the vertex lies at $(0,0,0)$ and the point $P_{j,0}$ lies at $(1,0,0)$. Using the known dihedral and sector angles, the position of the point $P_{j,m}$ is found by alternating rotations around the z- and x-axis:

$$P_{j,m} = \underbrace{\left(\prod_{i=1}^{m-1} \mathbf{R}_z(\alpha_{j,i})\mathbf{R}_x(\rho_{j,i})\right)\mathbf{R}_z(\alpha_{j,m})}_{\mathbf{B}} P_{j,0} \tag{1}$$

where $\mathbf{R}_x(\varepsilon)$ and $\mathbf{R}_z(\varepsilon)$ are the standard rotation matrices around the x- and z- axis, respectively (note that dihedral angles ρ are defined to be positive for valley folds). Based on this, we calculate the unit angle U_j:

$$U_j = \arccos(P_{j,0} \cdot P_{j,m}) \tag{2}$$



Supplementary Figure 6: Evaluation of an interior vertex using the PTU.

We use an analogous calculation to obtain the angles γ_{s_j} and γ_{e_j} for each unit. Now, the angle β_j can be calculated using the spherical law of cosines:

$$\beta_j = -\text{sign}(P^z_{j,m}) \arccos\left(\frac{\cos(\gamma_{s_j}) - \cos(\alpha_{j,1})\cos(U_j)}{\sin(\alpha_{j,1})\sin(U_j)}\right) \tag{3}$$

where $\text{sign}(P^z_{j,m})$ is the sign of the z-coordinate of the point $P_{j,m}$. The angle β_j is positive when measured "upwards" from the first sector to the connection between $P_{j,0}$ and $P_{j,m}$ and negative when measured "downwards". For example, in Supp. Fig. 5a, $\beta_2 > 0°$ and $\beta_{1,3} < 0°$. To calculate the angle δ_j, we first find the coordinates of $P'_{j,0}$ in the reference system (x', y', z') shown in Supp. Fig. 5b. In this system, $P'_{j,m}$ lies at (1,0,0). Since (1) is still valid, $P'_{j,0}$ can be found by solving the system

$$P'_{j,m} = \mathbf{B}P'_{j,0} \tag{4}$$

We now calculate δ_j using the spherical law of cosines:

$$\delta_j = -\text{sign}(P'^z_{j,0}) \arccos\left(\frac{\cos(\gamma_{ej}) - \cos(\alpha_{j,m})\cos(U_j)}{\sin(\alpha_{j,m})\sin(U_j)}\right) \tag{5}$$

In analogy to β_j, δ_j is positive when measured "upwards" from the last sector and accordingly negative when measured "downwards" ($\delta_{1,2} > 0°$ and $\delta_3 < 0°$ in Supp. Fig. 5a).

The vertex is rigid foldable if (and only if) the three unit angles U_{1-3} satisfy the Cauchy-Schwartz inequality:

$$\max(U_{1-3}) \leq \text{median}(U_{1-3}) + \min(U_{1-3}) \tag{6}$$

If this condition is not satisfied, the vertex is not foldable and no dihedral angles ρ_{1-3} can be found. If the condition is satisfied, we use the spherical law of cosines in the triangle formed by the angles U_{1-3} to find the angles ϑ_{1-3}:

$$\theta_1 = \arccos\left(\frac{\cos(U_1) - \cos(U_2)\cos(U_3)}{\sin(U_2)\sin(U_3)}\right) \tag{7}$$

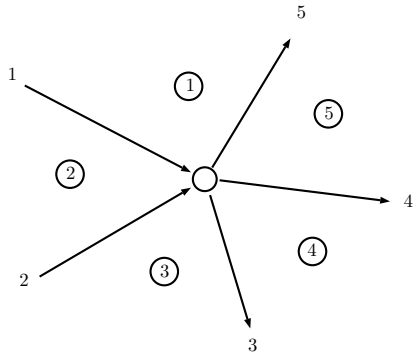
$$\theta_2 = \arccos\left(\frac{\cos(U_2) - \cos(U_1)\cos(U_3)}{\sin(U_1)\sin(U_3)}\right) \tag{8}$$

$$\theta_3 = \arccos\left(\frac{\cos(U_3) - \cos(U_1)\cos(U_2)}{\sin(U_1)\sin(U_2)}\right) \tag{9}$$

Next, we calculate the missing dihedral angles ρ_{1-3}:

$$\begin{pmatrix} \rho_1 \\ \rho_2 \\ \rho_3 \end{pmatrix} = \begin{cases} \begin{pmatrix} \beta_3 \\ \beta_1 \\ \beta_2 \end{pmatrix} + \begin{pmatrix} 180° - \theta_1 \\ 180° - \theta_2 \\ 180° - \theta_3 \end{pmatrix} + \begin{pmatrix} \delta_2 \\ \delta_3 \\ \delta_1 \end{pmatrix} & \text{mode "true"} \\ \begin{pmatrix} \beta_3 \\ \beta_1 \\ \beta_2 \end{pmatrix} + \begin{pmatrix} \theta_1 - 180° \\ \theta_2 - 180° \\ \theta_3 - 180° \end{pmatrix} + \begin{pmatrix} \delta_2 \\ \delta_3 \\ \delta_1 \end{pmatrix} & \text{mode "false"} \end{cases} \tag{10}$$

Supp. Fig. 5a illustrates the case for the mode "true". The evaluation of an interior vertex using the PTU is summarized in Supp. Fig. 6.



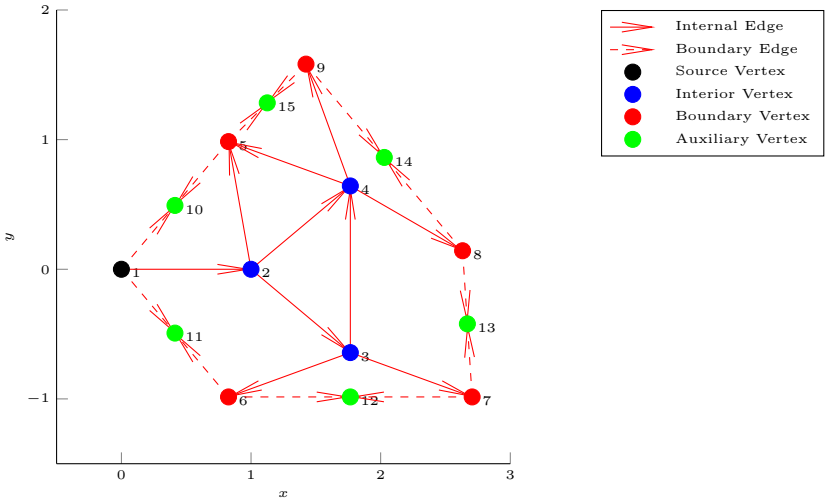
Supplementary Figure 7: Sample vertex in a crease pattern. The numbers indicate the local indexation of edges and faces (circled). Local indices increase in counterclockwise direction.

Interpretation of Vertices At each vertex, we consider the edges whose dihedral angle is known as inputs to the vertex, whereas the three edges whose dihedral angles are calculated based on the inputs and the sector angles are referred to as vertex outputs. This is illustrated in Supp. Fig. 7 on a vertex of degree 5 with two input edges (1 and 2) and three output edges (3, 4 and 5). Each vertex output, or, more precisely, its dihedral angle, serves as input to another vertex. Therefore, a crease line is interpreted as (or represented by) a directed edge, pointing from the vertex that is used to calculate its dihedral angle to the vertex where its dihedral angle is used to calculate the dihedral angles of further edges.

Vertex and Edge Types Each crease pattern contains exactly one source vertex.. Such a vertex has only one output and no inputs. The dihedral angle of its output must be predefined and is therefore considered to be known. Thus, the source vertex is unique and defines the origin of the crease pattern generation. The position of the source vertex in the folded configuration is always equal to its position in the unfolded configuration. An interior vertex is located at the interior of a crease pattern, has at least one input and exactly three outputs (e.g. the vertex in Supp. Fig. 7 is an interior vertex). A boundary vertex lies on the boundary of the crease pattern and has at least one input and no outgoing internal edges. To control the shape of boundary faces, boundary edges and auxiliary vertices are used. Boundary edges connect boundary or source vertices (all of them therefore have two additional outgoing boundary edges) with auxiliary vertices. Auxiliary vertices have exactly two incoming boundary edges and no outputs. They only control the shape of the boundary faces and have no impact on the folding kinematics.

Finally, a crease pattern graph is a directed acyclic graph that models a crease pattern and represents kinematic dependencies between the vertices. An example of a crease pattern graph is shown in Supp. Fig. 8, where all vertex and edge types are shown.

Restrictions All crease pattern graphs must be acyclic graphs to allow for a direct evaluation of the folding kinematics [1]. Furthermore, no internal voids are allowed, implying that all boundary edges must be connected to a closed curve.



Supplementary Figure 8: Crease pattern graph of the example presented in Fig. 1B and 1C of the main text. The numbers represent indices of vertices.

Supplementary Note 2: Kinematics Evaluation

Supp. Alg. 1 and Supp. Fig. 9 summarize the procedure to evaluate the folded configuration of a crease pattern with a single folding degree of freedom. The procedure involves the calculation of all dihedral angles ρ (cf. Fig. 1A of the main text for notational convention of angles), crease and normal vectors as well as vertex positions. The conditions that need to be fulfilled such that a vertex can be evaluated and how a vertex is evaluated depend on the vertex type.

Supplementary Algorithm 1: Algorithm to obtain the folded configuration of an origami

```
while not all vertices evaluated do
    for all vertices do
        if vertex is evaluated then
            continue
        else
            if vertex can be evaluated then
                evaluate vertex;
            end
        end
    end
end
```

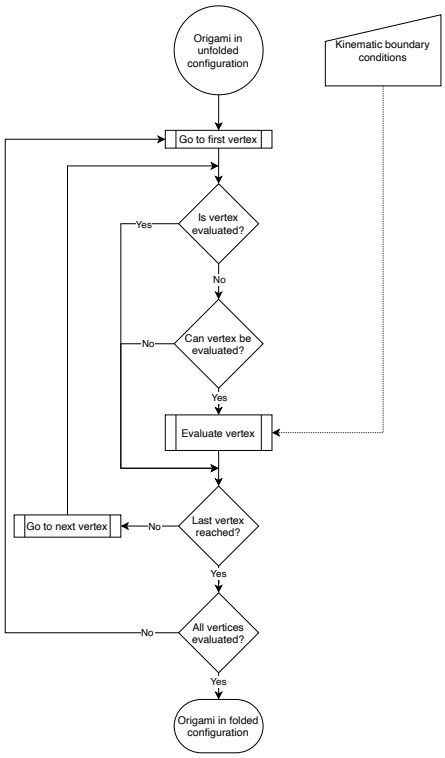
Supplementary Note 2.1: Source Vertex

Requirements for Evaluation The source vertex can be evaluated if a dihedral angle for its outgoing internal edge is known (must be given) and either an adjacent face is marked as fixed (Supp. Fig. 10) or symmetric folding is used (Supp. Fig. 11).

Evaluation with Fixed Face A possible configuration when evaluating a source vertex with an adjacent face being fixed is shown in Supp. Fig. 10. All sector angles α and edge lengths can be obtained from the unfolded/initial ($\rho = 0° \ \forall$ edges) configuration. The crease vector, i.e. a unit vector with the direction of the respective crease in the folded configuration, of the internal edge and the location of its end vertex can be calculated directly and with no change with respect to the unfolded configuration. Furthermore, the normal vector of the fixed face is also obtained from the unfolded configuration. Next, the crease vector of the internal edge is rotated around this normal vector by the sector angle of the fixed face at the source vertex ($-\alpha_1$ in Supp. Fig. 10). This gives the crease vector of the other edge on the boundary of the fixed face. Since its length is known, the location of the vertex at the end of that edge can be found too.

The normal vector to the non-fixed face can be found by rotating the normal vector of the fixed face around the internal edge by its dihedral angle (which was specified by the user; denoted as ρ in Supp. Fig. 10; defined as positive for valley folds). Rotating the direction vector of the internal edge around this vector by the corresponding sector angle (α_2 in the case shown in Supp. Fig. 10) results in the direction vector of the remaining boundary crease. Again, its length can be used to obtain the location of its end vertex (not shown in Supp. Fig. 10).

Evaluation with Symmetric Folding A possible configuration for this scenario is shown in Supp. Fig. 11. Again, the crease vector of the internal edge is given. The two face normals are



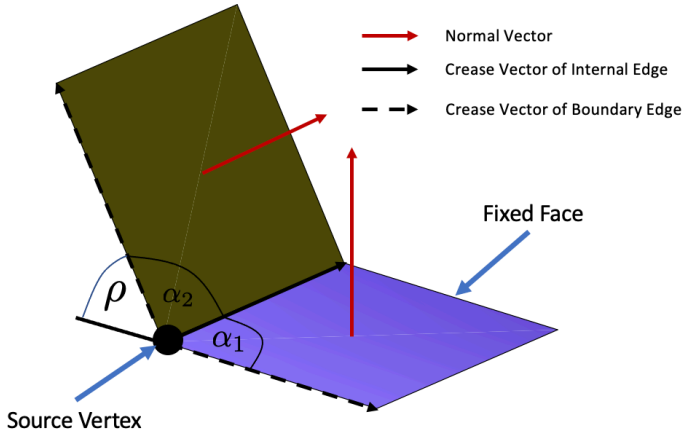
Supplementary Figure 9: Algorithm to obtain the folded configuration of an origami (Illustration of Supp. Alg. 1).

obtained by rotating the local z-Axis around this edge by half the (predefined) dihedral angle ρ in either direction. Rotating the direction vector of the internal edge around these normals by the respective sector angles results in the direction vectors of the two boundary edges. The lengths of the edges are then used to obtain the locations of their end vertices in the folded configuration (not shown in Supp. Fig. 11).

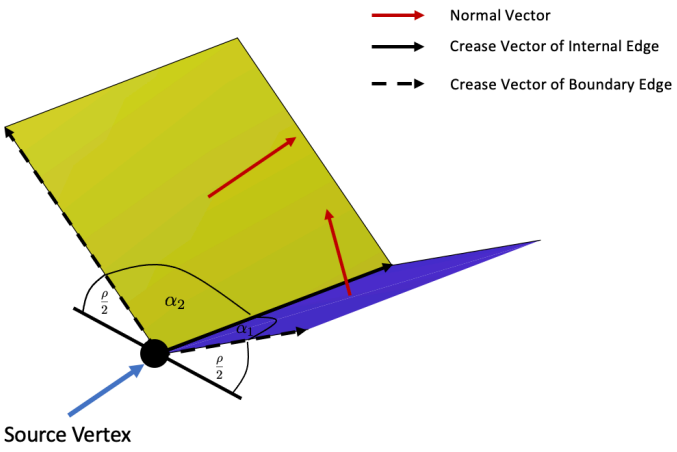
Supplementary Note 2.2: Interior Vertex

Requirements for Evaluation Interior vertices can be evaluated once all of its direct predecessors are evaluated.

Evaluation The evaluation of an interior vertex is shown on the example of a vertex of degree 4 in Supp. Fig. 12. Since all sector angles and all incoming dihedral angles are known, the remaining three dihedral angles (of the outgoing edges) can be calculated. This calculation yields two possible results, which can be chosen from. We refer to these two options as "modes". Each interior vertex can either have mode "true" or mode "false". These are marked with circles (true) and crosses (false) in the crease pattern graphs in this document. Then, the crease vector of the last input (see Supp. Fig. 7 for indexation convention), oriented such that it points away from the vertex, is rotated around the normal vector of the face between the last incoming and first outgoing edge by the corresponding sector angle. This results in the crease vector of the first outgoing edge. Next, the normal vector that was used to obtain the crease vector of the

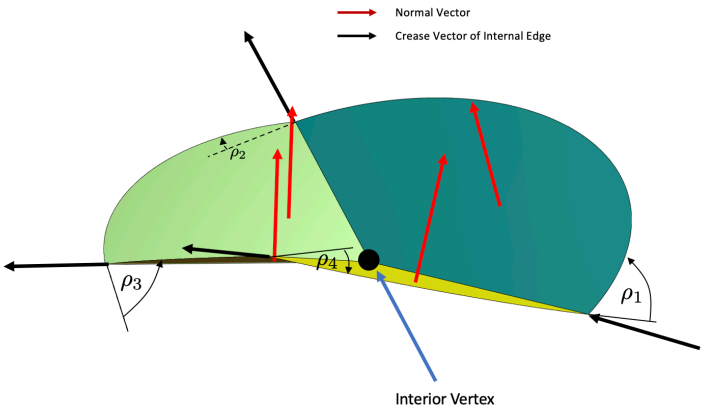


Supplementary Figure 10: Geometry of a source vertex in folded configuration where one face is fixed.



Supplementary Figure 11: Geometry of a source vertex in folded configuration where symmetric folding is used.

first outgoing edge is rotated around that same crease vector by its dihedral angle to obtain the normal vector of the next face. This is repeated until all crease vectors and all normal vectors around the vertex are known. These crease vectors are used to calculate the positions of the successor vertices (not shown in Supp. Fig. 13).



Supplementary Figure 12: Geometry of an interior vertex in folded configuration.

Supplementary Note 2.3: Boundary Vertex

Requirements for Evaluation Boundary vertices can be evaluated once its first and last direct predecessor/input are evaluated (cf. Supp. Fig. 7 for indexation convention). If there are less than three direct predecessors, this implies that all predecessors need to be evaluated.

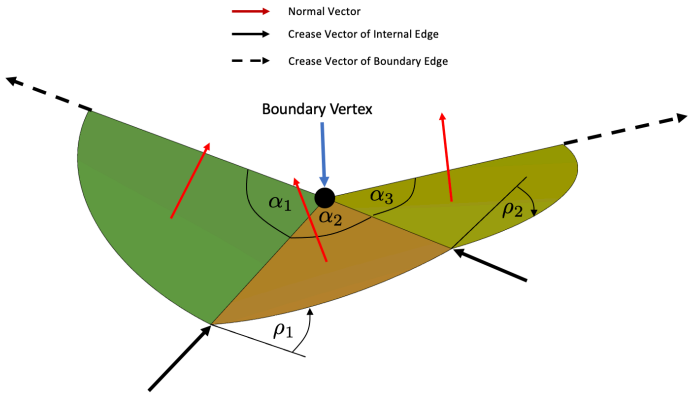
Evaluation Supp. Fig. 13 shows a possible configuration at a boundary vertex with two incoming edges. The crease vector of the first output (i.e. before the "void" sector when marching in counterclockwise order) is found by rotating the crease vector (pointing away from the vertex) of the last input around the normal vector of the and last face by the respective sector angle. Similarly, the crease vector of the last output is found by rotating the crease vector of the first input (pointing away from the vertex) around the normal vector of the first face by the first sector angle, but in the opposite direction ($-\alpha_1$ when using the right-hand rule with the face normal or α_1 when using the left-hand rule). These crease vectors can then be used to calculate the positions of the successor vertices (typically auxiliary vertices; not shown in Supp. Fig. 13).

Supplementary Note 2.4: Auxiliary Vertex

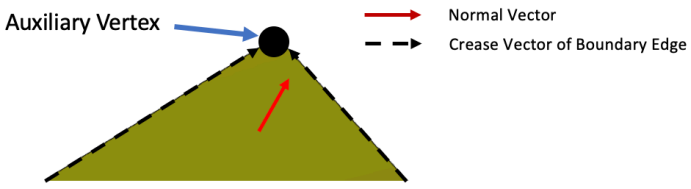
An auxiliary vertex (cf. Supp. Fig. 14) can be evaluated if its position is known, which is the case when either of its direct predecessors is evaluated. The evaluation itself is trivial, as the vertex is simply marked as evaluated.

Supplementary Note 2.5: Conditions for Rigid and Flat Foldability

For an arbitrary positioning of vertices, flat foldability is not guaranteed and the pattern may only fold rigidly for a limited range of input dihedral angles. Rigid foldability (for a given input dihedral angle) is given if (and only if) for all interior vertices, dihedral angles of its outgoing



Supplementary Figure 13: Geometry of a boundary vertex in folded configuration.



Supplementary Figure 14: Geometry of an auxiliary vertex in folded configuration.

edges can be found. To calculate these angles at any interior vertex, the sectors around the are grouped in three units, as described in further detail in [1]. The first unit contains all faces that border at least one input edge and the other two units consist of the single sectors between the outputs. These three units form a spherical triangle that forms the basis of the calculation of the remaining dihedral angles. If, however, the Cauchy-Schwarz inequality is not satisfied at this spherical triangle, the vertex (and therefore the pattern) is not rigidly foldable for this input dihedral angle.

The foldability of an interior vertex depends on its predecessors and can be changed by altering their sector angle configurations (i.e. moving vertices) or modes. It is also dependent on the sector angle configuration around the vertex itself, but not on its mode.

The range of input dihedral angles for which a pattern (with given mode assignments) folds rigidly can, for example, be obtained by the bisection method. We notice that this range (interval) can be empty. To obtain the number of kinematic paths of a pattern with n interior vertices, all 2^n possible mode combinations have to be enumerated and for each combination, it has to be evaluated whether the interval of valid input dihedral angles is empty (invalid kinematic path) or not (valid kinematic path).

Each folded state of an origami on a kinematic path (i.e. $\rho \neq 0°$ for at least one edge) can be mirrored to a "mirrored state" with $\rho' = -\rho$ for all edges. This is achieved by changing the sign of the input dihedral angle ($\rho'_{\text{in}} = \rho_{\text{in}}$) and switching the modes of all interior vertices. Since (due to the inherent symmetry) such a state can be considered to be on the same kinematic path as the original state, only one sign of the input dihedral angle ($\rho_{\text{in}} \geq 0°$) has to be considered when enumerating ("counting") kinematic paths.

If the pattern was generated using the quadrilateral-closing operation, some implications have to be considered when enumerating kinematic paths (cf. Supp. Note 5.4).

Supplementary Note 2.6: Examples

Fig. 1B/C of the main text shows a simple example with only three internal vertices. The crease pattern graph is shown in more detail in Supp. Fig. 8. To obtain the eight folded configurations that are shown, the option of symmetric folding and the parameters in Supp Tab. 1 are used.

