Three Problems in Discrete Geometry

Giovanni Viglietta

JAIST – April 19, 2018

Self-introduction: Research areas

Discrete and Computational Geometry

- Polyhedral Combinatorics
- Visibility-related problems
- Folding and Unfolding

Complexity of Games

- General techniques applied to video games
- Puzzles, board games, card games, etc.

Distributed Computing

- Motion Planning for Swarms of Robots
- Computing in Dynamic Networks
- Population Protocols

Self-introduction: Research areas

Discrete and Computational Geometry

- Polyhedral Combinatorics
- Visibility-related problems
- Folding and Unfolding

Complexity of Games

- General techniques applied to video games
- Puzzles, board games, card games, etc.

Distributed Computing

- Motion Planning for Swarms of Robots
- Computing in Dynamic Networks
- Population Protocols

The Shadows of a Cycle Cannot All Be Paths

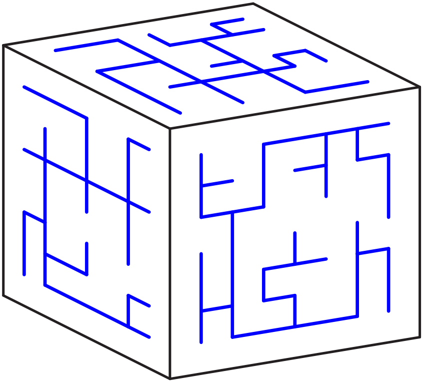
CCCG 2015

Joint work with P. Bose, J.-L. De Carufel,
M. G. Dobbins, and H. Kim

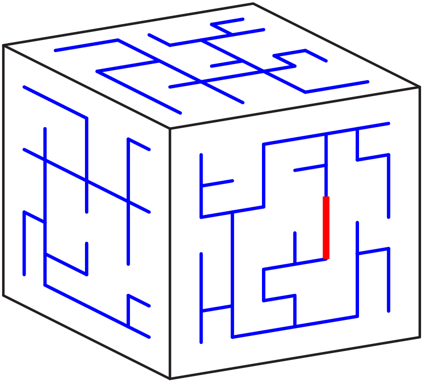
Oskar's maze

A 3D maze designed in the 1980s by Oskar van Deventer.
To move the rods around, one has to solve three 2D mazes.

Oskar's maze

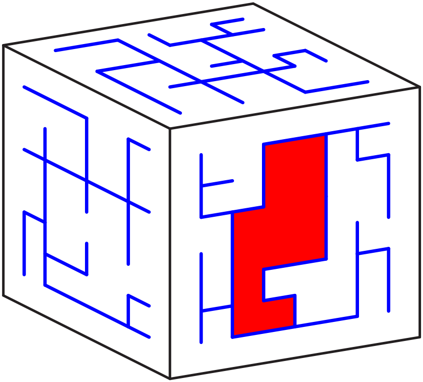


Observation: the maze on each face must be a tree.

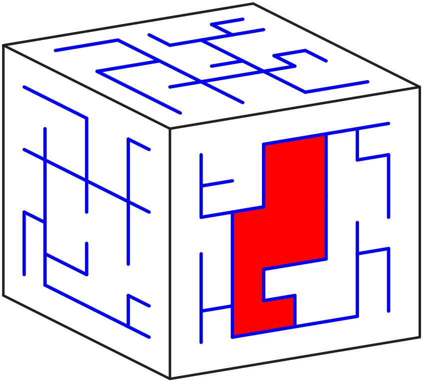


If there were a cycle, part of the structure would fall off.

Oskar's maze

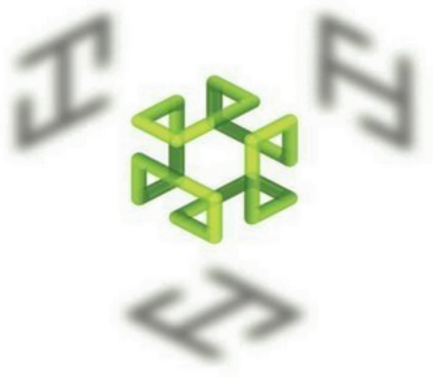


If there were a cycle, part of the structure would fall off.