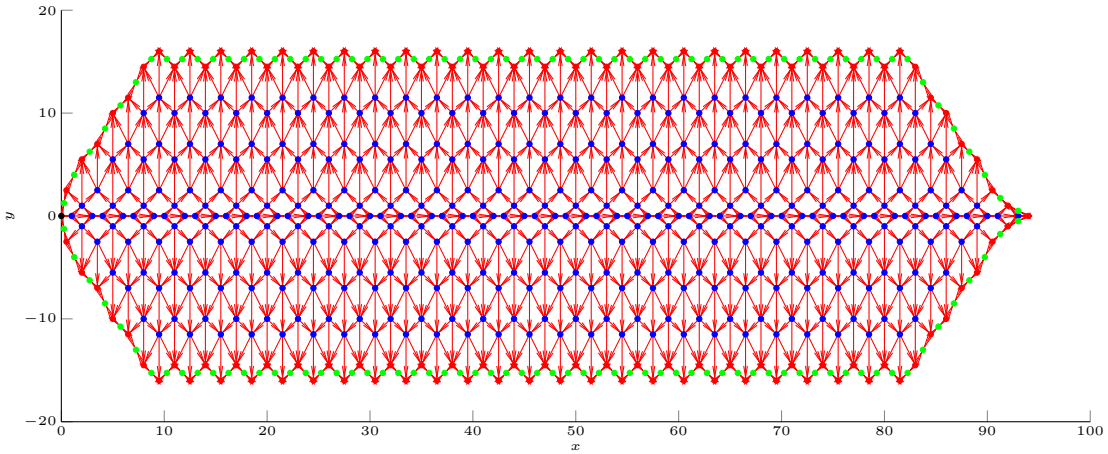
	Mode of Vertex 2	Mode of Vertex 3	Mode of Vertex 4	Dihedral Angle at Input Crease
1	false	false	true	90°
2	true	true	true	10°
3	true	true	false	50°
4	true	false	true	5°
5	true	false	false	20°
6	false	true	true	120°
7	false	true	false	70°
8	false	false	false	90°

Supplementary Table 1: Parameters for the folded configurations shown in Figs. 1B and 1C in the main text.

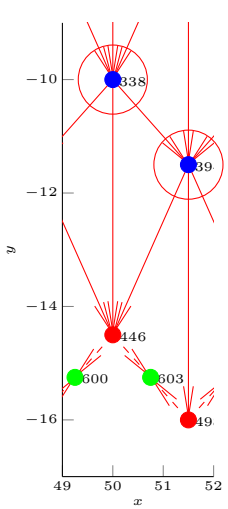
Fig. 1G in the main text hows a more complex crease pattern (Supp. Fig. 15) that was designed manually. The basis of the pattern is a "strip" of vertices of degree 4 that starts at the source vertex on the very left and ends at the vertex at the very right (vertices 62-74 in Supp.

Fig. 15c). This strip was found to fold to a curved line where the vertex positions can be used to adjust the magnitude of the curvature. The modes of the vertices on the axis of symmetry, on the other hand, can be used to change the sign of the curvature. The pattern added to both sides of the strip, with the mode selections as shown, was found to fold to a synclastic surface.

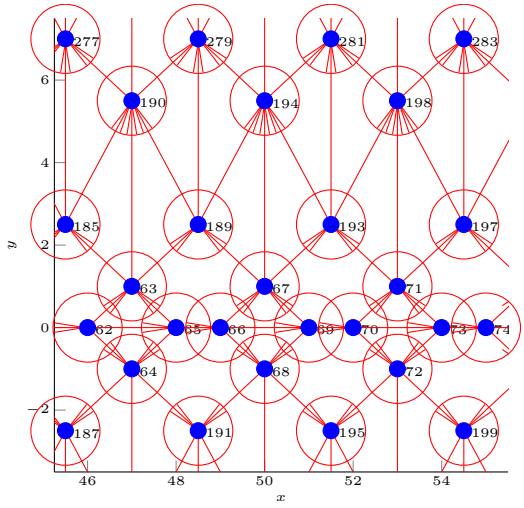
To obtain the folded configuration, the option of symmetric folding and an input dihedral angle of 5° is used. The complete pattern has 644 vertices, 1429 edges and 786 faces. Evaluation of the folded configuration takes 3.3 s on a 2.3 GHz Intel Core i9 processor.



(a) Complete crease pattern graph. For better clarity, no vertex indices and mode markers are shown.



(b) Detail from Supp. Fig. 15a



(c) Detail from Supp. Fig. 15a

Supplementary Figure 15: Crease pattern graph for the example shown in Fig. 1G of the main text. Interior vertices are marked with their assigned mode (convention defined in Supp. Note 2.2).

Supplementary Note 3: Pattern Optimization

Supplementary Note 3.1: Algorithm Overview

Supp. Algorithm 2 and Supp. Fig. 16 summarize the procedure to optimize an origami pattern. As optimization objective, we consider a point in 3D that a specific vertex needs to reach in the folded configuration. The success metric, which the optimization procedure minimizes, is the Euclidean distance between the point and the vertex. To that end, a set of predefined vertices is moved, which alters the folding kinematics and therefore the folded configuration. In the following, each step of the optimization problem is explained in further detail.

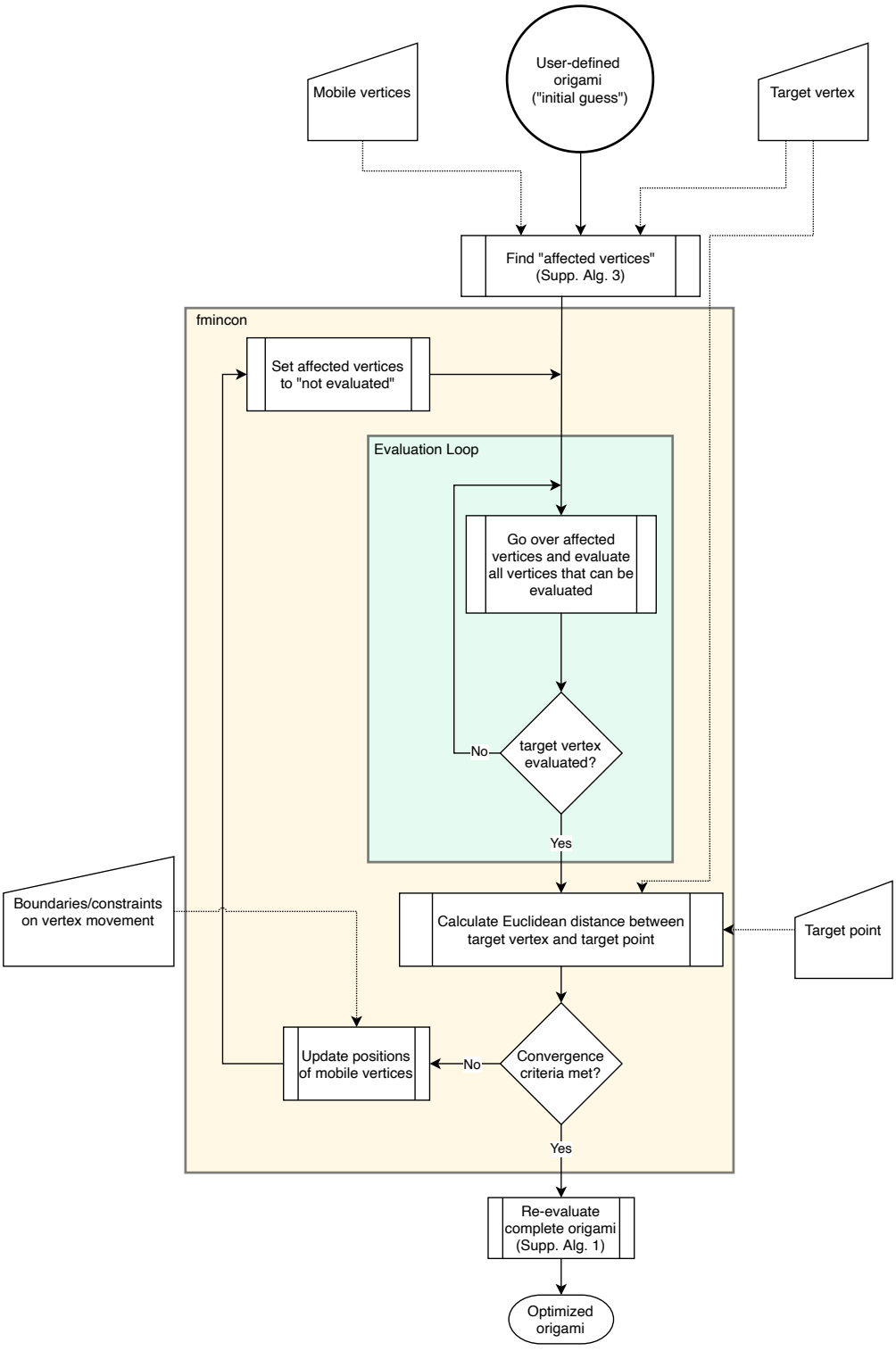
Supplementary Algorithm 2: Procedure for optimizing origami crease patterns

Input: 1) Fully evaluated origami (initial solution)
2) Definition of optimization problem

Result: Fully evaluated optimized origami

1 Identify the region that is affected by the optimization problem;
2 Use the interior-point optimizer in MATLAB to solve the optimization problem;
3 Re-evaluate the complete origami;

- The first input is an origami based on a valid crease pattern graph that is fully evaluated. It will serve as an initial solution to the optimization problem.
- The second input defines the optimization problem that will be solved. This definition includes:
  - A vertex whose position in the folded configuration should reach a certain point in 3D. This vertex is referred to as target vertex.
  - The coordinates of the point that the target vertex should reach in the folded configuration. This point is referred to as target point.
  - The vertices that are allowed to be moved. These vertices are referred to as mobile vertices and may contain the target vertex.
  - A distance by which the mobile vertices may, in the unfolded configuration, be moved in either direction.
  - (optional) Further constraints for the movement of the mobile vertices in the unfolded configuration. Examples are that they may be moved only in x- or y-direction or coupling two vertices to have identical x- or y-coordinates.
- The first step is to identify all vertices, edges and faces that are affected by the optimization problem (Line 1). As explained in the main text, not all vertices are affected in their folding kinematics by the movement of a mobile vertex. Also, not all vertices have an influence on the folding kinematics of the target vertex. To obtain the position of the target vertex (in the folded configuration) after mobile vertices are moved, only a subset of the origami needs to be reevaluated, which can constitute only a small partition of the full origami. These vertices, edges and faces are called "affected" by the optimization problem and are identified before starting the actual optimization. This step is explained in further detail in the next section.



Supplementary Figure 16: Procedure to optimize a pattern (Illustration of Supp. Alg. 2).

- Line 2 refers to the actual optimization. The interior-point optimizer in MATLAB is used to find the location of the mobile vertices such that the Euclidean distance between the target point and the location of the target vertex in the deformed configuration is minimized:

$$\begin{aligned} \underset{\vec{p}^{\,u}_{\mathrm{mv}}}{\text{minimize}} \quad & \|\vec{p}^{\,f}_{\mathrm{tv}}(\vec{p}^{\,u}_{\mathrm{mv}}) - \vec{p}_{\mathrm{tp}}\| \\ \text{subject to} \quad & \vec{x}_{\min} \leq \vec{p}^{\,u}_{\mathrm{mv}} \leq \vec{x}_{\max} \end{aligned} \tag{11}$$

  Where $\vec{p}^{\,u}_{\mathrm{mv}}$ denotes the positions of the mobile vertices in the unfolded configuration, $\vec{p}^{\,f}_{\mathrm{tv}}$ the position of the target vertex in the folded configuration, $\vec{p}_{\mathrm{tp}}$ the target point and $\vec{x}_{\min}$ and $\vec{x}_{\max}$ to lower and upper bounds, respectively. At each evaluation of an objective function, the mobile vertices are moved (updating $\vec{p}^{\,u}_{\mathrm{mv}}$), the relevant edge lengths and sector angles are recalculated and the affected vertices (as obtained in the previous step) are reevaluated to obtain $\vec{p}^{\,f}_{\mathrm{tv}}(\vec{p}^{\,u}_{\mathrm{mv}})$. The optimization is aborted if either the Euclidean distance between target vertex and target point becomes less than 1×10^{-4} or the objective function is evaluated 200 times. The interior-point optimizer of MATLAB (fmincon) is a general-purpose nonlinear solver that was chosen because it adheres well to our optimization model formulation.

- In the last step (Line 3), after the actual optimization, all edge lengths and sector angles that were affected by the movement of the mobile vertices are recalculated and all vertices that are affected by their movement are reevaluated. This step is necessary because during an objective function evaluation, only those edge lengths and sector angles are recalculated that affect the target vertex. Therefore, after the optimization, those edge lengths and sector angles that were changed during the optimization (due to the movement of the mobile vertices) but do not affect the target vertex were not recalculated. This last step fixes this by recalculating all edge lengths and sector angles that are affected by the movement of the mobile vertices and reevaluating all successors of the mobile vertices.

Supplementary Note 3.2: Finding Affected Vertices

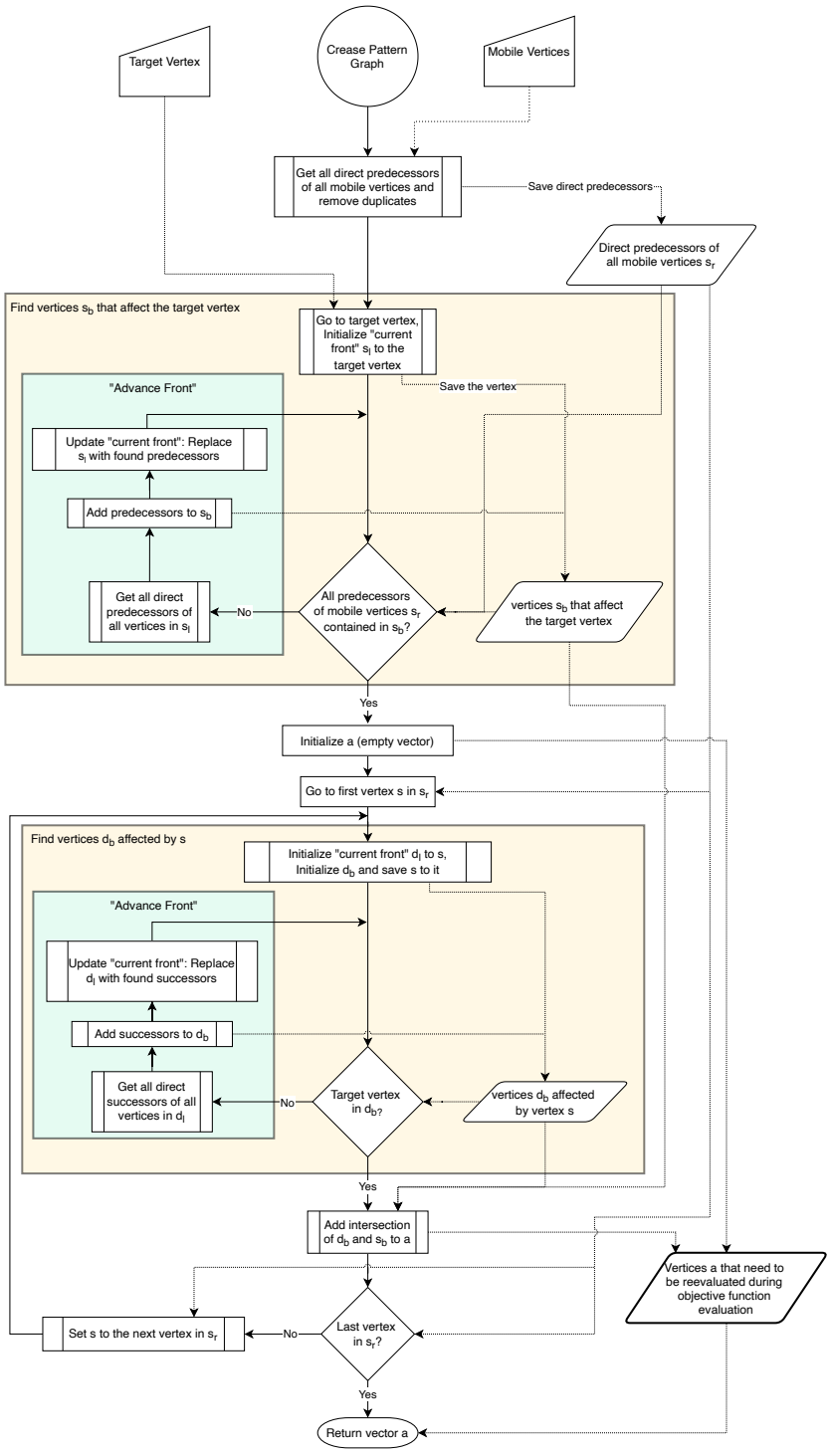
A core step in the optimization workflow is the identification of affected vertices, i.e. vertices that are affected by the movement of mobile vertices and also affect the target vertex. Supp. Algorithm 3 and Supp. Fig. 17 give an overview of how this is accomplished. An equivalent algorithm is used to obtain all faces and edges whose normal or crease vectors change when mobile vertices are moved.

Also, we note that the presented algorithm assumes that the target vertex requires reevaluation during optimization. However, reevaluating any of its direct predecessors is sufficient to get the updated position of the target vertex. The algorithm presented here is simple to implement and, compared to an alternative algorithm where the evaluation of a predecessor of the target vertex is sufficient, shows a computational cost that is only slightly higher.

Supplementary Note 3.3: Simple Example

Fig. 2B in the main text shows an example of crease pattern optimization in which the target vertex needs to be located in the desired position in the folded configuration.

The crease pattern graph of the initial solution is shown in Supp. Fig. 18. The optimization problem is defined as follows:



Supplementary Figure 17: Algorithm to obtain the vertices that need to be reevaluated after moving mobile vertices during an optimization (Illustration of Supp. Alg. 3).

Supplementary Algorithm 3: Algorithm to obtain the vertices that need to be reevaluated after moving mobile vertices during an optimization

Input: 1) crease pattern graph
2) mobile vertices and target vertex
Output: Vector a with all affected vertices

$s_r \leftarrow$ All direct predecessors of all mobile vertices;
Remove all duplicates from s_r;
Initialize s_l and s_b to contain the target vertex.;
while *not all members in* s_r *are in* s_b **do**
  Add all direct predecessors of all vertices in s_l to s_b;
  Update s_l to instead contain all predecessors of all vertices in s_l;
end
Initialize a (empty vector that will contain all affected vertices);
for *All vertices* s *in* s_r **do**
  Initialize d_l and d_b to contain s;
  while *target vertex is not a member of* d_b **do**
    Add all direct successors of all vertices in d_l to d_b;
    Update d_l to instead contain all predecessors of all vertices in d_l;
  end
  $a \leftarrow a \cup (d_b \cap s_b)$
end

Objective Function In the folded configuration (input dihedral angle = 30°, symmetric folding (Supp. Note 2.1)), the Euclidean distance between the target vertex (nr. 11) and the target point ($x = 3.5$, $y = 0.25$, $z = 2.5$) is to be minimized.

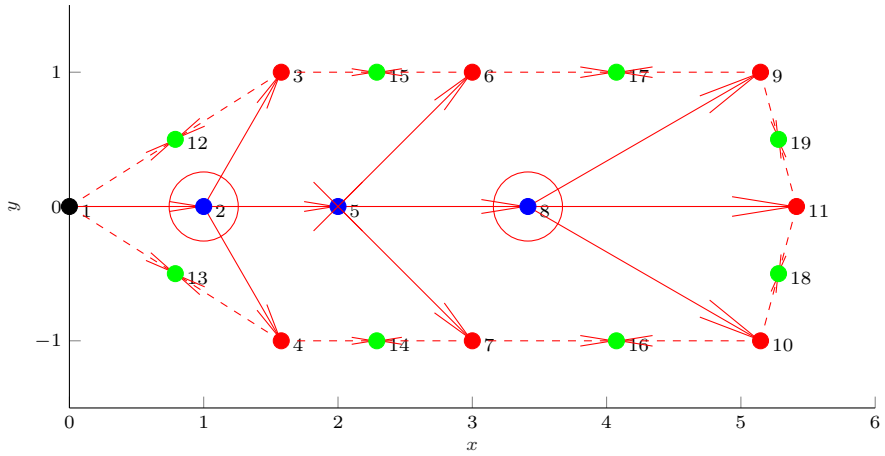
Constraints Vertices 8, 9 and 10 may be moved (mobile vertices), but only along the x-Axis in the unfolded configuration. They may move a distance of 1 in either direction ($x_{\text{old}} - 1 \leq x_{\text{new}} \leq x_{\text{old}} + 1$)

The solution is obtained using the optimization described in Supp. Alg. 2. After 22 iteration steps and 127 objective function evaluations, the Euclidean distance between target vertex and target point reaches 1×10^{-4} The optimization time is 1.315 s on a 2.3 GHz Intel Core i9 processor. The graph after the optimization is shown in Supp. Fig. 19.

Supplementary Note 3.4: Example "Elephant"

In the example presented in Fig. 2C of the main text, a pattern of 317 unit cells is optimized subsequently. The goal for each unit cell is that a target point sampled from the elephant curve [2] is reached in the folded configuration. Consequently, the complete pattern approximates the curve in its folded configuration.

Base Pattern The basis for the example presented in Fig. 2C in the main text is a periodic pattern with 317 unit cells that are optimized one unit cell a time. Supp. Fig. 20 shows two unit cells of the pattern before optimizing. Each unit cell consists of four vertices: two interior vertices (e.g. nr. 66 and 67 in Supp. Fig. 20) that are connected to two boundary vertices (nr. 68 and 69 in Supp. Fig. 20). The dihedral angle at the source vertex is set to 15° and the option of symmetric folding is used. In total, the pattern has 1900 vertices, 3165 edges and 1266 faces.



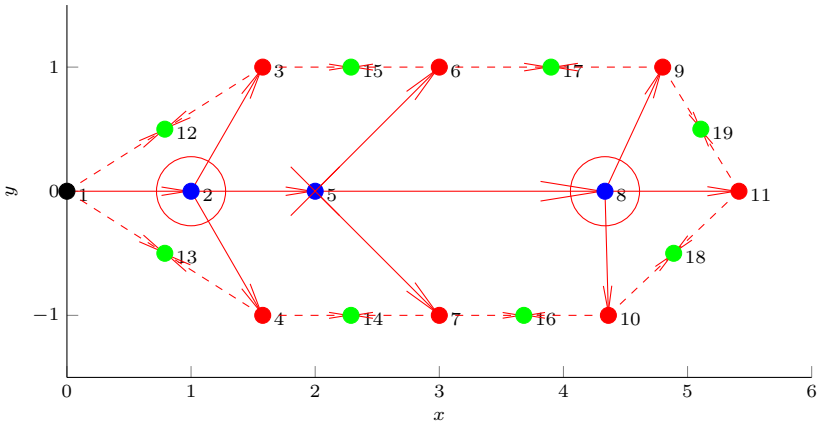
Supplementary Figure 18: Crease pattern graph of simple example before the optimization (initial solution).

Optimization Procedure The 317 unit cells are optimized one unit cell a time. The goal for each unit cell is that the first interior vertex of the next unit cell (nr. 70 in Supp. Fig. 20) reaches a target point. The target point is sampled from the "elephant curve" available on WolframAlpha [2]. The optimization procedure for the complete pattern is summarized in Supp. Alg. 4 and Supp. Fig. 21. In the following, the optimization that consists of two steps (Lines 2 and 3) applied to each unit cell are discussed in further detail.

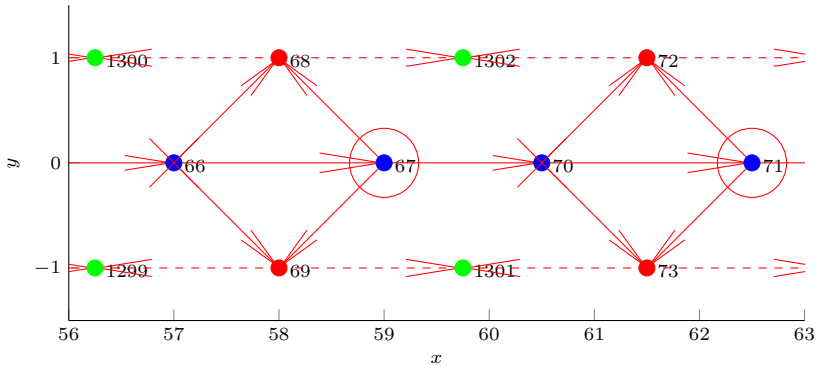
Supplementary Algorithm 4: Overview of the procedure to obtain a pattern that approximates the elephant curve.

Input: Base pattern (initial solution)
Input: Points sampled from the elephant curve
Result: Evaluated origami in the shape of the elephant curve
1 **for** *Each unit cell* **do**
2 | Find modes of interior vertices and rough shape of unit cell by enumeration;
3 | Use Supp. Algorithm 2 to optimize the unit cell;
4 **end**

- A preliminary step (Line 2) in which the rough shape of the unit cell and the modes of the two internal vertices are determined. To that end, the following options are considered:
  - Changing the geometry of the unit cell:
    - Keep the default geometry, as shown in Supp. Fig. 20
    - Move both boundary vertices by $\Delta x = 1.5$ ($\Delta y = 0$)
    - Move both boundary vertices by $\Delta x = -1.5$ ($\Delta y = 0$)
  - Change the modes of the two interior vertices. This results in $2^2 = 4$ possible mode combinations/kinematic paths (two possible modes per vertex).



Supplementary Figure 19: Graph of the simple example after the optimization.

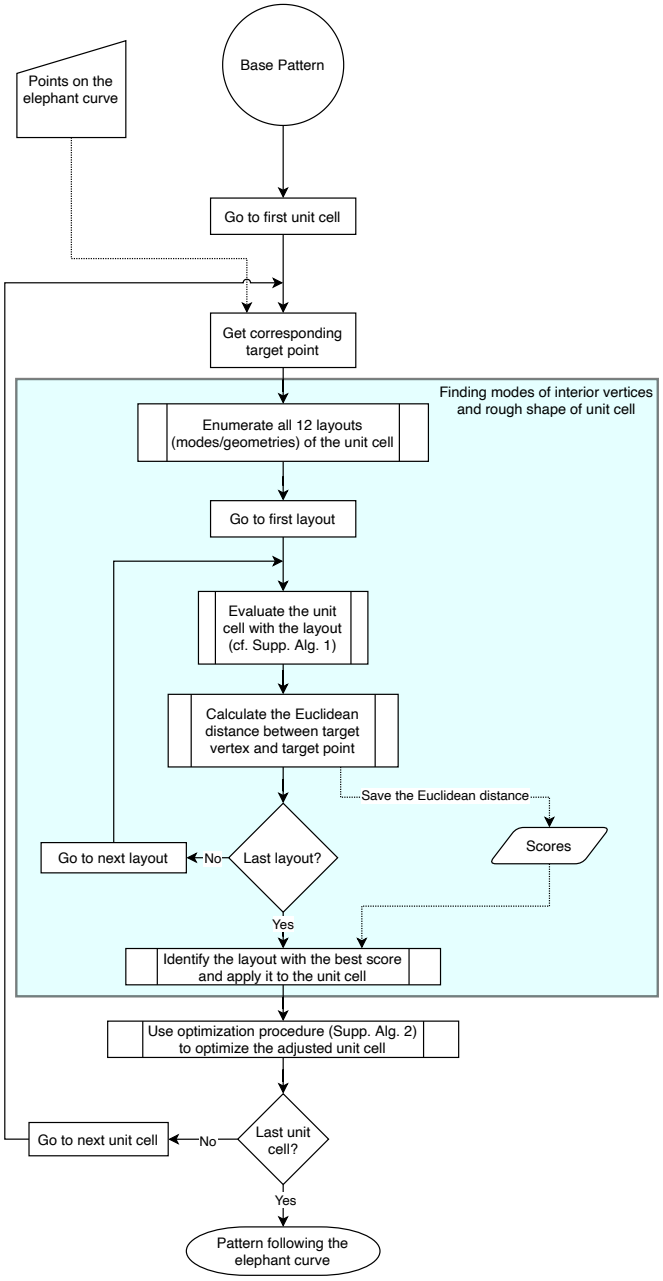


Supplementary Figure 20: Detail of the crease pattern graph of the example "elephant" before the optimization.

This results in 12 possible combinations which are enumerated (the increased computational cost over a more sophisticated optimization algorithm is expected to be minimal). Each combination (i.e. combination of geometric modification and mode assignments) is applied to the crease pattern graph and the distance between the target vertex and the target point for this unit cell is calculated. The combination where this distance is the lowest is chosen and applied to the crease pattern graph. The resulting graph forms the basis for the optimization step in L. 3.

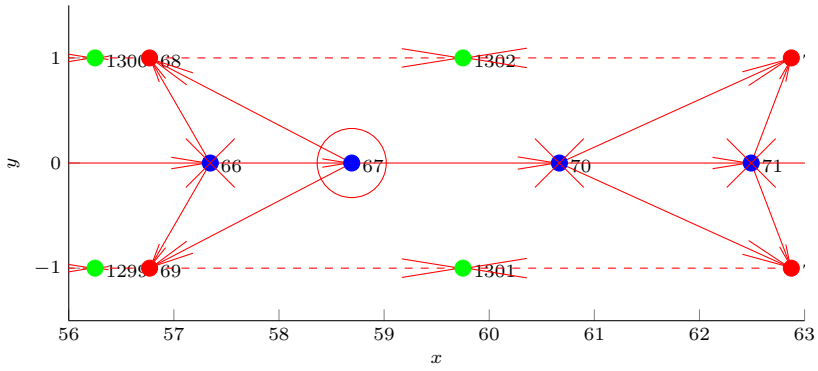
- Next, the locations of the vertices in the unit cell are optimized (Line 3). All four vertices of the unit cell as well as the target vertex are allowed to move, i.e. marked as mobile vertices. Furthermore, they are constrained to only change their x-coordinate and, to maintain the pattern symmetry, the x-coordinates of the two boundary vertices are linked (i.e. constrained to remain identical). The mobile vertices are permitted to change their x-coordinate by no more than ± 0.5 (i.e. $x_{\text{old}} - 0.5 \leq x_{\text{new}} \leq x_{\text{old}} + 0.5$).

On average, the optimization of one unit cell takes 22.0 s (the Euler cluster of ETH Zurich



Supplementary Figure 21: Procedure to obtain a pattern that approximates the elephant curve (Illustration of Supp. Alg. 4).

was used for these computations). This duration is comparable with the time to evaluate the folded configuration of the complete origami (22.5 s). A detail of the optimized pattern is shown in Supp. Fig. 22. The detail shows the same part of the crease pattern as the one shown in Supp. Fig. 20.



Supplementary Figure 22: Detail of the crease pattern graph of the example "elephant" after the optimization.

Regarding the runtimes, we notice that with larger crease patterns like this one, read/write operations affect the performance significantly. When evaluating a folded configuration on this example, less than 2 % or 0.45 s of the runtime are spent with the calculation of dihedral angles, direction vectors of edges, vertex positions and normal vectors, while approx. 75 % are spent writing these results to all affected variables (the remaining 23 % consist mainly of checking whether vertices can be evaluated, which is read-intensive).

Supplementary Note 4: Automated Pattern Generation

A simple pattern can be expanded by adding more vertices to it. We apply the expansion operation to boundary vertices and choose the locations of the added vertices such that the resulting origami reaches a predefined point. Supp. Algorithm 5 and Supp. Fig. 23 give an overview of the procedure for such an expansion step. In the following, the steps are explained in further detail.

Supplementary Algorithm 5: Procedure to expand an evaluated pattern.

Input: 1) Evaluated origami pattern
2) Point to be reached after expansion (target point)
3) List of boundary vertices that can be expanded and options for their expansion
Result: Expanded origami reaching the target point

```
1  for all expansion candidates (boundary vertices) do
2      for all expansion options specified for this vertex do
3          for all suboptions for this option do
4              Calculate the smallest possible distance to the target point;
5              Save this distance ("score") and the corresponding parameters;
6          end
7      end
8  end
10 Identify the option with the best (lowest) score;
12 Apply this expansion option with the corresponding parameters;
14 (optional) Optimize the expanded pattern;
```

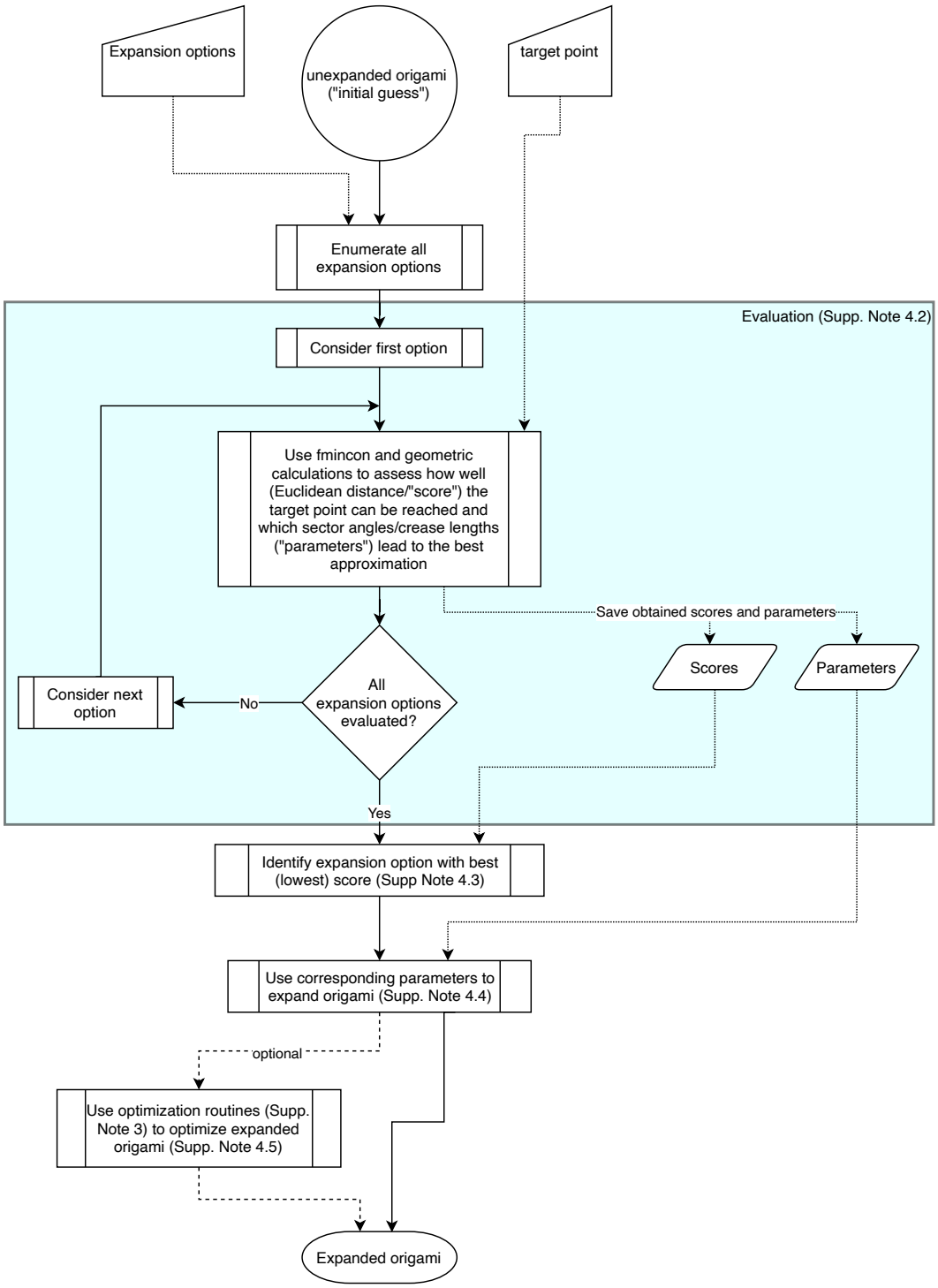
Supplementary Note 4.1: Input

Initial Pattern A (valid) origami that is fully evaluated is required. This origami will be expanded later on (Line 12).

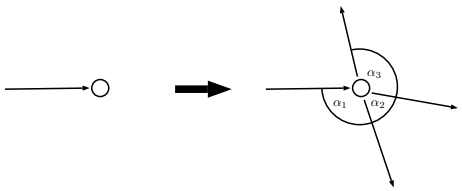
Target Point A point (i.e. its coordinates in 3D) is required. The goal of the expansion of the pattern is that a boundary vertex of the pattern will reach this point in the folded configuration.

Expansion Options A set of (i.e. at least one) boundary vertex is required that can serve as basis for the expansion. For each of these vertices, a specification of possible expansion options (at least one) is required. Possible expansion options include

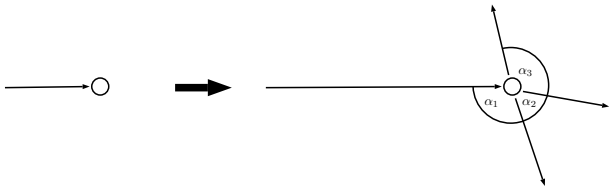
- Generic Expansion: Three outgoing edges are added to the vertex at arbitrary sector angles, as shown in Supp. Fig. 24. Any of the three created vertices can be the target vertex and reach the target point. The interior-point optimizer in MATLAB (fmincon) is used to find suitable sector angles α_{1-3}.
- Generic Expansion with Input Stretch: The vertex is moved along its input edge away from its direct predecessor (i.e. its input edge is stretched, cf. Supp. Fig. 25). Afterwards, a generic expansion is applied. The distance by which the input crease is stretched is a further variable in the optimization. Compared with regular generic expansion, this option may pose a viable alternative when interference with adjacent vertices poses a significant limitation on sector angles.



Supplementary Figure 23: Procedure to expand a pattern (Illustration of Supp. Alg. 5).



Supplementary Figure 24: Application of generic expansion option



Supplementary Figure 25: Application of option "generic expansion with input stretch"

Supplementary Note 4.2: Evaluation Phase

All expansion options are "evaluated" in Lines 1-5 on how well they can reach the target point. The specific steps for the evaluation of an expansion option depend on the expansion option:

Generic Expansion The target point can be approached with any of the three successors of the expanded vertex (i.e. three possible target vertices) and the expanded vertex can take 2 modes. This results in six suboptions (3 gripping vertices $\times$ 2 modes) for the for-loop in Line 3.

To obtain the score in line 4, a geometric calculation is used. For any choice of sector angles α_{1-3}, the crease vector $\vec{n}_d$ of the output edge at whose end the target vertex is located is calculated using the procedure described in Supp. Note 2.2. Based on this vector and the vector between the vertex to expand and the target point $\vec{v}_c$, the minimal reachable Euclidean distance d ("score") can be calculated to (cf. Supp. Fig. 26):

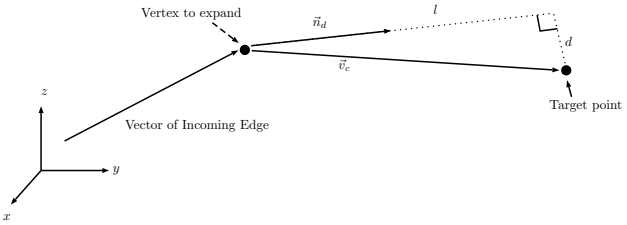
$$d = \|\vec{v}_c - (\vec{v}_c \cdot \vec{n}_d)\vec{n_d}\| \tag{12}$$

The length l between the expanded vertex and the target vertex is given by (cf. Supp. Fig. 26):

$$l = \vec{v}_c \cdot \vec{n}_d \tag{13}$$

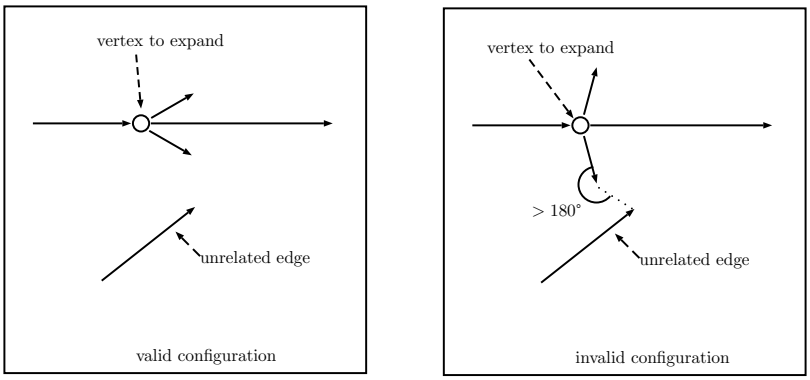
The interior-point optimizer in MATLAB (fmincon) is used to find the sector angles α_{1-3} that minimize d. The solver was chosen because it adheres well with our optimization model formulation. During the optimization, the following constraints need to be considered:

- Neither the first nor the last outgoing edge are allowed to interfere with other edges and vertices as shown in Supp. Fig. 27.
- All sector angles (including $360° - \alpha_1 - \alpha_2 - \alpha_3$) are between 10° and 170°.
- The absolute values of all dihedral angles should be above 10° (avoid trivial folds) and below 150° (avoid self-collisions). This is enforced by adding a penalty to the score if such angles are observed.
- The lengths l may not be negative (enforced by penalization)



Supplementary Figure 26: Calculation of d and l.

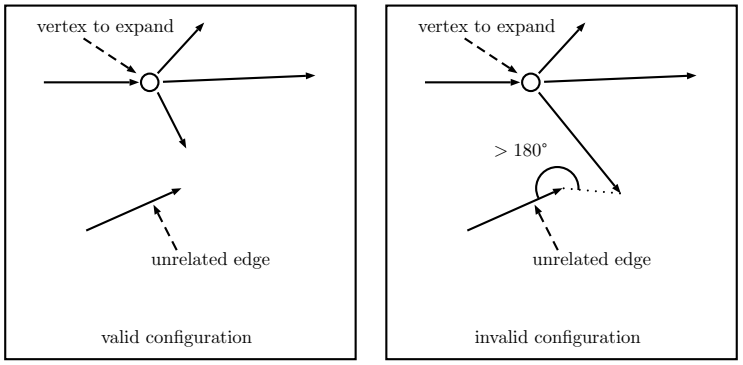
- If a length l results that leads to an interference as shown in Supp. Fig. 28, the score is penalized.



Supplementary Figure 27: Valid and invalid sector angle configuration due to interference with other edges of the same crease pattern graph. The invalid configuration contains a boundary vertex with a sector angle that is larger than 180°. Such vertices are not desired since they render an expansion of the boundary vertex impossible.

All optimizations are started with an initial solution of $\alpha_1 = 120°, \alpha_2 = \alpha_3 = 60°$. If the optimization does not yield a solution with a score $d \leq 1 \times 10^{-2}$, it is restarted with the initial solution $\alpha_1 = 60°, \alpha_2 = \alpha_3 = 120°$ and the previously result is overwritten if this initial guess results in a better solution (lower d). The optimization termination criteria are given by the default criteria of fmincon (objective function value below 1×10^{-6}, more than 3000 objective function evaluations or 400 iterations, step size below 1×10^{-10}). For each optimization, the obtained sector angles, the lengths l and the score d are saved (Line 4 and 5).

Generic Expansion with Input Stretch This expansion option allows for the same six suboptions as generic expansion. While for the generic expansion, only the three sector angles (α_{1-3}) are considered as optimization variables, the length by which the input edge is elongated Δl constitutes a fourth optimization variable in this case. Even though most of the optimization is carried out analogously to the case of a generic expansion, the prevention of interference is enforced differently in this case. While for the generic expansion, appropriate bounds for the sector angles could be calculated that prevented interference, this is no longer the case here. This is because these bounds vary as the vertex is moved along its input edge. Therefore, a nonlinear constraint is used to prevent interference. The fourth optimization variable (the stretch of the



Supplementary Figure 28: Valid and invalid configuration. In the invalid configuration, the edge is too long and results in a sector angle of more than 180°, which is not desired.

input crease) has a lower bound of 0 and its upper bound is determined such that sector angles of more than 180° are avoided at adjacent vertices.

Supplementary Note 4.3: Identification of Best Option

The task of identifying the best option (Line 10) consists of finding the minimum of all scores and the corresponding boundary vertex, expansion option and suboption.

Supplementary Note 4.4: Expansion of Pattern

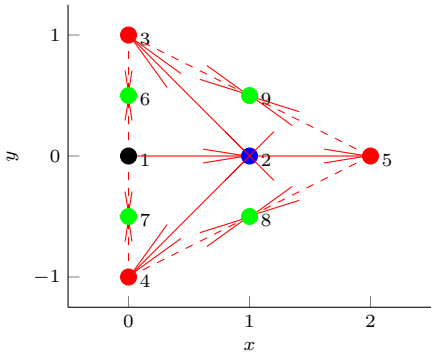
Based on the identified expansion option (and suboption), the respective parameters that were calculated in line 5 are used in line 12 to expand the crease pattern graph. To expand the pattern, first the faces adjacent to the vertex to expand are deleted. This involves the deletion of all auxiliary vertices on their boundaries. Next, the new vertices and edges are added. Finally, faces are (re)created around the expanded vertex, which involves the addition of auxiliary vertices and boundary edges.