But can there be cycles in the "internal" 3D maze?

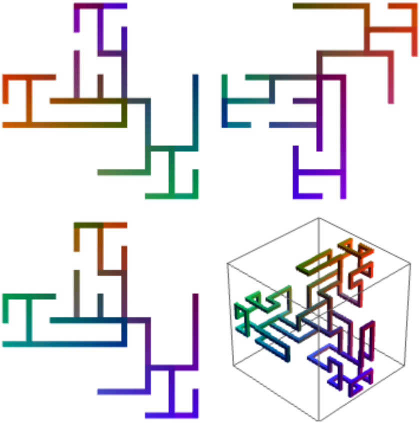
Rickard's curve



A 3D cycle whose shadows are all trees.

(Illustration by Afra Zomorodian.)

Goucher's "treefoil"

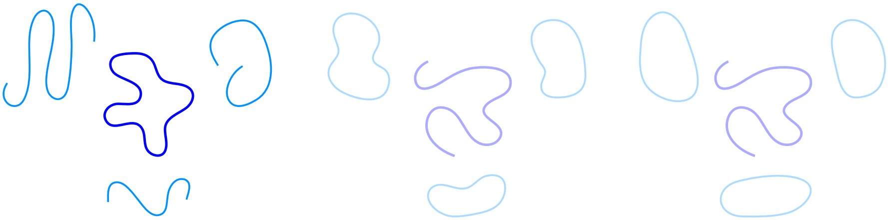


A trefoil knot whose shadows are all trees.

(Illustration courtesy of Adam P. Goucher.)

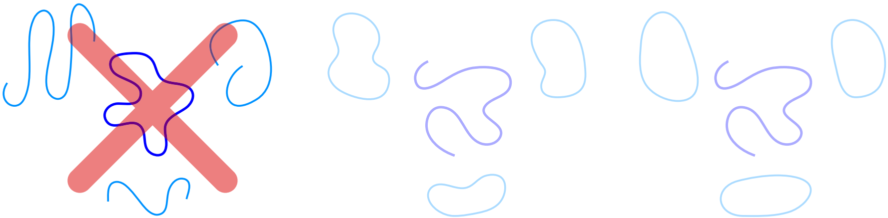
In this talk...

- Can the shadows of a 3D cycle be all paths?
  (Open problem from CCCG 2007)



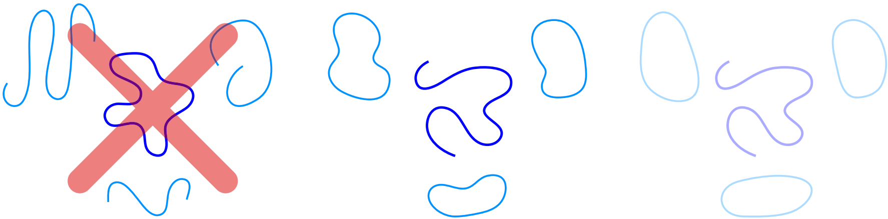
In this talk...

- Can the shadows of a 3D cycle be all paths? **NO.**
  (Open problem from CCCG 2007)



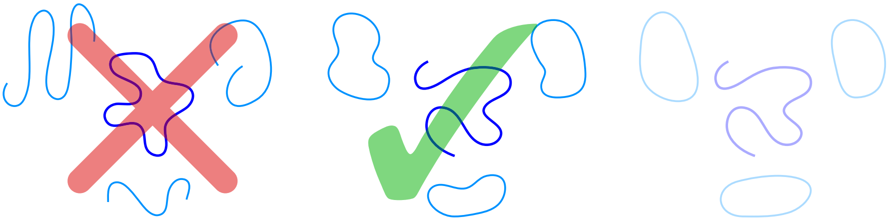
In this talk...

- Can the shadows of a 3D cycle be all paths? **NO.**
  (Open problem from CCCG 2007)
- Can the shadows of a 3D path be all cycles?



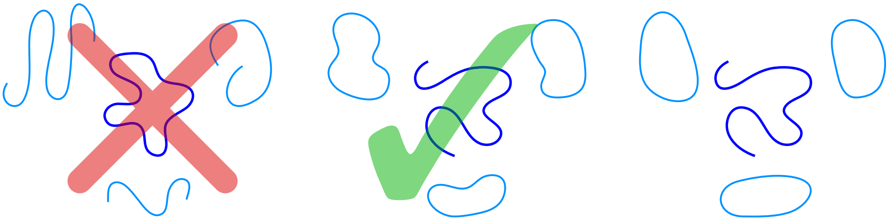
In this talk...

- Can the shadows of a 3D cycle be all paths? **NO.**
  (Open problem from CCCG 2007)
- Can the shadows of a 3D path be all cycles? **YES.**



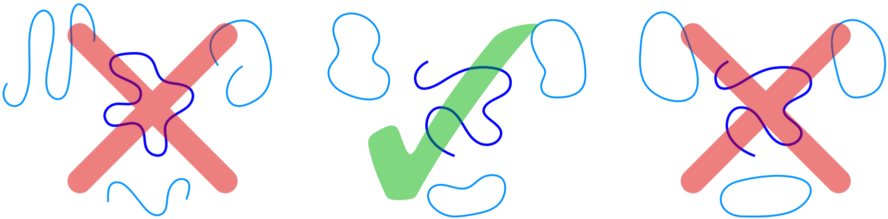
In this talk...

- Can the shadows of a 3D cycle be all paths? **NO.**
  (Open problem from CCCG 2007)
- Can the shadows of a 3D path be all cycles? **YES.**
- Can the shadows of a 3D path be all convex cycles?



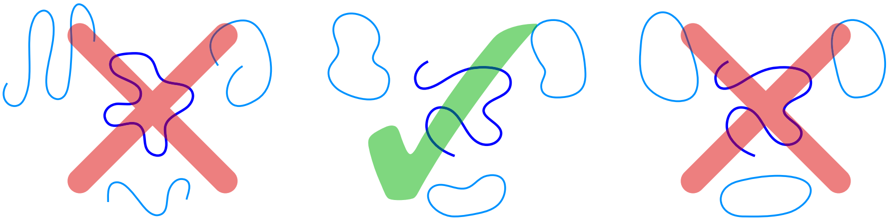
In this talk...

- Can the shadows of a 3D <u>cycle</u> be all <u>paths</u>? **NO.**
  (Open problem from CCCG 2007)
- Can the shadows of a 3D <u>path</u> be all <u>cycles</u>? **YES.**
- Can the shadows of a 3D <u>path</u> be all <u>convex cycles</u>? **NO.**



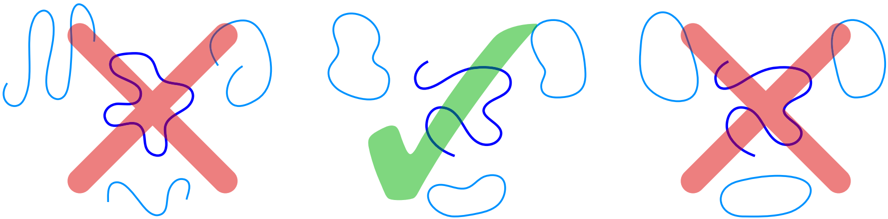
In this talk...

- Can the shadows of a 3D cycle be all paths? **NO.**
  (Open problem from CCCG 2007)
- Can the shadows of a 3D path be all cycles? **YES.**
- Can the shadows of a 3D path be all convex cycles? **NO.**
- What about higher dimensions?