Supplementary Note 4.5: Optimization of Expanded Pattern

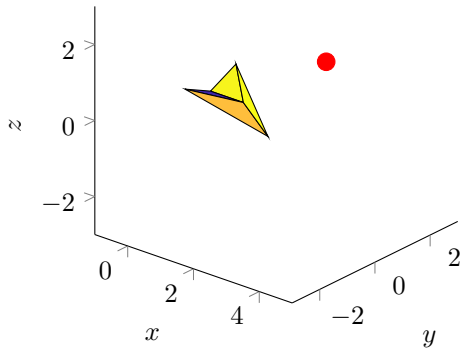
After expanding the crease pattern graph, a pattern optimization (cf. Supp. Note 3) can be carried out (Line 14). The optimization problem is defined during the expansion of the pattern. This is done by marking all added vertices other than auxiliary vertices as mobile vertices and the target vertex as such. Furthermore, inequality constraints are specified to prevent interference as shown in Supp. Figs. 27 and 28. Solving this optimization problem can help bringing the target vertex closer to the target point in certain cases. However, the parameters obtained during the evaluation step usually yield a very good initial guess. Furthermore, optimizations during the evaluation step are more efficient than the pattern optimization routines (Supp. Alg. 2), since the involved read/write operations are less expensive.

Supplementary Note 4.6: Simple Example

In the example shown in Fig. 2D of the main text, the goal is to expand the crease pattern shown in Supp. Fig. 29 such that it reaches the marked red point in the folded configuration. The base

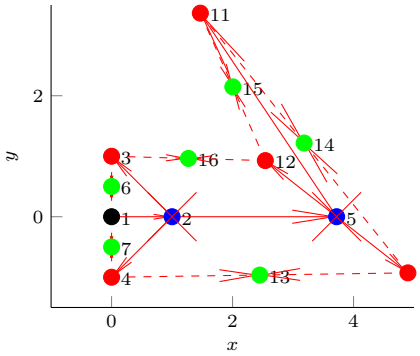


(a) Crease Pattern Graph

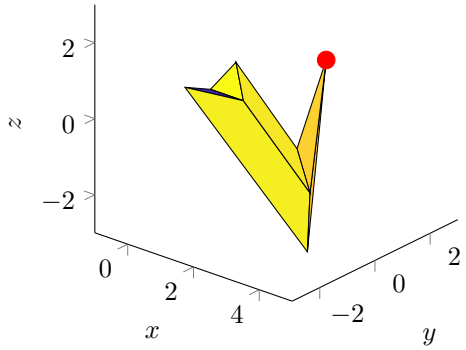


(b) Folded Configuration

Supplementary Figure 29: Graph and folded configuration of the base pattern of a simple example demonstrating the algorithm to extend crease pattern graphs.

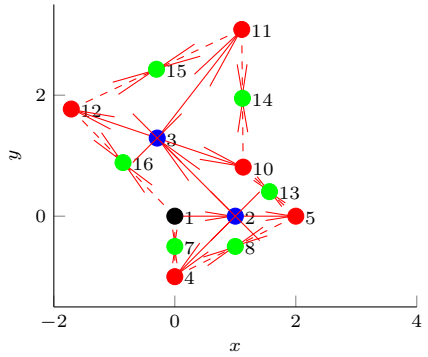


(a) Crease Pattern Graph

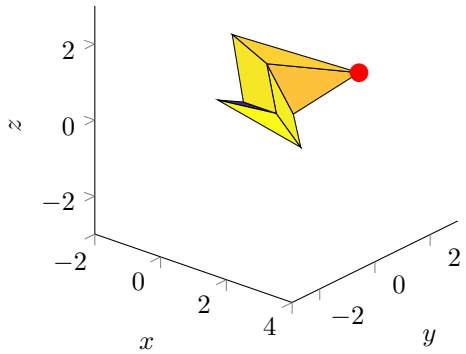


(b) Folded Configuration

Supplementary Figure 30: Graph and folded configuration of the expanded pattern.

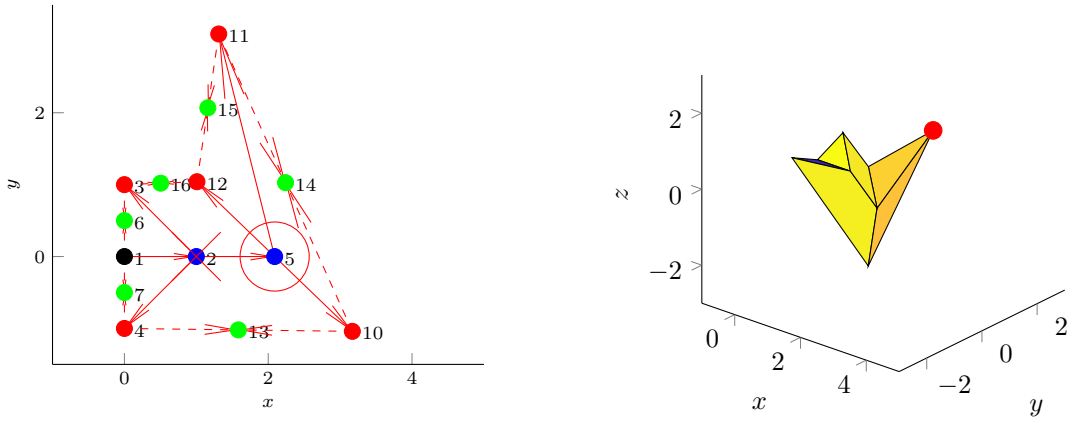


(a) Crease Pattern Graph



(b) Folded Configuration

Supplementary Figure 31: Graph and folded configuration of the expanded pattern, first alternative.



(a) Crease Pattern Graph

(b) Folded Configuration

Supplementary Figure 32: Graph and folded configuration of the expanded pattern, second alternative. The pattern strongly resembles the one in Supp. Fig. 30 with the key difference that the mode of vertex nr. 5 is switched. Another difference (though much less notable) is that in the pattern in Supp. Fig. 30, the sector angles around the incoming edge to vertex 5 sum to (very slightly) less than 180° while they sum to (very slightly) more than 180° in this pattern. If the sum of sector angles around all incoming edges crosses 180°, this leads to a sharp change in the folding kinematics, which can be avoided by switching the mode of the vertex.

pattern consists of a single symmetric vertex. To obtain the folded configuration, the option of symmetric folding is turned on and a dihedral angle of 45° is specified for the input edge.

For the expansion, vertices 3, 4 and 5 are specified as candidates and generic expansion with input stretch as only expansion option. This results in 18 possible expansions (3 expansion vertices, 2 modes per vertex, 3 possible target vertices). Of the 18 options, three reached a score of 1×10^{-6} or less. On average, one optimization is completed in 0.4 s (on a 2.3 GHz Intel i9 processor). Of these three options, the one shown in Supp. Fig. 30 was chosen since the score it reached, 5.69×10^{-9}, was the best reached by any optimization. This choice is random to some extent as in all three cases, the optimization was terminated because the objective function reached 1×10^{-6} and the other two options represent valid choices as well. They are shown in Supp. Figs. 31 and 32. In accordance with the scores reached during the evaluation step, the generated designs reach the target point with near-perfect precision. Further optimization is unnecessary and terminates immediately as the objective function value is below the predefined threshold for the initial solution.

Supplementary Note 4.7: Example "Tennis Ball"

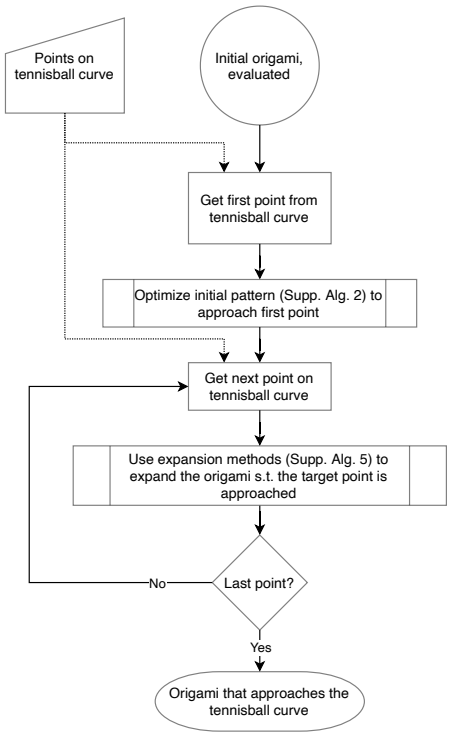
The generation of the pattern shown in Fig. 2E in the main text, which approaches the curve of a seam on a tennis ball or baseball, is based on 38 points sampled from a curve [3] approximating such a shape. The process to generate this pattern is summarized in Supp. Alg. 6 and Supp. Fig. 33.

The pattern generation starts with the crease pattern graph shown in Supp. Fig. 34. Its folded configuration is obtained by using symmetric folding and a dihedral angle of 45° at its input edge. The first target point is approximated by vertex 5, using crease pattern graph optimization (cf. Supp. Note 3). For this optimization, vertices 3, 4 and 5 were used as mobile vertices and

Supplementary Algorithm 6: Procedure to generate an origami whose folded configuration approaches the tennis ball curve

Input: Initial origami, evaluated
Input: Points sampled from the tennis ball curve (target points)
Result: Evaluated origami whose folded configuration approaches the shape of the tennis ball curve

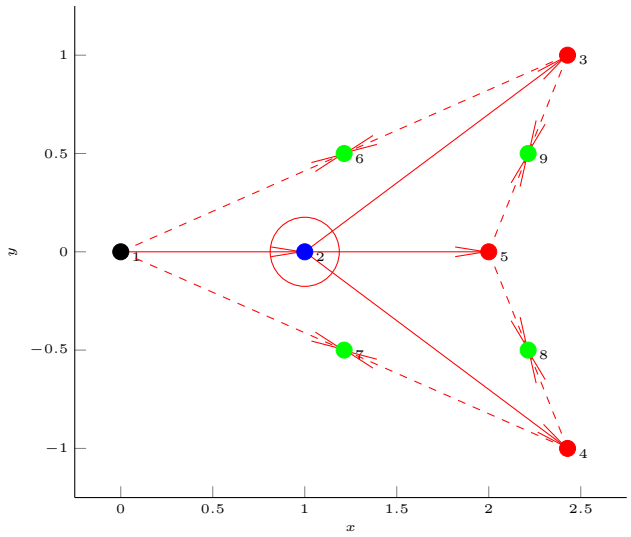
Use pattern optimization (Supp. Alg. 2) to approach the first target point;
for *second target point to last target point* **do**
    Use pattern expansion (Supp. Alg. 5) to expand the origami s.t. the target point is approached;
end



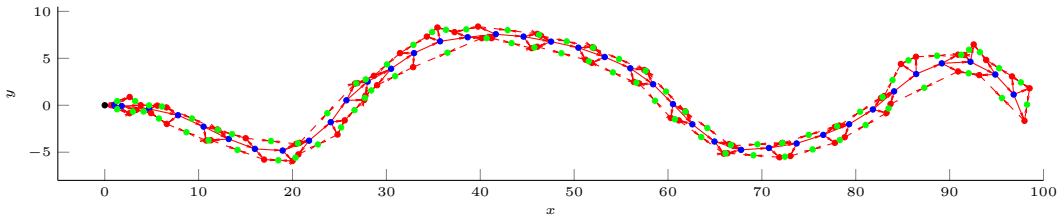
Supplementary Figure 33: Procedure to generate an origami whose folded configuration approaches the tennis ball curve (Illustration of Supp. Alg. 6).

were allowed to move within a square with side length 1.1 centered around their original position.

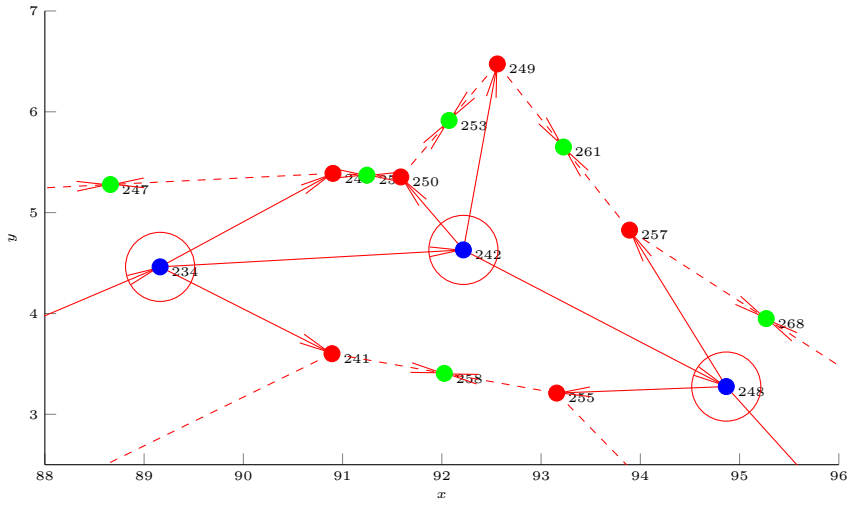
From then on, each point is approximated with a further expansion of the pattern. Whichever vertex was used to approximate the previous point is expanded using the option generic expansion. After application of an expansion, a pattern optimization is carried out to improve the pattern. However, this optimization usually terminates after evaluating the initial solution that was built based on the results of the evaluation step. On a 2.3 GHz Intel Core i9 processor, the pattern generation is completed in 56 s. The final pattern shown in Supp. Fig. 35 has 194 vertices, 271 edges and 78 faces.



Supplementary Figure 34: Initial crease pattern graph for the generation of an origami that approaches the tennis ball curve in its folded configuration.



(a) Complete crease pattern graph. For clarity, no vertex index labels and mode markers are shown.



(b) Detail from Supp. Fig. 35a.

Supplementary Figure 35: Resulting crease pattern graph that approaches the tennis ball curve in its folded configuration.

Supplementary Note 5: Quadrilateral Patterns

In this section, the quadrilateral-closing operation and the extrusion scheme are explained and the examples presented in the main text are detailed.

Supplementary Note 5.1: Quadrilateral-Closing Operation

A summary of the implementation of the quadrilateral-closing operation is given in Supp. Alg. 7 and illustrated in Supp. Figs. 36 and 37. The implementation is based on the calculation by Feng et al. [4]. However, that work uses different conventions for sector angles and modes than here. This complicates the application of that calculation (Line 9 in Supp. Alg. 7) somewhat as the input and output quantities need to be converted between the two conventions with dedicated methods (Supp. Algs. 8 (illustrated in Supp. Fig. 38) and 9 (illustrated in Supp. Fig. 39)).

Supplementary Algorithm 7: Implementation of Quadrilateral-Closing operation.

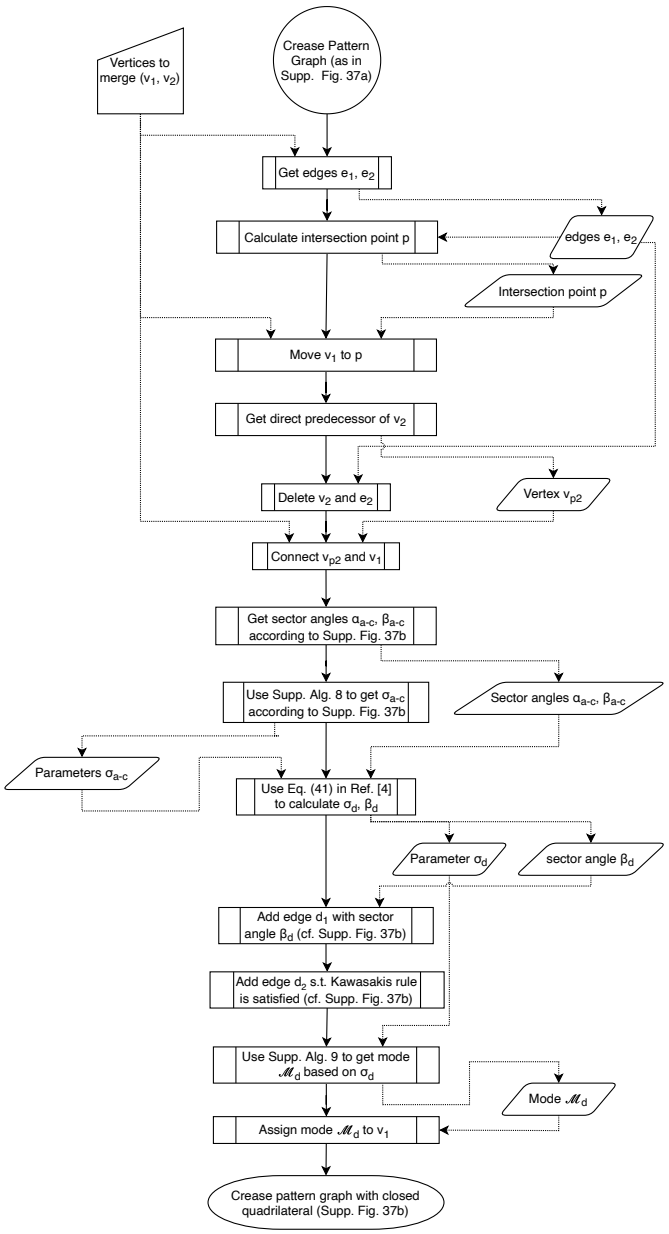
Input: Indices of vertices to merge v_1, v_2 (dark gray in Supp. Fig. 37a)
Result: Quadrilateral-closing operation applied

1 $e_1, e_2 \leftarrow$ Incoming edges to v_1, v_2;
2 $\vec{p} \leftarrow$Intersection point of e_1 and e_2;
3 Move v_1 to $\vec{p}$;
4 $v_{p2} \leftarrow$ Direct predecessor of v_2;
5 Delete v_2 and e_2;
6 Connect v_{p2} to v_1;
7 $\alpha_{a-c}, \beta_{a-c} \leftarrow$ Obtain the sector angles of the formed quadrilateral according to Supp. Fig. 37b.;
8 $\sigma_{a-c} \leftarrow$ Use Supp. Alg. 8 to obtain σ_{a-c} of the involved vertices.;
9 $\sigma_d, \beta_d \leftarrow$ Calculate using [4, Eq. (41)];
10 Add edge d_1 based on β_d;
11 Add edge d_2 such that Kawasaki's rule is satisfied;
12 $\mathcal{M}_d \leftarrow$ Use Supp. Alg. 9 to get mode for quadrilateral-closing vertex from σ_d.;
13 Assign $\mathcal{M}_d$ to quadrilateral-closing vertex.;

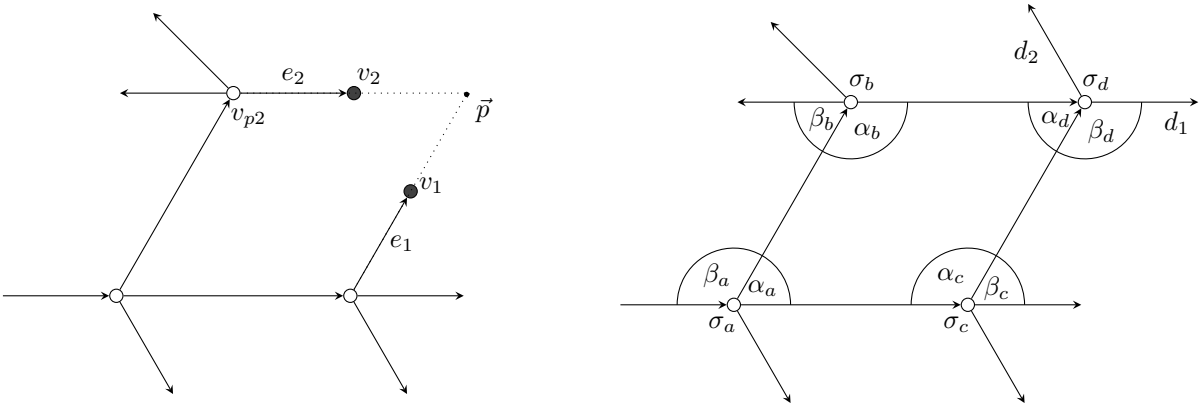
Supplementary Algorithm 8: Getting a σ-value for a degree-4-vertex.

Input: index of vertex
Output: σ for the vertex

1 Use Supp. Alg. 10 to get signs of dihedral angles of all edges incident to vertex;
2 $\Sigma_1 \leftarrow$ sign of dihedral angle of edge that is adjacent to the α-sector but not to the β-sector;
3 $\Sigma_2 \leftarrow$ sign of dihedral angle edge that is adjacent to the β-sector but not to the α-sector;
4 **if** $\Sigma_1 \neq \Sigma_2$ **then**
5 $\quad \sigma = +$
6 **else**
7 $\quad \sigma = -$
8 **end**



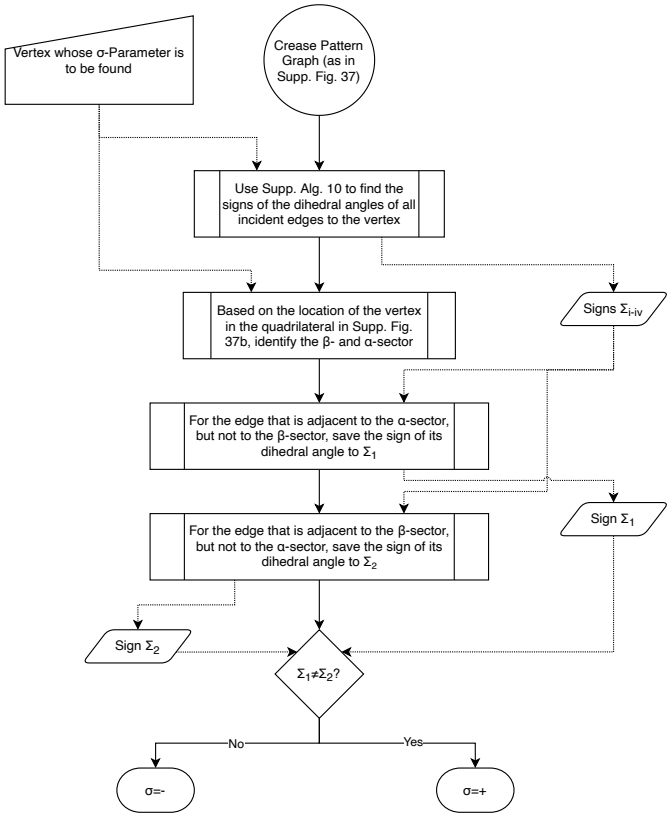
Supplementary Figure 36: Implementation of Quadrilateral-Closing operation (Illustration of Supp. Alg. 7).