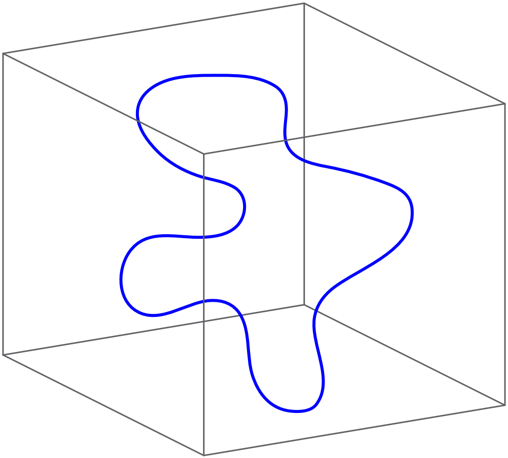


In this talk...

- Can the shadows of a 3D cycle be all paths? **NO.**
  (Open problem from CCCG 2007)
- Can the shadows of a 3D path be all cycles? **YES.**
- Can the shadows of a 3D path be all convex cycles? **NO.**
- What about higher dimensions?
  Rickard's curve generalizes to any dimension.

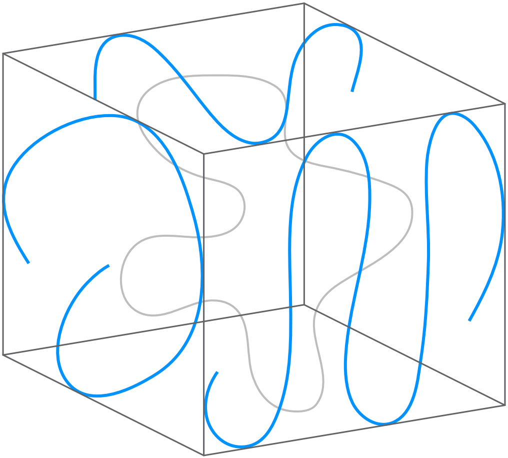


The shadows of a 3D cycle cannot all be paths



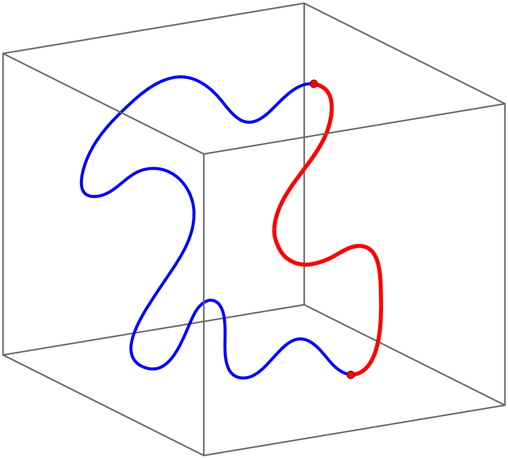
Suppose that the shadows of a 3D cycle are all paths.

The shadows of a 3D cycle cannot all be paths



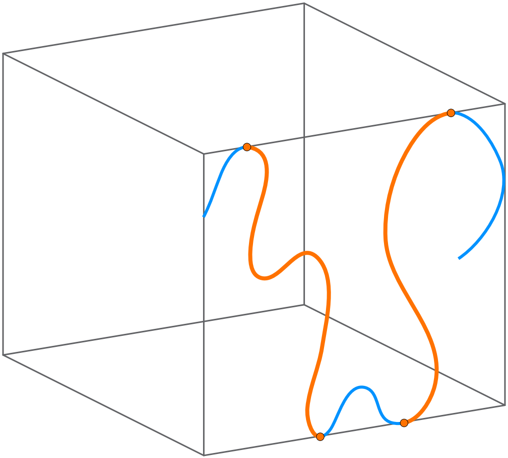
Suppose that the shadows of a 3D cycle are all paths.

The shadows of a 3D cycle cannot all be paths



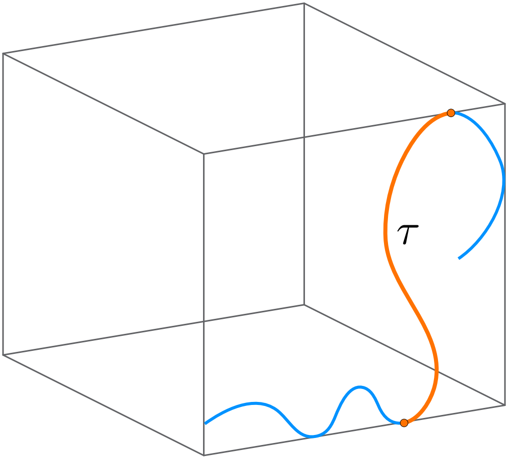
Definition: a *strand* is a minimal sub-curve connecting the top and bottom faces of the bonding box.

The shadows of a 3D cycle cannot all be paths



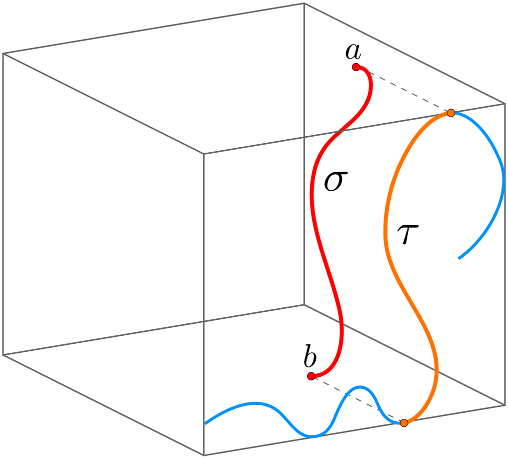
Claim: a lateral shadow has two (internally disjoint) strands.

The shadows of a 3D cycle cannot all be paths



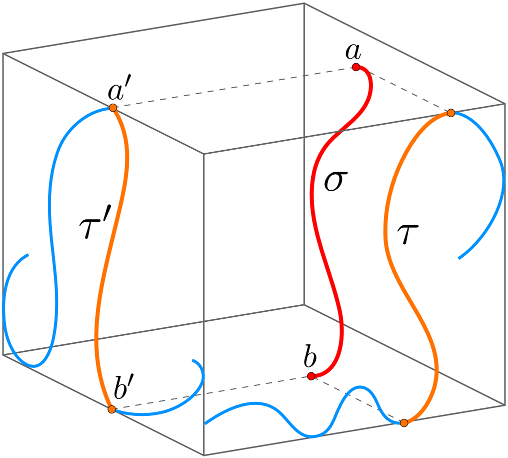
Suppose it has a unique strand τ.

The shadows of a 3D cycle cannot all be paths



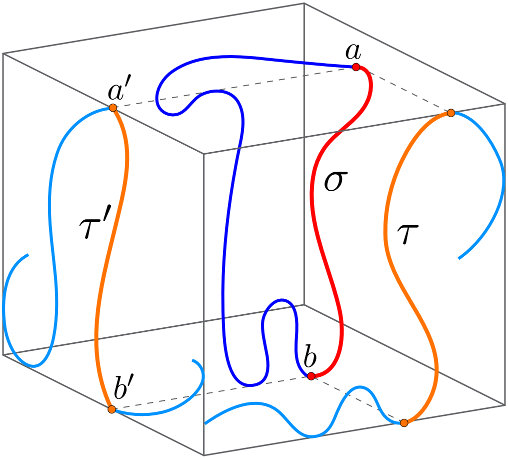
Then, the shadow of any strand σ of the 3D cycle is τ.

The shadows of a 3D cycle cannot all be paths



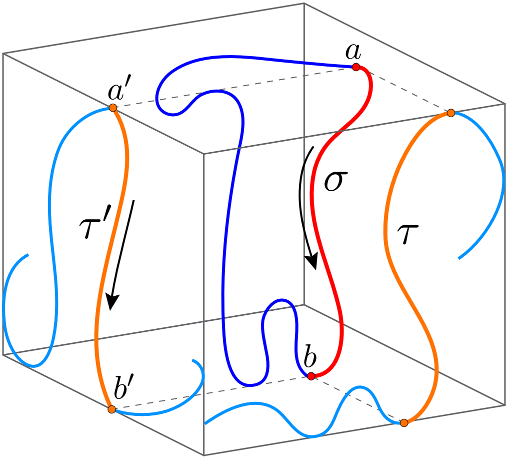
Let τ' be the other lateral shadow of σ.