(a) Before

(b) Afterwards

Supplementary Figure 37: Application of the quadrilateral-closing operation.



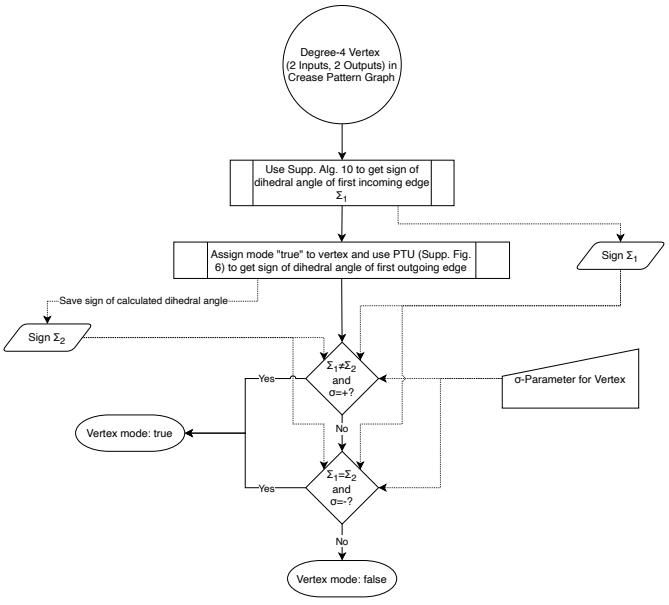
Supplementary Figure 38: Getting a σ-value for a degree-4-vertex (Illustration of Supp. Alg. 8).

Supplementary Algorithm 9: Getting the mode of a vertex based on a σ-value.

Input: 1) index of vertex (must be degree 4; 2 inputs and 2 outputs)
2) σ-value

Output: Mode for the vertex

$\Sigma_1 \leftarrow$ Use Supp. Alg. 10 to get sign of dihedral angle of first incoming edge;
Apply mode true to the vertex;
$\Sigma_2 \leftarrow$Use PTU to get the sign of the dihedral angle of the first outgoing edge;
if $(\sigma = +)$ *and* $(\Sigma_1 \neq \Sigma_2)$ **then**
  return mode=true
else if $(\sigma = -)$ *and* $(\Sigma_1 = \Sigma_2)$ **then**
  return mode=true
else
  return mode=false
end

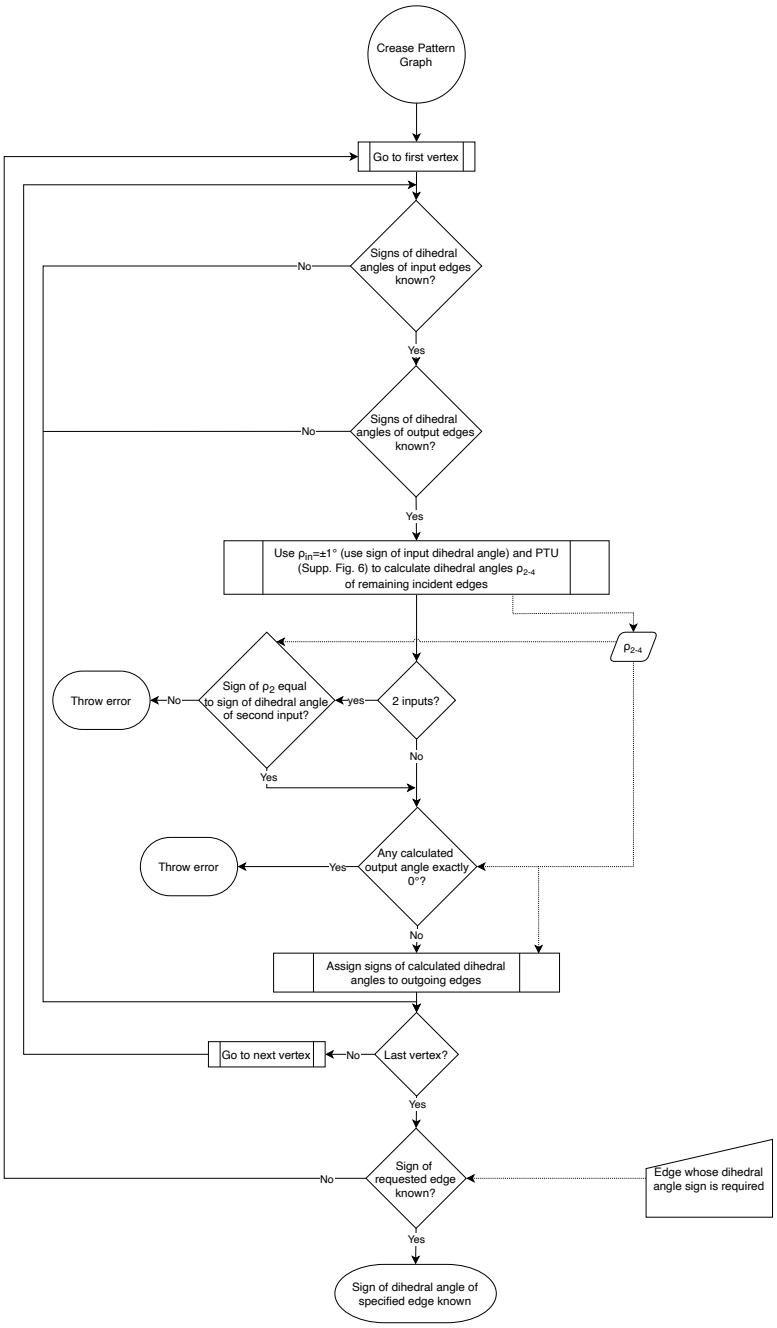


Supplementary Figure 39: Getting the mode of a vertex based on a σ-value (Illustration of Supp. Alg. 9).

Supplementary Algorithm 10: Getting the sign of a dihedral angle of an edge. The algorithm is illustrated as flow chart in Supp. Fig. 40.

Input: Index of edge e whose dihedral angle sign is needed
Output: Sign Σ_e of the dihedral angle of the specified edge

while Σ_e *not known* **do**
  for *all vertices* v_j *in crease pattern graph* **do**
    if *Signs of dihedral angles of all input and output edges known* **then**
      continue;
    end
    if *Signs of dihedral angles* Σ_{in} *all input edges to* v_j *known* **then**
      $\rho_{\text{in}} \leftarrow 1^\circ \cdot \Sigma_{\text{in},1}$;
      ρ_{2-4} Use the PTU to get the dihedral angles of the remaining incident edges;
      if v_j *has two inputs* **then**
        if $\Sigma_{in,2} \neq sgn(\rho_2)$ **then**
          throw error;
        end
      end
      for *all output edges* e_k *of* v_j **do**
        if *Corresponding previously calculated dihedral angle* $\rho_k > 0^\circ$ **then**
          Assign $\Sigma = +$ to e_k;
        else if *Corresponding previously calculated dihedral angle* $\rho_k < 0^\circ$ **then**
          Assign $\Sigma = -$ to e_k;
        else
          throw error;
        end
      end
    end
  end
end



Supplementary Figure 40: Getting the sign of a dihedral angle of an edge (Illustration of Supp. Alg. 10).

Supplementary Note 5.2: Layer-Adding Procedure

Based on the quadrilateral-closing operation, we introduce the layer-adding procedure. This procedure adds a layer to an existing pattern given a vector with all vertices that currently form the pattern boundary that is to be expanded. The algorithm is summarised in Supp. Alg. 11 and illustrated in Supp. Figs. 41 and 42.

Supplementary Algorithm 11: Layer-adding procedure

Input: 1) Vector $\mathcal{V}$ with vertices on boundary of pattern (gray in Supp. Fig. 42)
2) Parameters for expansion of first vertex in $\mathcal{V}$ (black in Supp. Fig. 42)
Result: Layer added to pattern

```
Move V_1 along its input edge to desired position;
Expand V_1;
Assign desired mode to V_1;
v_n ← first successor of V_1;
for v_b ← V_2 ··· V_end do
    Apply quadrilateral-closing operation (Supp. Alg. 7) with v_n and v_b;
    v_c ← vertex that closes the formed quadrilateral;
    v_n ← First successor of v_c;
end
```

Supplementary Note 5.3: Evaluation of Interior Vertices

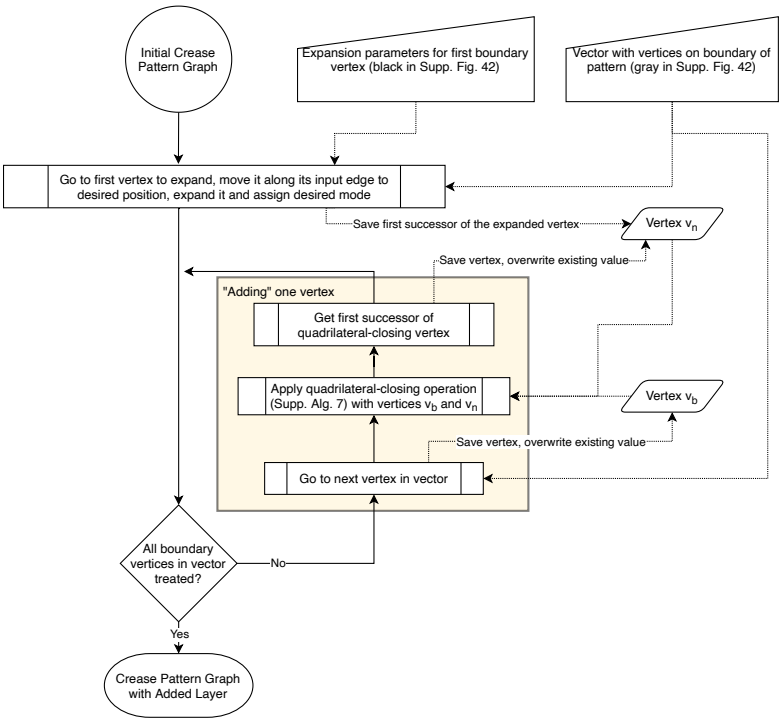
After applying the quadrilateral-closing operation, a vertex of degree four with two inputs and two outputs results. Such a vertex represents an exception to the rule that each interior vertex must have exactly three outputs and may only be created with a quadrilateral-closing operation.

The requirements for evaluating an interior vertex with two outputs and the evaluation procedure are similar to the procedure described in Supp. Note 2.2. For the calculation of the dihedral angles, the first input is used to calculate the dihedral angles of the other three edges, one of them corresponding to the second input. The comparison with the previously calculated dihedral angle for this edge is used as validation of the calculation of the dihedral and sector angles.

Supplementary Note 5.4: Enumeration of Kinematic Paths

To enumerate the kinematic paths for quadrilateral patterns, the following changes need to be considered as compared to "regular" patterns (Supp. Note 2.5):

- The modes of quadrilateral-closing vertices are automatically assigned and cannot be changed. The upper bound for the number of kinematic paths is therefore 2^m where m is the number of vertices whose mode can be determined by the user during construction of the pattern.
- Since the sector angles and the mode of the quadrilateral-closing vertex depends on the modes of the other involved vertices (cf. Supp. Alg. 7), using different modes when constructing the pattern generally leads to different crease patterns. When evaluating a possible mode combination, a crease pattern geometry that differs from the original pattern automatically corresponds to an invalid kinematic path ("does not count").



Supplementary Figure 41: Layer-adding procedure (Illustration of Supp. Alg. 11).

- Foldability of all patterns is always guaranteed. Therefore, if a mode combination yields the same crease pattern as the original pattern, it can be "counted" as valid kinematic path without evaluating the interval of valid input dihedral angles.

Supplementary Note 5.5: Examples for Quadrilateral Patterns

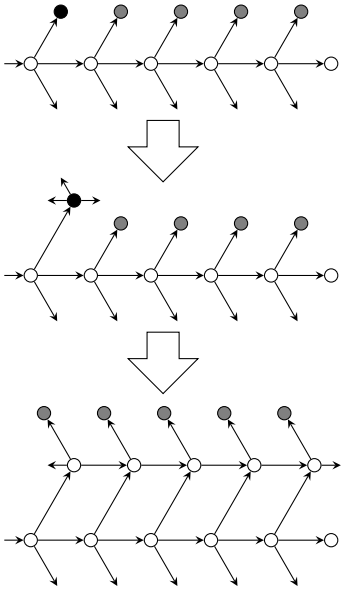
Yoshimura's Pattern (Fig. 3D in the main text) The full crease pattern graph is shown in Supp. Fig. 43. The horizontal "layers" are spaced a distance of 3 apart. The diagonal edges have an angle of 30° to the horizontal. The folded configuration shown in Fig. 3D of the main text is obtained using a dihedral angle of 50° at the input edge and symmetric folding (Supp. Note 2.1).

As shown in Supp. Fig. 43b and 43c, some vertices are spaced very closely. These vertex couples are referred to as "double vertices". This technique is used to represent vertices that, when considering a length scale that is much larger than the distance between the two vertices, appear as vertices of degree 6.

Huffman's Grid (Fig. 3E of the main text) The crease pattern graph is shown in Supp. Fig. 44. In this example, the "seed layer", i.e. the sequence of central outgoing edges starting from the source vertex is diagonal rather than horizontal (it ends with vertex with index 16 in Supp. Fig. 44).

All faces in the pattern are identical. Exemplified on the quadrilateral formed by vertices 7, 10, 20 and 22, the parameters for this example were chosen such that the quadrilaterals have the following geometry:

- Sector angle of 70° at vertex 7



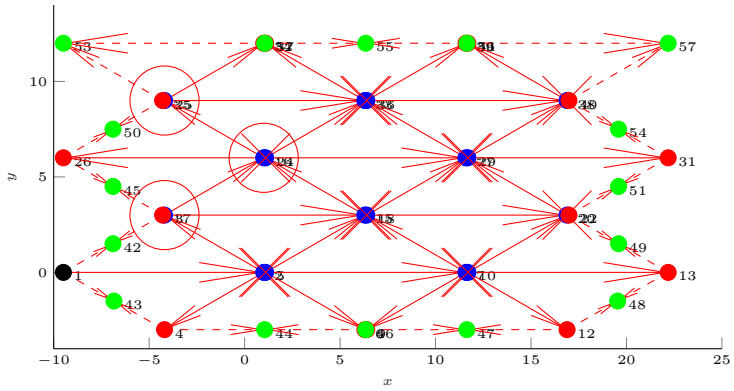
Supplementary Figure 42: Adding a layer to an existing pattern. The middle figure illustrates the situation after Line 2 in Supp. Alg. 11.

- Sector angle of 30° at vertex 10
- Length 3 for the crease between vertices 7 and 20
- Length 1.2 for the crease between vertices 20 and 22

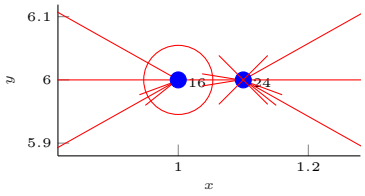
The deformed configuration shown in the main text is obtained by fixing the face defined by vertices 1, 2, 6, 18 and 127 and specifying a dihedral angle of 35° at the input edge.

Square Twist (Fig. 3F in the main text) The pattern (along with some variants) is documented in [5]. The corresponding crease pattern graph is shown in Supp. Fig. 45. The side length of the tilted squares is 1.5 (e.g. the length of the crease between vertices 2 and 7), the tilt angle (i.e. the angle between the edge from vertex 2 to vertex 7 and the horizontal) 60° and the distance between the tilted squares (e.g. the length of the crease between vertices 7 and 10) 0.3. To obtain the folded configuration shown in Fig. 3F of the main text, the face with vertices 1,2,6 and 46 was fixed and a dihedral angle of 90° was specified for the input crease (i.e. the one from vertex 1 to vertex 2).

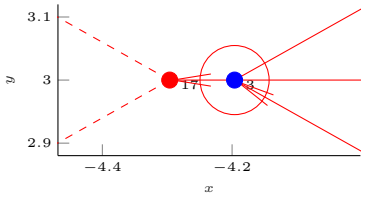
Waterbomb Pattern (Fig. 3G in the main text) The crease pattern graph is shown in Supp. Fig. 46. Like for Yoshimura's pattern, double vertices were often used to model vertices that appear as degree-6 vertices on the scale of the complete pattern. The horizontal layers are spaced a vertical distance of 1 apart and all diagonal edges form an angle of 45° with the x-axis. The folded configuration shown in Fig. 3G of the main text was obtained by using symmetric folding and a dihedral angle of $-120°$, i.e. a mountain crease, for the input edge.



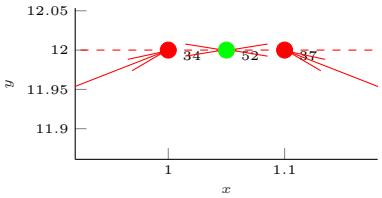
(a) Complete graph. Overlapping indices are caused by very short distances between vertices (cf. Supp. Figs. 43b to 43d)



(b) Detail from Supp. Fig. 43a



(c) Detail from Supp. Fig. 43a



(d) Detail from Supp. Fig. 43a

Supplementary Figure 43: Crease pattern graph of Yoshimura's pattern

Supplementary Note 5.6: Approximating Piecewise Linear Curves

We present a method to directly calculate the parameters (lengths l, sector angles α and vertex modes) of an origami strip as shown in Supp. Fig. 48 such that the strip approximates a piecewise linear curve, as shown in the example in Supp. Fig. 47, with its central crease when its input edge is folded by a user-defined dihedral angle of ρ.

The lengths l can be taken directly from the piecewise linear curve. The problem is thus reduced to calculating the sector angle α and the mode of the vertex from the fold angle φ and the dihedral angle ρ at each vertex of the piecewise linear curve. Regarding the dihedral angle ρ, we notice that due to the geometry of the interior vertices, its absolute value remains constant throughout the complete strip while its sign switches at each vertex.

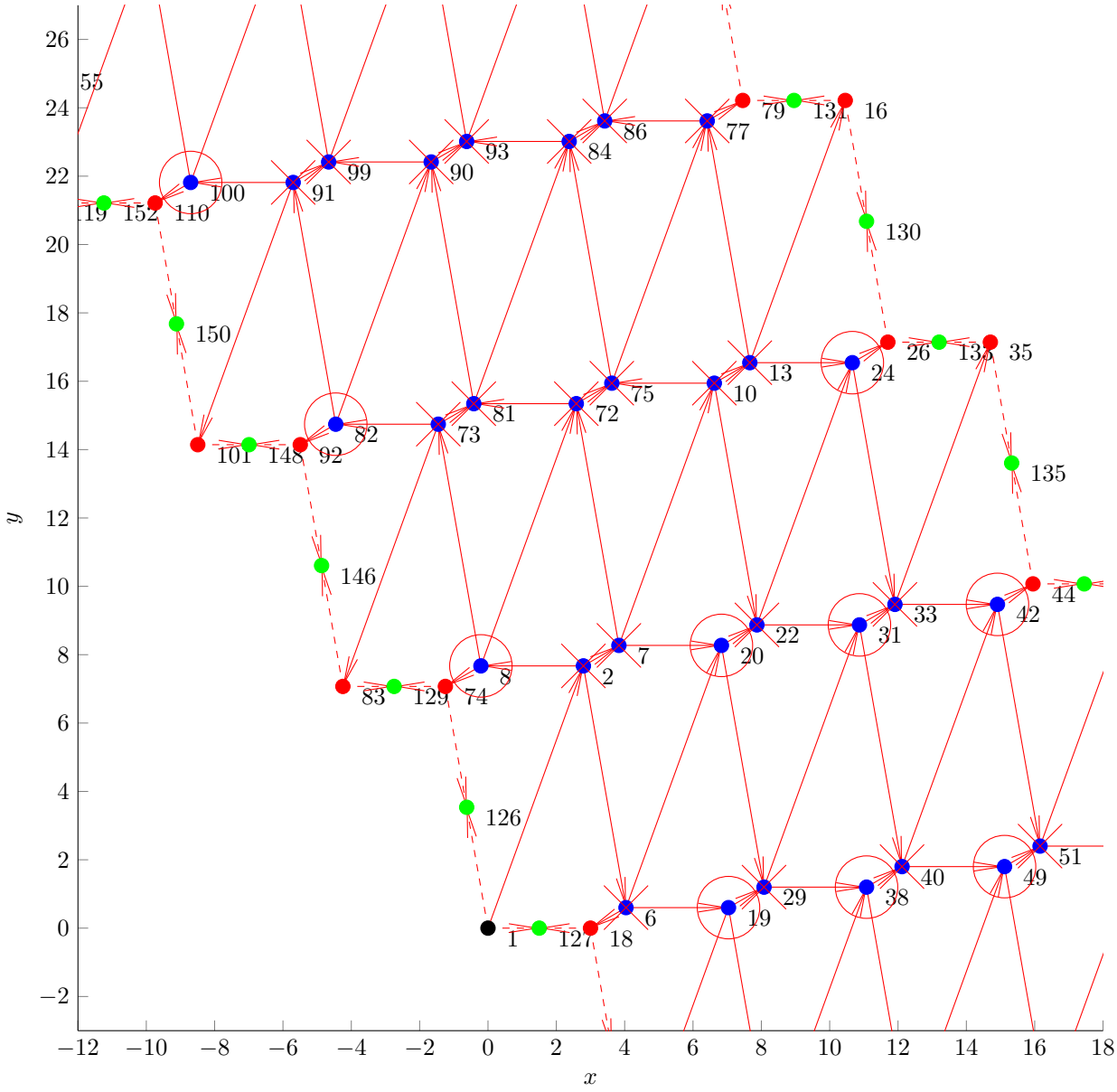
An arbitrary vertex is depicted in Supp. Fig. 49. The sector angle α is calculated by first calculating the angle γ using the spherical law of cosines:

$$\cos\gamma = \sin\left(\frac{\rho}{2}\right)^2 - \cos\left(\frac{\rho}{2}\right)^2 \cos(\varphi) \tag{14}$$

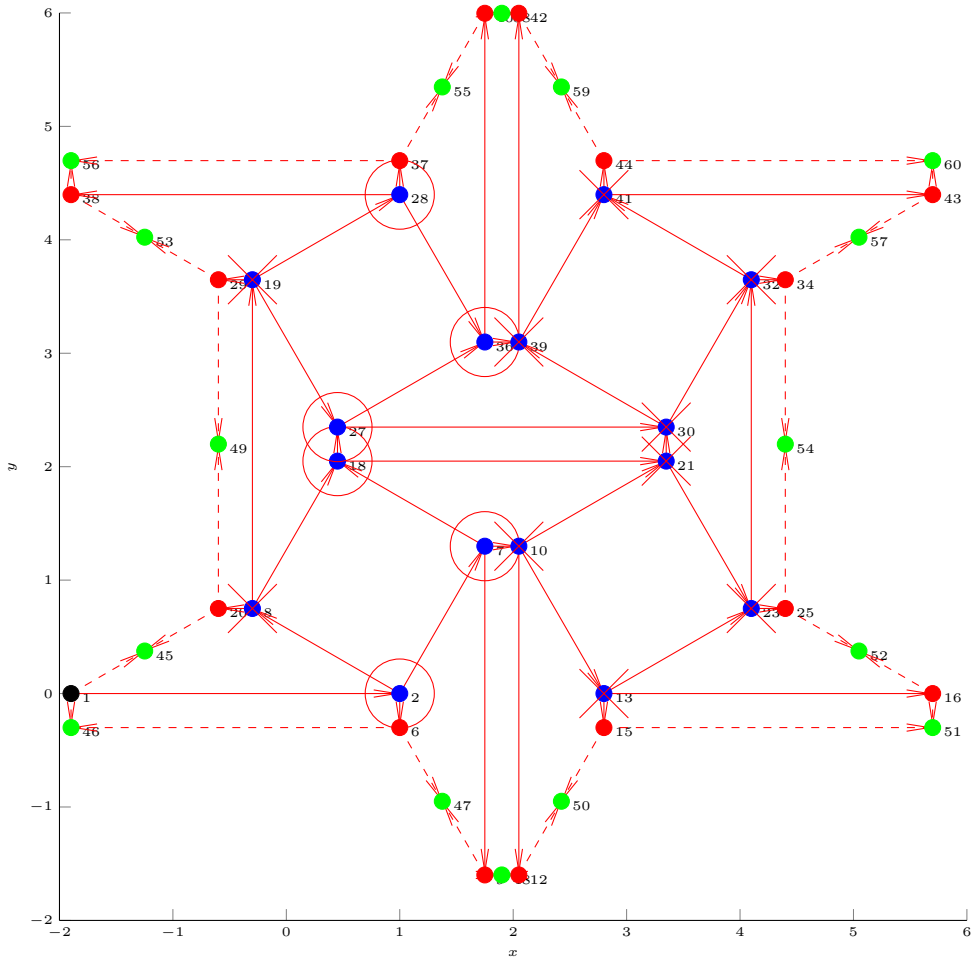
Using Eq. (14) and the spherical law of sines, α is calculated to:

$$\sin(\alpha) = \cos\left(\frac{\rho}{2}\right) \frac{\sin(\varphi)}{\sin(\gamma)} \tag{15}$$

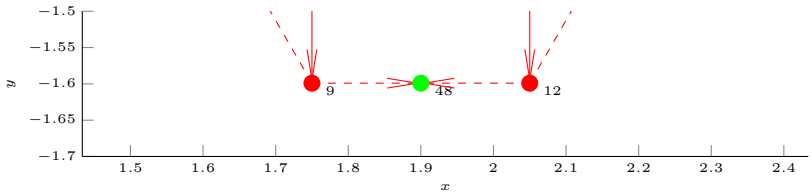
Equivalent derivations were used to obtain the expressions and results in Supp Tab. 2 where all combinations of the signs of φ and ρ are considered (also cf. [6]).



Supplementary Figure 44: Crease pattern graph of Huffman's grid

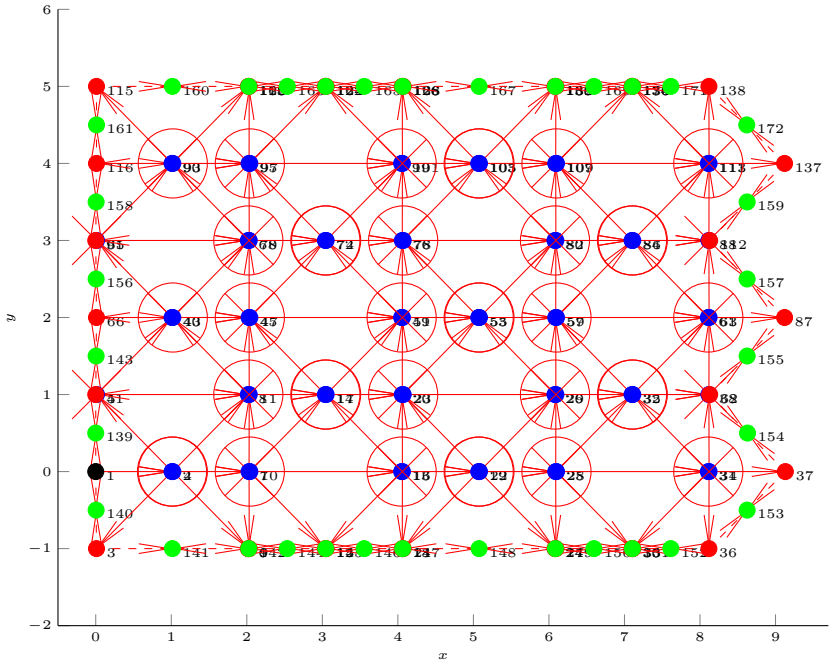


(a) Full Graph.

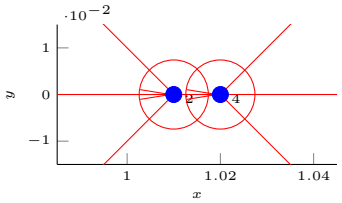


(b) Detail from Supp. Fig. 45a.

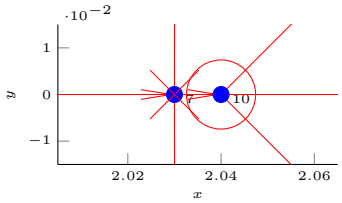
Supplementary Figure 45: Crease pattern graph of square twist pattern



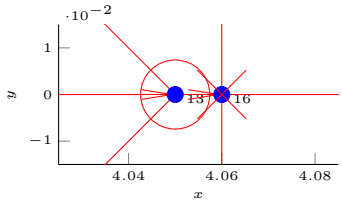
(a) Complete graph. Overlapping indices are caused by very close distances between vertices (cf. Supp. Figs. 46b to 46g).



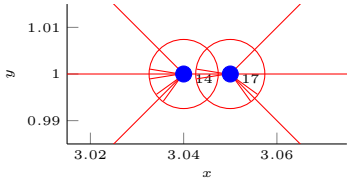
(b) Detail from Supp. Fig. 46a



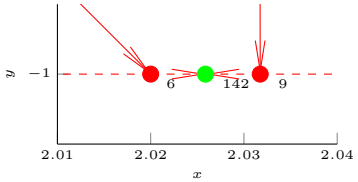
(c) Detail from Supp. Fig. 46a



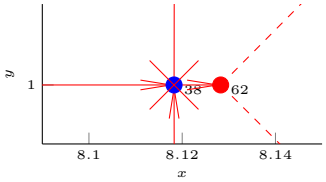
(d) Detail from Supp. Fig. 46a



(e) Detail from Supp. Fig. 46a

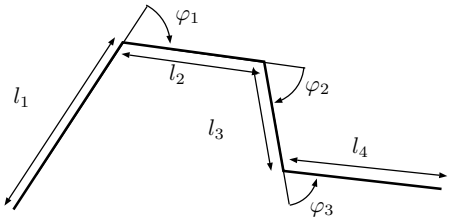


(f) Detail from Supp. Fig. 46a

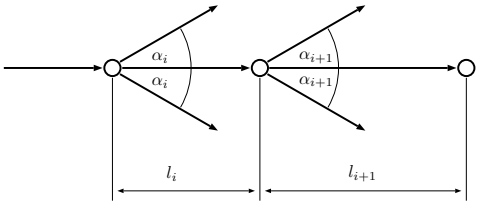


(g) Detail from Supp. Fig. 46a

Supplementary Figure 46: Waterbomb crease pattern graph.



Supplementary Figure 47: Piecewise linear curve with segment lengths l and fold angles φ. The sign of the fold angles φ is defined such that an arrow in counterclockwise direction marks a positive angle.



Supplementary Figure 48: Segments of an origami strip.

	$\rho > 0$	$\rho < 0$
$\varphi > 0$	$\alpha = \arcsin\left(\cos\left(\frac{\rho}{2}\right)\frac{\sin(\varphi)}{\sin(\gamma)}\right)$ mode: true	$\alpha = 180° - \arcsin\left(\cos\left(\frac{\rho}{2}\right)\frac{\sin(\varphi)}{\sin(\gamma)}\right)$ mode: true
$\varphi < 0$	$\alpha = 180° + \arcsin\left(\cos\left(\frac{\rho}{2}\right)\frac{\sin(\varphi)}{\sin(\gamma)}\right)$ mode: false	$\alpha = -\arcsin\left(\cos\left(\frac{\rho}{2}\right)\frac{\sin(\varphi)}{\sin(\gamma)}\right)$ mode: false

Supplementary Table 2: Sector angles and mode assignments for achieving a fold angle φ at a dihedral angle ρ. The angle ρ (or its sign, respectively) is measured on the segment with length l_i when a fold angle of φ_i is to be achieved (cf. Supp. Fig. 47).

Supplementary Note 5.7: Extruding Procedure

Linear strips such as those designed with the method outlined in the previous subsection can serve as the basis for a tessellated pattern. To that end, an edge can be extended until the vertex at its end has a distance of h_1 to the central edges, as shown in Supp. Fig. 51. Next, the moved vertex is expanded with three outgoing edges at sector angles equal to those at its direct predecessor. From here, the quadrilateral-closing operation can be applied repeatedly to obtain a second layer. The scheme is repeated to obtain a bigger pattern using a distance h_{i-1} to obtain the layer i. The algorithm to extrude a linear strip to a pattern is summarized in Supp. Alg. 12 and illustrated in Supp. Fig. 50.

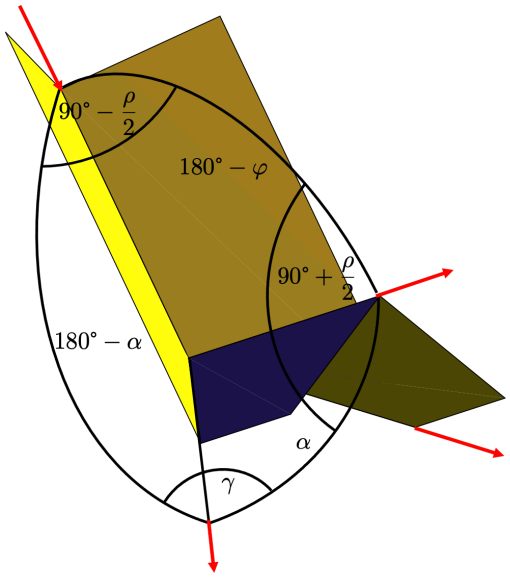
If h_i is held constant for all layers, this leads to every second layer being identical (also in the folded configuration). The maximum feasible layer height (exceeding it results in intersecting edges which is not allowed) can be calculated by iterating through all segments of the origami strip and calculating the maximum layer height for this segment. Using basic trigonometry, the maximum layer height for the segment with length l_i can be calculated to

$$h_{\max,i} = \begin{cases} l_i \frac{\sin(\alpha_i)\sin(\alpha_{i-1})}{\sin(\alpha_i - \alpha_{i-1})} & \alpha_i < \alpha_{i-1} \\ \infty & \alpha_i \geq \alpha_{i-1} \end{cases} \tag{16}$$

The maximum layer height $h_{\max}$ is then given by the minimum over all $h_{\max,i}$.

If h_i is not constant, layer i is identical to the second layer that was obtained with the height

$$\xi(i) = \sum_{j=1}^{i} (-1)^{j+1} h_j \tag{17}$$



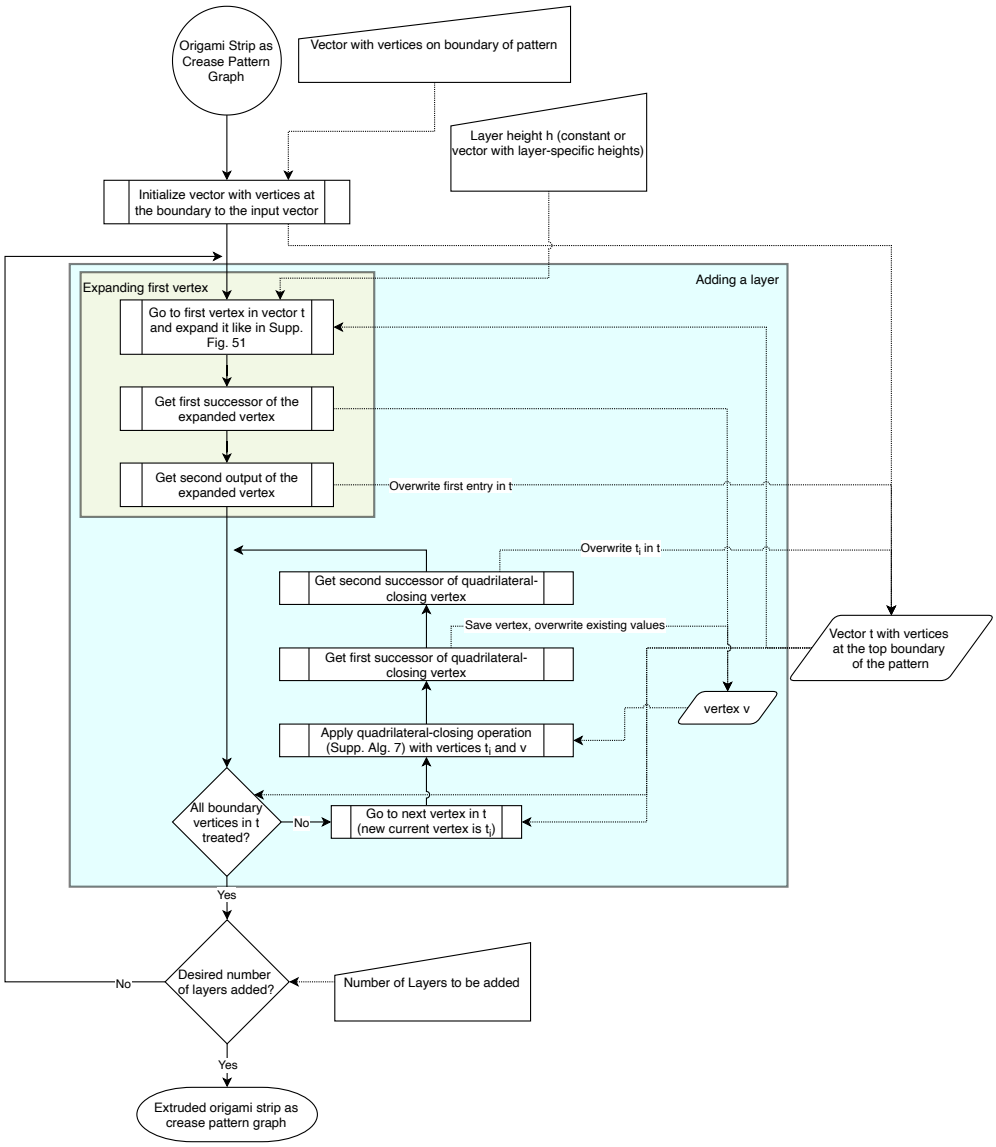
Supplementary Figure 49: Geometry of a partially folded symmetric vertex of degree 4.

and may not be larger than $h_{\max}$ for any i. Furthermore, it may not be smaller than a value $h_{\min} \leq 0$. This value can be obtained similarly to $h_{\max}$ with the adaption that α_i and α_{i-1} are switched in Eq. (16). We notice that if one segment with a very small l_i exists in the initial strip, $h_{\min}$ is almost 0.

In patterns where h_i is not constant for all i, the cross-sectional shape varies from layer to layer while all layers with identical ξ remain identical for the complete folding process. Therefore, ξ is a variable that describes the shape of the cross-section and can take any value between $h_{\min}$ and $h_{\max}$ for all layers. As shown in Fig. 3I in the main text, the values for h_i can be chosen such that ξ approximates an arbitrary continuous function. The values for ξ used for the example in the main text are shown in Supp. Fig. 52. Moving the two guiding functions closer together (blue curves in Supp. Fig. 52; an adapted cosine function was used in this example) leads to more and therefore smaller layers (lower heights h_i). The geometry of the folded configuration then appears smoother. Another option to edit the resulting surface is to model a single fold vertex with fold angle φ_i in the original piecewise linear curve as two fold vertices with fold angles $\varphi_{i1} + \varphi_{i2} = \varphi_i$ separated by a very short segment. This introduces one editable variable per fold vertex.

Supplementary Note 5.8: Tube Example

The square tube that changes its cross-section along its axis that is shown in Fig. 3I of the main text is obtained by first designing a strip that approximates a square with side length l and then applying an irregular extrusion. More precisely, the square that was initially approximated is an octagon where every second side length ε is very small. The side lengths and bend angles of the



Supplementary Figure 50: Extruding procedure (Illustration of Supp. Alg. 12).

Supplementary Algorithm 12: Extruding procedure.

Input: 1) Origami strip
2) List of boundary vertices at the top of the strip (in vector t)
3) Number of layers to add
4) Layer height h

Result: Extruded strip

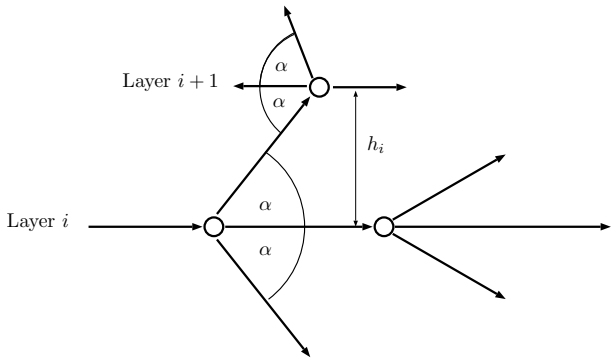
for *number of layers to add* **do**
  `/* Apply layer-adding procedure (Supp. Alg. 11) */`
  Move vertex t_1 along its input edge until it is h away from the central crease of the previous layer (cf. Supp. Fig. 51);
  Add outputs to t_1 as shown in Supp. Fig. 51;
  $v \leftarrow$ first output to t_1;
  $t_1 \leftarrow$ second output to t_1;
  for t_2 *to* t_{end} **do**
    Apply quadrilateral-closing operation (Supp. Alg. 7) with v and current t_i;
    $v \leftarrow$ First output created during quadrilateral-closing operation;
    $t_i \leftarrow$ Second output created during quadrilateral-closing operation;
  end
end

piecewise linear curve are

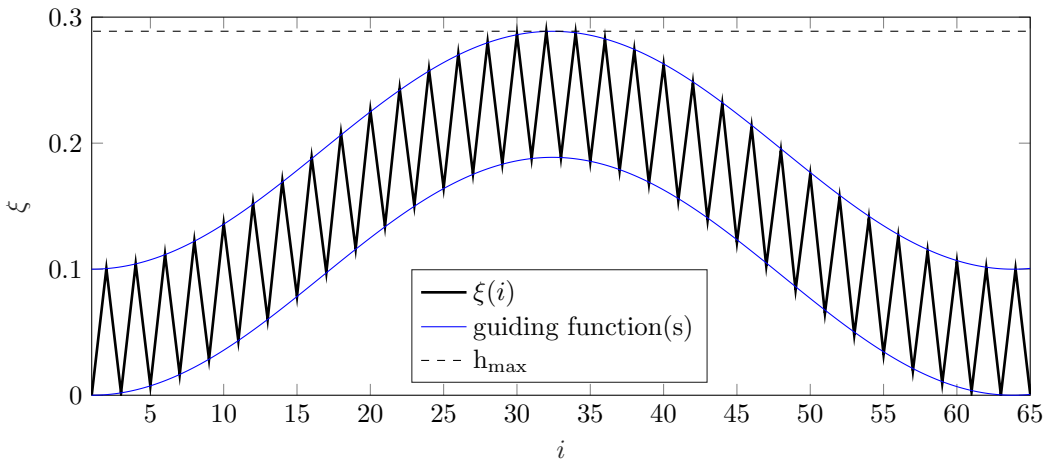
$$\{l_i\} = \left\{ l - \frac{\varepsilon}{\sqrt{2}}, \varepsilon, l - \frac{\varepsilon}{\sqrt{2}}, \varepsilon, l - \frac{\varepsilon}{\sqrt{2}}, \varepsilon, l - \frac{\varepsilon}{\sqrt{2}}, \varepsilon \right\} \tag{18}$$

$$\varphi_i = -45^\circ \quad \forall i \in \{1 \cdots 7\} \tag{19}$$

An input dihedral angle of $\rho = 90^\circ$ (and symmetric folding) is used and the strip is designed using the equations in Supp Tab. 2. Supp. Alg. 12 is used to extrude the strip to a pattern, but the layer heights h were varied from layer to layer based on the values for ξ shown in Supp. Fig. 52. The first and last length of the central crease at each layer was chosen such that a closed curve is formed.



Supplementary Figure 51: Addition of a layer to an origami strip.



Supplementary Figure 52: Values of ξ for the example shown in Fig. 3I in the main text. For the base strip used there, $h_{\min}$ is approximately 0.

Supplementary Note 6: Translating and Mirroring Operations

In this section, we give a detailed description of the translating and mirroring operations and discuss the corresponding examples presented in the main text.

Supplementary Note 6.1: Evaluation of Mirrored Vertices

Vertices that are generated by mirroring another "parent vertex" by a plane defined by three vertices can be evaluated if (and only if) the parent vertex and the three vertices defining the mirror plane have been evaluated. This condition is used to evaluate Line 6 in Supp. Alg. 1 for mirrored vertices regardless of the type of the vertex. The sector angles of the mirrored vertex must remain identical to those of the parent vertex after the mirroring operation. Therefore, they can essentially be copied from the parent vertex, but need to be permuted to preserve the indexation convention shown in Supp. Fig. 7. A vertex with n inputs and m outputs takes the sector angles α from the parent vertex permuted as

$$\underbrace{\alpha_1 \cdots \alpha_{m+n}}_{\text{mirrored vertex}} \Leftarrow \underbrace{\alpha_{n+1}, \alpha_n \cdots \alpha_2, \alpha_1, \alpha_{m+n}, \alpha_{m+n-1}, \cdots \alpha_{n+2}}_{\text{parent vertex}} \tag{20}$$

The position of the mirrored vertex $\vec{m}$ can be obtained from the position of a vertex on the mirror plane $\vec{v}_1$, the normal vector to the mirror plane $\vec{n}$ and the position of the parent vertex $\vec{p}$ as

$$\vec{m} = \vec{p} - 2\left(\vec{n} \cdot (\vec{p} - \vec{v}_1)\right)\vec{n} \tag{21}$$

Similarly, mirroring a vector $\vec{v}$ about a plane with normal $\vec{n}$ results in the vector

$$\vec{v}_m = \vec{v} - 2(\vec{v} \cdot \vec{n})\vec{n} \tag{22}$$

When evaluating a mirrored vertex, this relation is used to obtain the normal vectors of the adjacent faces and the crease vectors of the incident edges of the mirrored vertex from its parent vertex. While the face normals, after mirroring using Eq. (22), need to be permuted like the sector angles (Eq. (20)), the dihedral angles and the crease vectors undergo a different permutation (after mirroring). If the vertex has n inputs and m outputs, the dihedral angles ρ (and the crease vectors) are permuted as

$$\underbrace{\rho_1 \cdots \rho_{m+n}}_{\text{mirrored vertex}} \Leftarrow \underbrace{\rho_n, \rho_{n-1} \cdots \rho_2, \rho_1, \rho_{m+n}, \rho_{m+n-1} \cdots \rho_{n+1}}_{\text{parent vertex}} \tag{23}$$

Supplementary Note 6.2: Evaluation of Translated Vertices

Vertices that are generated by translating another parent vertex by a vector defined by a start and an end vertex can be evaluated if (and only if) the parent vertex and the two vertices defining the translation vector have been evaluated. This condition is used in Line 6 in Supp. Alg. 1 regardless of the type of the vertex. Since changing the sector angles around a translated vertex is not allowed, they can be directly copied from the parent vertex. During evaluation, the translation vector can then be calculated from the positions of the two vertices defining the translation vector. The dihedral angles, direction vectors of the creases and face normals can also be directly copied from the parent vertex. They are then used to calculate the positions of the neighbouring vertices.

Supplementary Note 6.3: Applicability

An origami mechanism in three dimensions has seven degrees of freedom (three translational, three rotational degrees of freedom and its mechanism/folding degree of freedom). If a single face is fixed and one dihedral angle is set, each vertex has one unique (and determinable) position in space. We now consider an arbitrary second origami (e.g. a copy of the first one) and look for conditions for connecting it to the first one such that it also has the property that the positions of all of its vertices is unique and determinable, This is achieved if seven of its degrees of freedom are fixed as well. In general, each point (vertex) at which the two origamis are connected constrains three degrees of freedom. Therefore, if the second origami is connected to the first (fixed) one at three or more vertices, the second origami has the desired property (i.e. it is fixed in space as well). However, if all three connecting vertices of either origami belong to the same face, only six degrees of freedom are "transmitted", which is insufficient.

If the mirroring operation, as presented in the main text, is considered, the first (fixed) origami refers to the original origami and the mirrored copy refers to the second origami. The two origamis are, by definition, always connected at three or more vertices (the three vertices that define the mirror plane). Therefore, in general, the structure formed by the original and mirrored origami has a single folding degree of freedom. If, however, the three vertices belong to the same face, the compound structure has two degrees of freedom. Similarly, if the three mirror vertices form a line[1] and the original origami is considered to be fixed, the mirrored origami can still rotate about the axis defined by the three vertices. This, too, implies that the property of having a single folding degree of freedom is not preserved. Therefore, the mirroring application is not allowed if the three mirroring vertices are part of the same face or form a line for all folding states.

Nevertheless, finding possible ways to apply a mirroring operation is therefore comparatively easy. One of the biggest challenges remaining is the avoidance of self-collisions. Considering a translation operation, a contact at one vertex can be guaranteed if the two vertices defining the translation operation are part of the translated set. The case when the translation results in three contact vertices that remain in contact throughout the complete folding process (and are neither on the same face nor form a line), as required, remains a rather specific case and typically relies on some kind of symmetry within the pattern and its folding kinematics. This is the reason why the translation operation can be applied much less often.

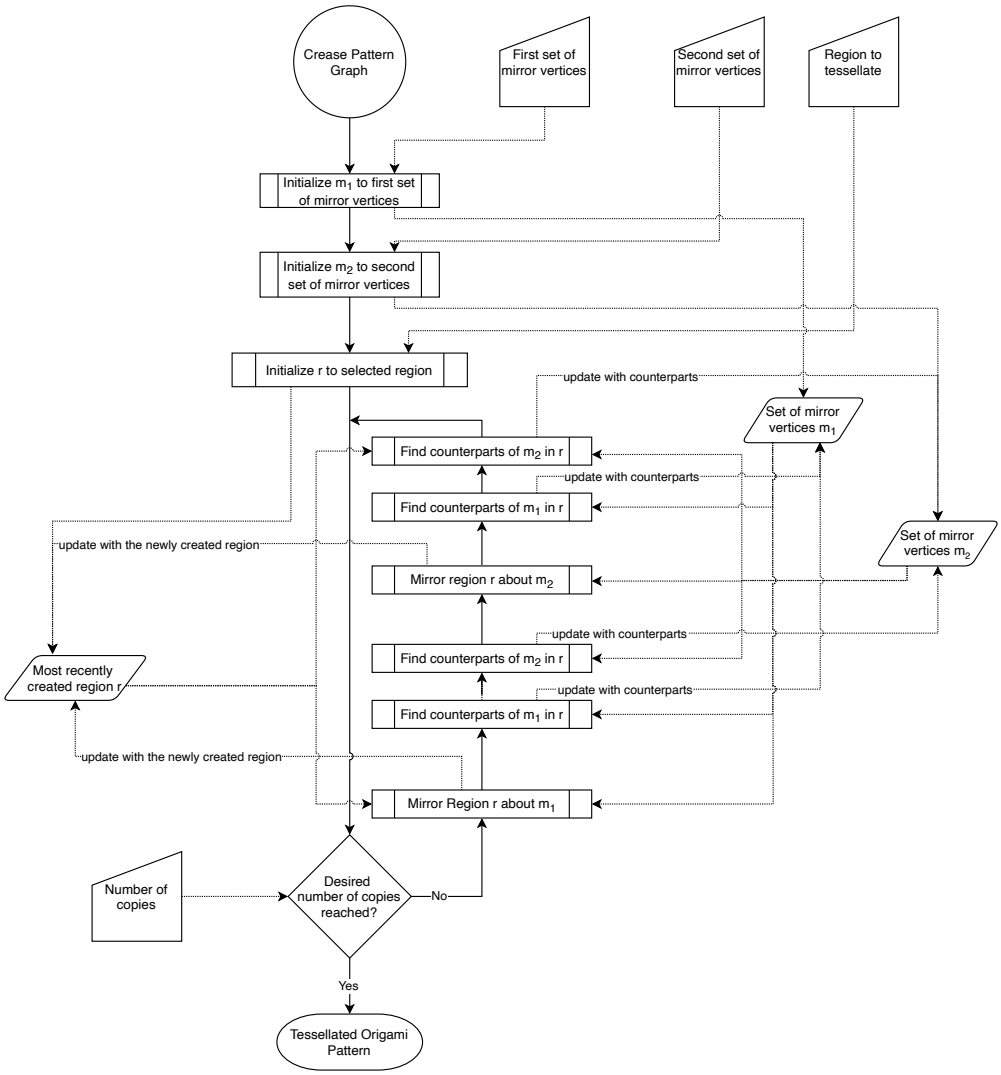
Supplementary Note 6.4: Tessellating Procedure

The tessellations shown in Fig. 4E-I are obtained using two sets of (three) mirroring vertices. The two sets (or their mirrored copies, respectively) are used alternatingly to mirror the most recently obtained pattern. The method to tessellate a pattern using the mirroring operation is shown in Supp. Alg. 13 and illustrated in Supp. Fig. 53.

Lines 4 and 8 represent special cases. The set of vertices that is being replaced (overriden) was, in the previous mirroring operation, used as mirroring vertex set. Since these vertices were located on the mirroring plane, they were mirrored to vertices that have the same location (these mirrored vertices are the counterparts that are used to override the sets in these lines). The only difference between the original vertex set and the one it is overriden with here is that the new set belongs to the mirrored pattern.

The mirroring operations in L. 3 and 6 return ($\leftarrow$) only the pattern that was newly created during the operation, i.e. the mirrored pattern.

[1] This also implies computational issues since no normal vector to the mirror plane can be calculated.



Supplementary Figure 53: Tessellating Procedure (Illustration of Supp. Alg. 13).

Supplementary Algorithm 13: Tessellating Procedure

Input: 1) Original Origami Pattern
2) First set of mirror vertices $m^1 = \{m_1^1, m_2^1, m_3^1\}$
3) Second set of mirror vertices $m^2 = \{m_1^2, m_2^2, m_3^2\}$
4) Desired number of copies n

Result: Tessellated Origami Pattern

Most recently obtained pattern $r \leftarrow$ original pattern;
for $1 \cdots n$ **do**
  $r \leftarrow$Mirror r by m^1;
  $m^1 \leftarrow$ Counterparts of m^1 in r;
  $m^2 \leftarrow$ Counterparts of m^2 in r;
  $r \leftarrow$Mirror r by m^2;
  $m^1 \leftarrow$ Counterparts of m^1 in r;
  $m^2 \leftarrow$ Counterparts of m^2 in r;
end

Supplementary Note 6.5: Examples

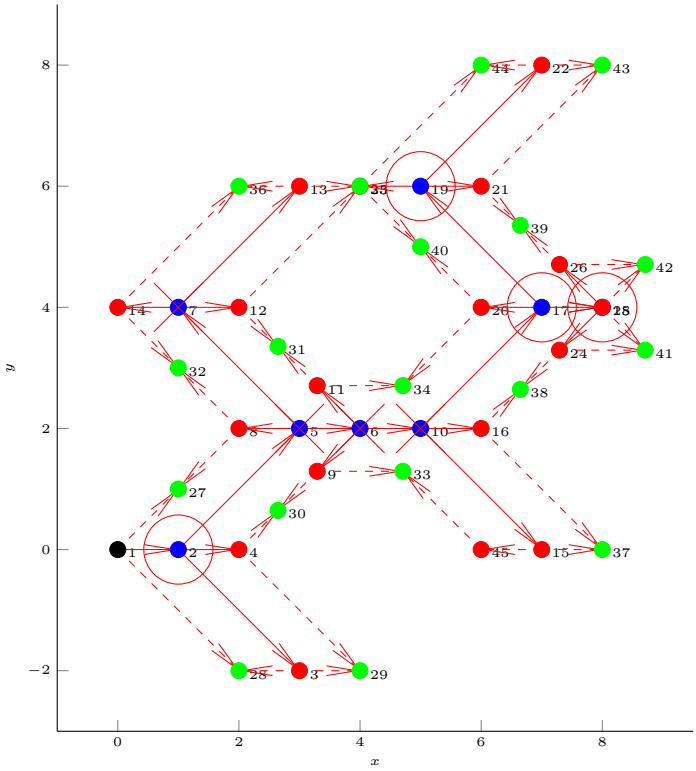
Fig. 4C The example shown in Fig. 4C of the main text ("translation example") is based on the crease pattern graph shown in Supp. Fig. 54. First, a translation operation is used to obtain a pattern p_2 by translating the initial pattern (p_1) by a vector from vertex 3 to vertex 13. The patterns p_1 and p_2 are then translated by the vector from vertex 1 to vertex 25 to obtain patterns p_3 and p_4. Next, a translation along the vector from vertex 1 to vertex 6 of patterns $p_1 - p_4$ yields patterns $p_5 - p_6$. Symmetric folding and a dihedral angle of 120° at the input edge are considered to obtain the shown configuration.

Fig. 4F The example shown in Fig. 4F of the main text ("rotational symmetry") is based on the crease pattern graph shown in Supp. Fig. 55. The pattern is then tessellated using Supp. Alg. 13. As first mirror set, $m^1 = \{3, 6, 10\}$ and as second mirror set, $m^2 = \{4, 7, 11\}$ are used. Using $n = 5$ leads to the pattern shown in the main text.

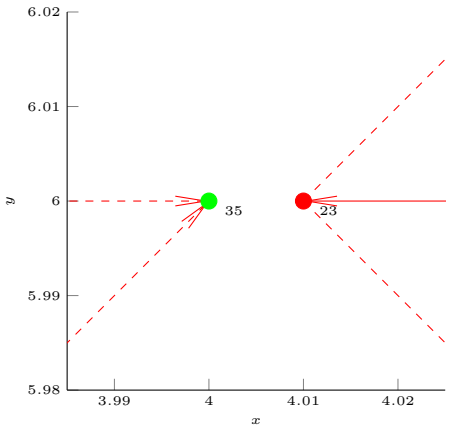
In the unfolded configuration, all mirrored copies coincide. Folding the pattern then by increasing the dihedral angle at the input edge results in many self-collisions. The folding is free of self-collisions if the input dihedral angle is larger than $\approx 84.3°$ (this configuration is shown in Fig. 4C of the main text; symmetric folding was used). When increasing the input dihedral angle, the origami remains free of self-collisions until it is fully folded and all patterns coincide again.

Fig. 4G The example shown in Fig. 4G of the main text ("3D tessellation") is based on a strip-like crease pattern graph. The strip is obtained by repeating the unit cell shown in Supp. Fig. 56 five times in x-direction. The strip is then mirrored by procedure in Supp. Alg. 13 using $m^1 = \{9, 19, 22\}$ as first mirroring set and $m^2 = \{10, 20, 23\}$ as second mirror vertex set. Using $n = 2$ results in a total of five parallel strips (notice that within one iteration of the for-loop, two mirroring operations are executed).

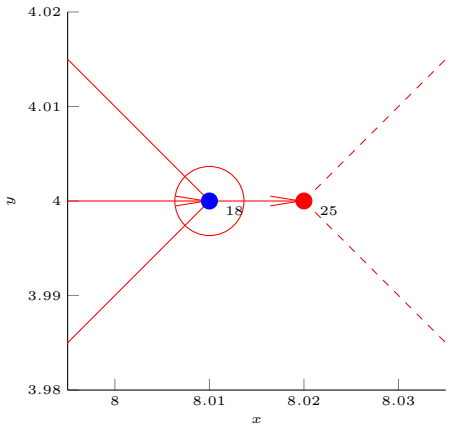
Next, Supp. Alg. 13 is applied to the complete pattern (consisting of all five strips). The first set of mirror vertices m^1 consists of the vertices 9, 10 and one of their counterparts in another unit cell in the first strip. The second set of mirror vertices m^2 consists of the vertices 18, 21 and one of their counterparts in another parallel strip. $n = 3$ is used to mirror the "base layer"



(a) Complete graph. Overlapping indices are caused by very closely spaced vertices (cf. Supp. Figs. 54b and 54c)

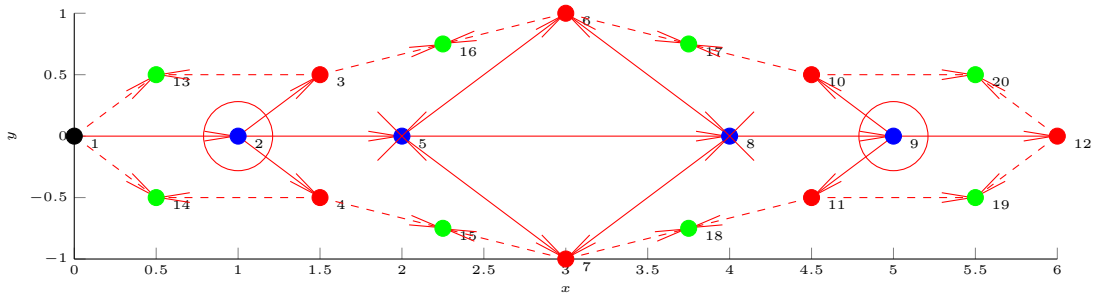


(b) Detail from Supp. Fig. 54a

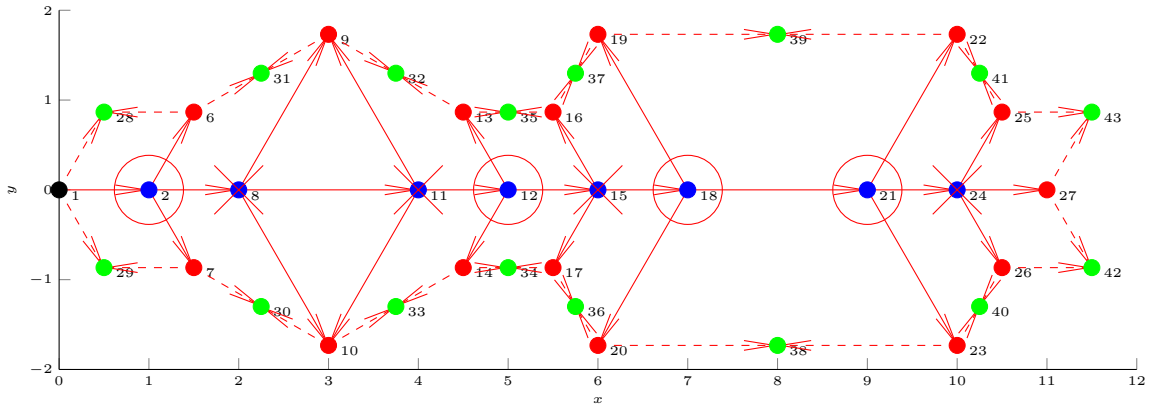


(c) Detail from Supp. Fig. 54a

Supplementary Figure 54: Basis of crease pattern graph for the example shown in Fig. 4E of the main text.



Supplementary Figure 55: Basis of crease pattern graph for the example shown in Fig. 4C of the main text.



Supplementary Figure 56: Basis of crease pattern graph for the example shown in Fig. 4D of the main text.

(consisting of five strips) until seven layers are obtained (i.e. six layers are added).

The complete structure consists of 6443 vertices, 9278 edges and 2879 faces. For the configuration shown in the main text, an input dihedral angle of 40° and the option of symmetric folding was used. Evaluating a folded configuration takes 225.5 s on a 2.3 GHz Intel Core i9 processor.

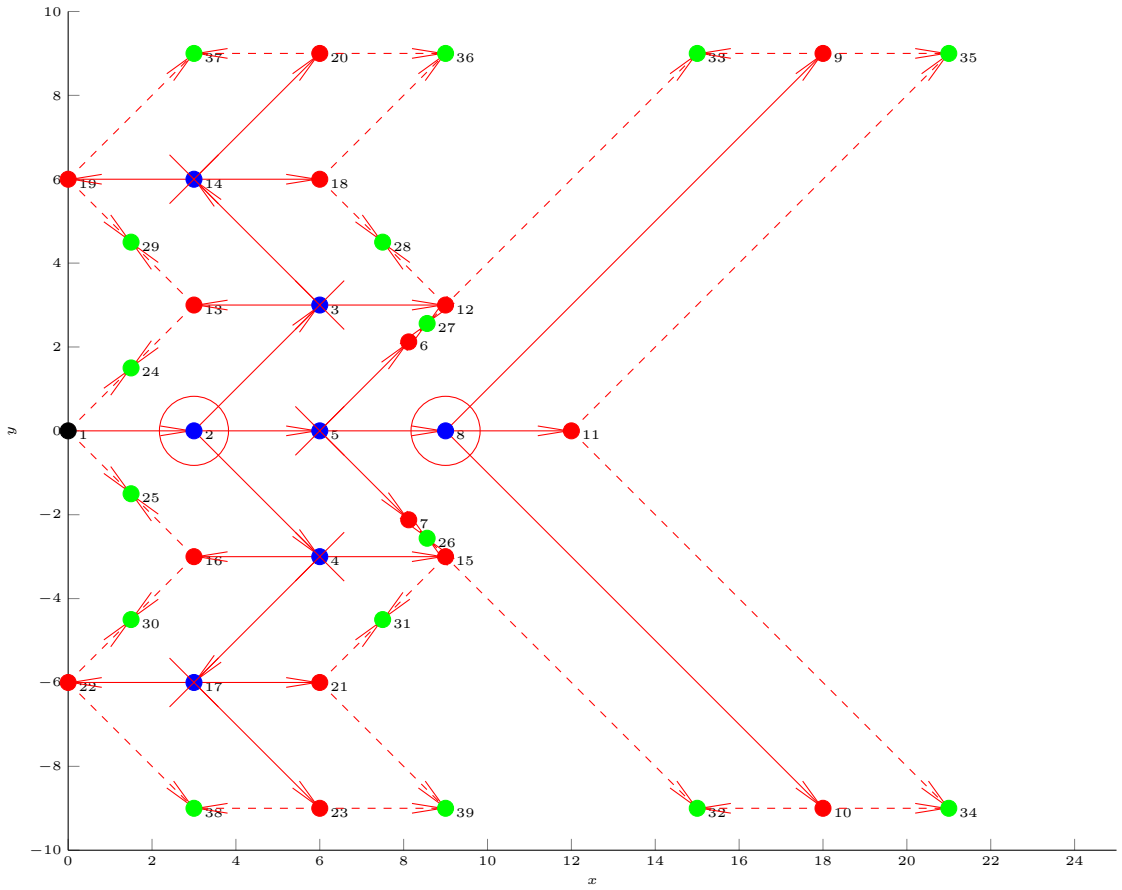
Fig. 4H The example shown in Fig. 4H of the main text is based on the square twist pattern whose pattern graph is also shown in Supp. Fig. 45. The pattern in Supp. Fig. 45 is equivalent to a quarter of one "layer" used in this example. The structure in the presented example is obtained by using Supp. Alg. 13, with $m^1 = \{18, 21, 30\}$ as first set of mirroring vertices and $m^2 = \{1, 2, 6\}$ as second set. $n = 6$ is used to obtain a structure with a total of 13 layers.

For both sets m^1 and m^2, the three vertices belong to the same face. Usually, as explained in Supp. Note 6.3, this would be insufficient since the resulting structure is not a mechanism with a single degree of freedom. In this case, however, mirroring one origami by the mirror vertex sets m^1 or m^2 results in mirrored copies that are in contact with the original origami at several faces, which is sufficient to preserve the property of having a single folding degree of freedom.

The full structure consists of 1716 vertices, 2756 edges and 1053 faces. The face with indices 1, 2, 6 and 46 is fixed and a dihedral angle of 90° at the input edge is used to obtain the shown configuration. Evaluating the folded configuration takes 13.0 s on a 2.3 GHz Intel Core i9 processor.

Fig. 4I The example shown in Fig. 4I uses a pattern described by Eidini and Paulino[7]. The graph that was used here is shown in Supp. Fig. 57. In a first step, a "base layer" is built by first translating the original pattern by the vector from vertex 37 to vertex 35 and then translating the original and the translated pattern by a vector from 23 to vertex 20 (resulting in a total of four instances of the pattern shown in Supp. Fig. 57). Supp. Alg. 13 is used to tessellate this "base layer", using $m^1 = \{1, 13, 16\}$ as first and $m^2 = \{2, 3, 4\}$ as second set of mirror vertices. $n = 6$ is used to obtain a structure with a total of 13 instances of the base layer.

The structure consists of 2028 vertices, 2808 edges and 832 faces. For the shown configuration, the option of symmetric folding and a dihedral angle of 100° at the input edge is used. Evaluating the folded configuration takes 12.7 s on a 2.3 GHz Intel Core i9 processor.



Supplementary Figure 57: Basis of crease pattern graph for the example shown in Fig. 4G of the main text.

Supplementary Note 7: Gluing Operation

Supplementary Note 7.1: Evaluation

An interface between a base and an attached origami consists of one vertex and two edges on the base origami. The source vertex of the attached origami (and therefore the complete attached origami) can be evaluated if (and only if) the position of the connecting vertex and the two direction vectors of the two connecting edges were evaluated. This is the case after the connecting vertex has been evaluated in the base origami. To evaluate the complete structure, Supp. Alg. 1 is extended to Supp. Alg. 14, which is illustrated in Supp. Fig. 58.

Supplementary Algorithm 14: Algorithm to obtain the folded configuration of an origami

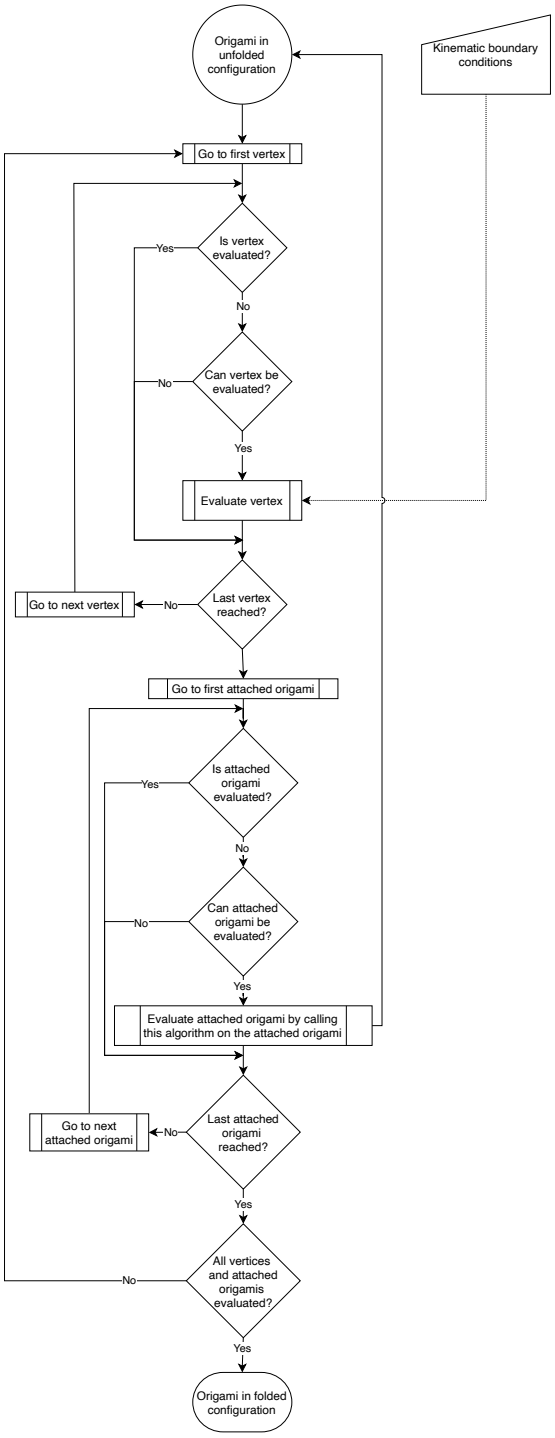
```
while not (all vertices and all attached origamis evaluated) do
    for all vertices do
        if vertex is evaluated then
            continue
        else
            if vertex can be evaluated then
                evaluate vertex;
            end
        end
    end
    for all attached origamis do
        if origami is evaluated then
            continue
        else
            if origami can be evaluated then
                Evaluate attached origami (call this algorithm for the attached origami);
            end
        end
    end
end
```

The geometry of such an interface is shown in Supp. Fig. 59. The edge direction (unit) vectors $\vec{v}_1$ and $\vec{v}_2$ are known from the evaluation of the interfacial vertex in the base origami. Furthermore, the sector angles α_1 and α_2 are known from the unfolded configuration of the attached origami. From the vectors $\vec{v}_1$ and $\vec{v}_2$, the vector $\vec{n}$ and the angle γ can be calculated directly using

$$\vec{n} = \frac{\vec{v}_1 \times \vec{v}_2}{\|\vec{v}_1 \times \vec{v}_2\|} \tag{24}$$

and

$$\gamma = \arccos\left(\frac{\vec{v}_1 \cdot \vec{v}_2}{\|\vec{v}_1\|\|\vec{v}_2\|}\right) \tag{25}$$



Supplementary Figure 58: Algorithm to obtain the folded configuration of an origami (Illustration of Supp. Alg. 14).

respectively. The angles β_1 and β_2 can be calculated according to

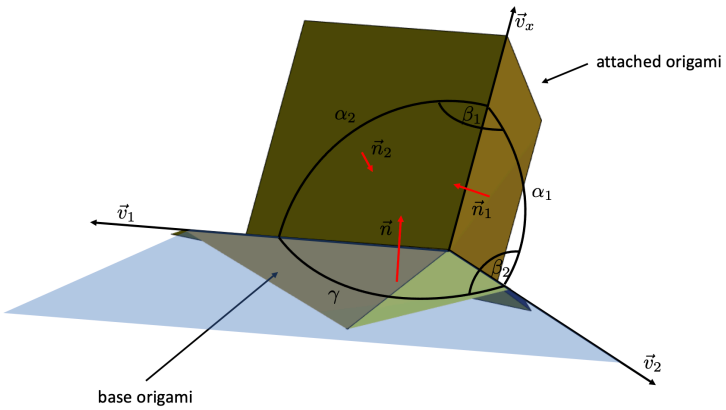
$$\beta_1 = \arccos\left(\frac{\cos(\gamma) - \cos(\alpha_1)\cos(\alpha_2)}{\sin(\alpha_1)\sin(\alpha_2)}\right) \tag{26}$$

$$\beta_2 = \arccos\left(\frac{\cos(\alpha_2) - \cos(\gamma)\cos(\alpha_1)}{\sin(\gamma)\sin(\alpha_1)}\right) \tag{27}$$

using the spherical law of cosines. Next, the vector $\vec{n}_1$ can be obtained by rotating $\vec{n}$ around $\vec{v}_2$ by an angle of $180° - \beta_2$. Next, the vector $\vec{v}_2$ is rotated around $\vec{n}_1$ by α_1 to obtain the vector $\vec{v}_x$, the crease vector of the central outgoing edge of the attached origami. The dihedral angle ρ at this edge is

$$\rho = 180° - \beta_1 \tag{28}$$

Finally, rotating $\vec{n}_1$ around $\vec{v}_x$ by ρ yields $\vec{n}_2$.



Supplementary Figure 59: Geometry of a source vertex of a glued origami in folded configuration.

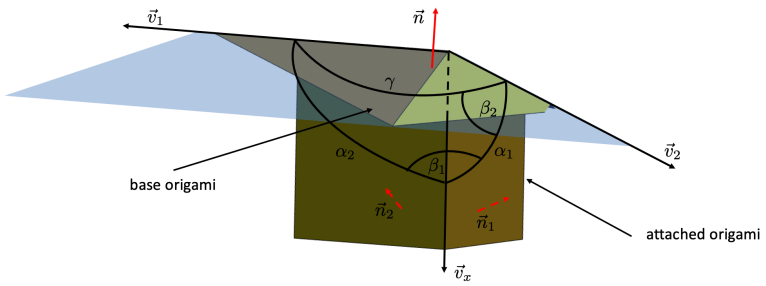
Similarly to the evaluation of an interior vertex (where two sets of dihedral angles for the output edges can be chosen from), there are two possible geometric solutions which can be chosen from. For example, another way to glue the two origamis shown in Supp. Fig. 59 is shown in Supp. Fig. 60. In this case, the angles γ, β_1, β_2 and the vector $\vec{n}$ are calculated like for the situation shown in Supp. Fig. 59 (using Eqns. (24)-(27)). To obtain the vector $\vec{n}_1$, $\vec{n}$ is rotated by $-(180° - \beta_2)$ around $\vec{v}_2$. The dihedral angle ρ at the main outgoing edge is

$$-(180° - \beta_1). \tag{29}$$

Supplementary Note 7.2: Examples

Interface Edge Convention The convention is that the second interface edge is connected to the first outgoing edge of the source vertex of the attached origami (e.g. the edge from vertex 1 to vertex 14 in Supp. Fig. 61b) and the first interface edge is connected to the last outgoing edge of the source vertex (e.g.the edge from vertex 1 to vertex 13 in Supp. Fig. 61b).

Fig. 4K The example in Fig. 4K of the main text is based on two crease pattern graphs. They are shown in Supp. Fig. 61. The generation of the structure is described in Supp. Alg. 15.



Supplementary Figure 60: Alternative gluing connection for the same origamis as in Supp. Fig. 59.

Supplementary Algorithm 15: Generating the example in Fig. 4K of the main text.

1 $p_1 \leftarrow$ instance of pattern in Supp. Fig. 61a;
2 $p_{1,1} \leftarrow$ Gluing an instance of pattern in Supp. Fig. 61b to p_1 (interfacce vertex: 5, first interface edge: 5-7, second interface edge: 5-6);
3 $p_{1,2} \leftarrow$ Gluing an instance of pattern in Supp. Fig. 61b to p_1 (interface vertex: 8, first interface edge: 8-6, second interface edge: 8-7;
4 $p_{1,1,1} \leftarrow$ Gluing an instance of pattern in Supp. Fig. 61b to $p_{1,1}$ (interface vertex: 5, first interface edge: 5-7, second interface edge: 5-6);

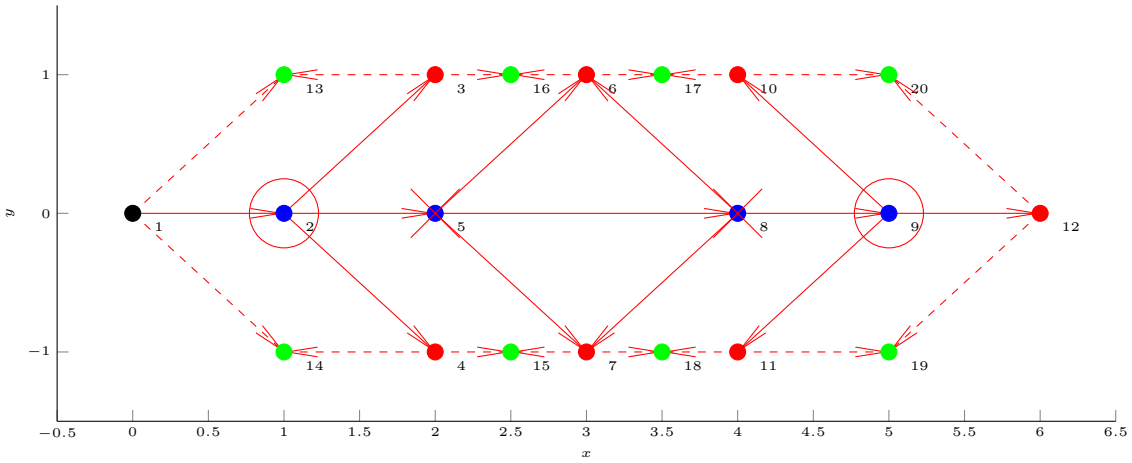
Due to the symmetry of the involved patterns, vertices 12, 19 and 20 of pattern $p_{1,1,1}$ remain in contact with pattern $p_{1,2}$ throughout the folding process.

For the shown configuration, symmetric folding and a dihedral angle of 80° at the input edge of p_1 are used.

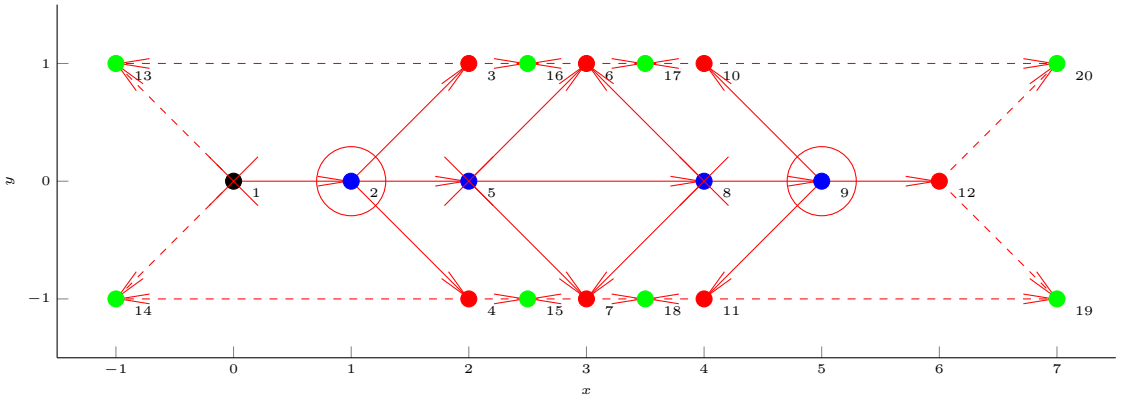
Fig. 4L The structure shown in Fig. 4L of the main text is based on the crease pattern graphs in Supp. Fig. 62. The generation of the structure is described in Supp. Alg. 16.

Due to the symmetry of the patterns, vertex 13 of pattern $p_{1,1}$ (Supp. Fig. 62b) is always in contact with vertex 10 of pattern p_1 (Supp. Fig. 62a). The same applies for their mirrored counterparts.

For the shown configuration, symmetric folding and a dihedral angle of 90° is used at the input edge at pattern p_1.

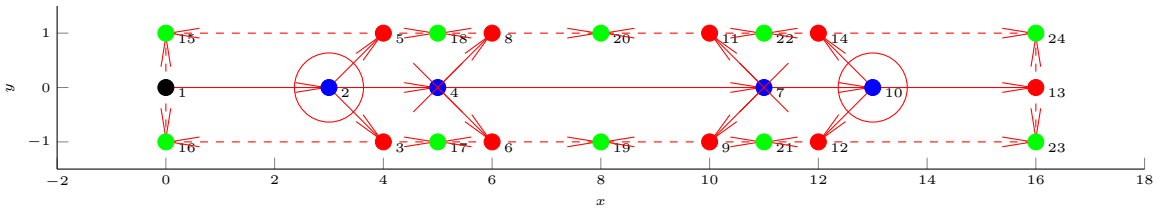


(a) Base Pattern.

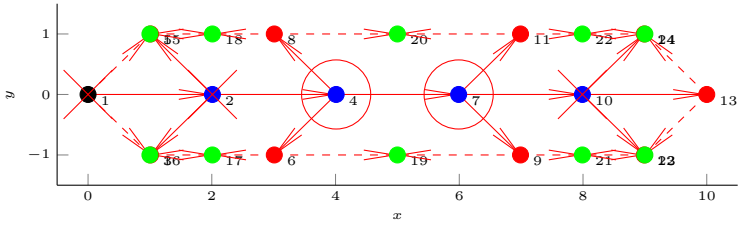


(b) Attached Pattern.

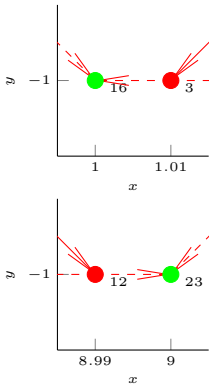
Supplementary Figure 61: Crease pattern graphs for the structure in Fig. 4I of the main text.



(a) Pattern 1.

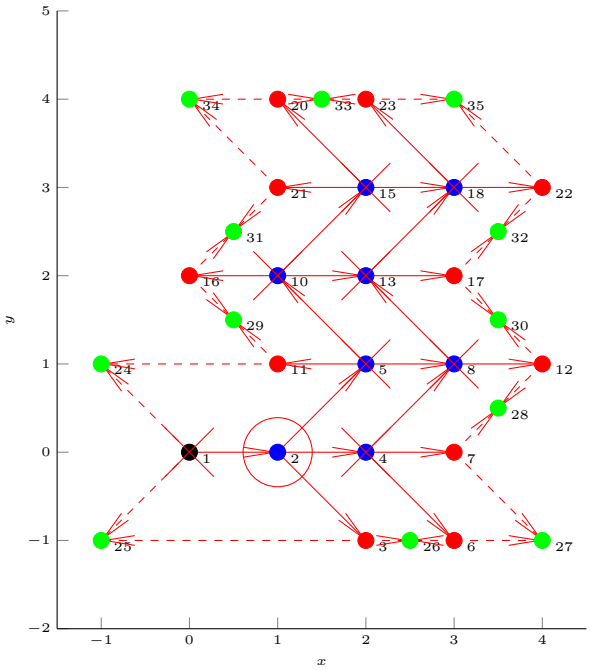


(bi) Full Graph. Overlapping indices are caused by very closely spaced vertices (cf. Supp. Fig. 62bii)

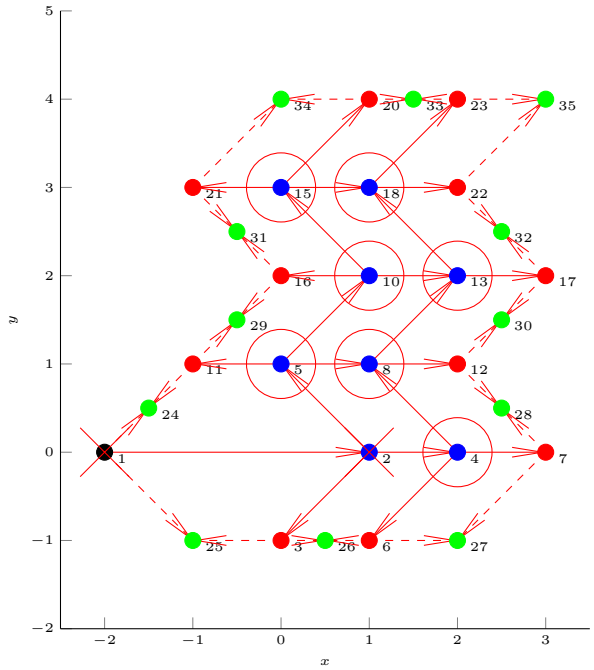


(bii) Details from Supp. Fig. 62bi

(b) Pattern 2.



(c) Pattern 3.



(d) Pattern 4.

Supplementary Figure 62: Crease Pattern Graphs for Structure in Fig. 4J of the main text.

Supplementary Algorithm 16: Generating the example in Fig. 4L of the main text. The notation p_{n*} refers to p_n and, recursively, all patterns attached to it.

1 $p_1 \leftarrow$ instance of pattern in Supp. Fig. 62a;
2 $p_{1,1} \leftarrow$ Gluing an instance of pattern in Supp. Fig. 62b to p_1 (interface vertex: 2, first interface edge: 2-5, second interface edge: 2-3);
3 $p_{1,1,1} \leftarrow$ Gluing an instance of pattern in Supp. Fig. 62d to $p_{1,1}$ (interface vertex: 4, first interface edge: 4-6, second interface edge: 4-8);
4 $p_{1,1,2} \leftarrow$ Gluing an instance of pattern in Supp. Fig. 62d to $p_{1,1}$ (interface vertex: 7, first interface edge: 7-11, second interface edge: 7-9);
5 $p_{1,2} \leftarrow$ Gluing an instance of pattern in Supp. Fig. 62c to p_1 (interface vertex: 4, first interface edge: 4-6, second interface edge: 4-8);
6 $p_{1,3} \leftarrow$ Gluing an instance of pattern in Supp. Fig. 62c to p_1 (interface vertex: 7, first interface edge: 7-11, second interface edge: 7-9);
7 $p_{2*} \leftarrow$ Mirror p_{1*} using $\{13, 23, 24\}$ of p_1;
8 $\{p_{3*}, p_{4*}\} \leftarrow$ Mirror p_{1*} and p_{2*} by $\{20, 23, 34\}$ of $p_{1,2}$;
9 $\{p_{5*} \cdots p_{8*}\} \leftarrow$ Mirror $p_{1*} \cdots p_{4*}$ by $\{8, 12, 22\}$ of $p_{1,2}$;

Supplementary References

[1] Zimmermann, L., Shea, K. & Stanković, T. Conditions for Rigid and Flat Foldability of Degree-n Vertices in Origami. *Journal of Mechanisms and Robotics* **12**, 011020 (2020).

[2] Wolfram Alpha LLC. Elephant curve. `https://www.wolframalpha.com/input/?i=elephant+curve&wal=header`, last checked 18.06.2021.

[3] Bourke, P. Geometry of sports balls (2017). `http://paulbourke.net/geometry/spherical/`, last checked 23.06.2021.

[4] Feng, F., Dang, X., James, R. D. & Plucinsky, P. The designs and deformations of rigidly and flat-foldable quadrilateral mesh origami. *Journal of the Mechanics and Physics of Solids* **142**, 104018 (2020).

[5] Lang, R. J. *Twists, Tilings, and Tessellations: Mathematical Methods for Geometric Origami* (CRC Press, Boca Raton, 2018).

[6] Kamrava, S., Mousanezhad, D., Felton, S. M. & Vaziri, A. Programmable Origami Strings. *Advanced Materials Technologies* **3**, 1700276 (2018).

[7] Eidini, M. & Paulino, G. H. Unraveling metamaterial properties in zigzag-base folded sheets. *Science Advances* **1**, e1500224 (2015).