The shadows of a 3D cycle cannot all be paths



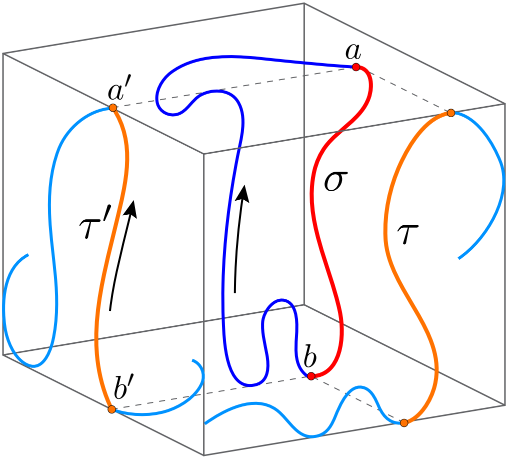
Let τ' be the other lateral shadow of σ.

The shadows of a 3D cycle cannot all be paths



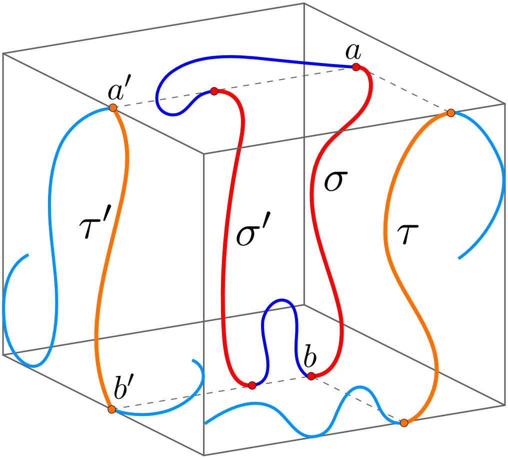
The shadow of a point moving from a to b moves from a' to b'.

The shadows of a 3D cycle cannot all be paths



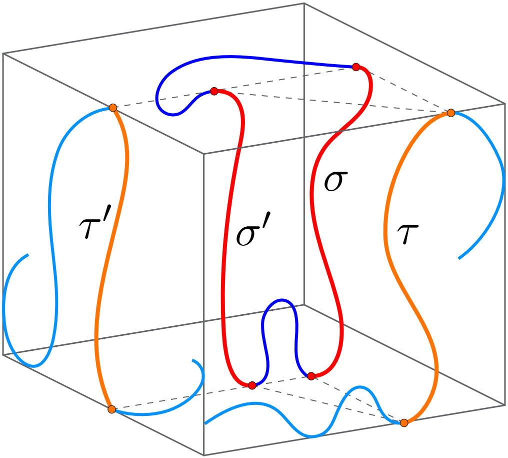
As it keeps moving, its shadow traverses τ' again, from b' to a'.

The shadows of a 3D cycle cannot all be paths



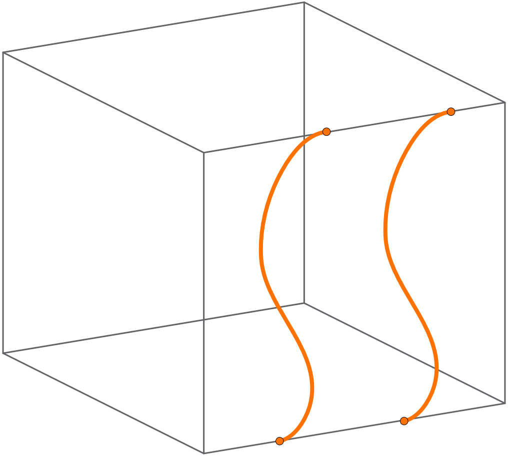
Thus a second strand σ' is found, whose shadow is again τ'.

The shadows of a 3D cycle cannot all be paths



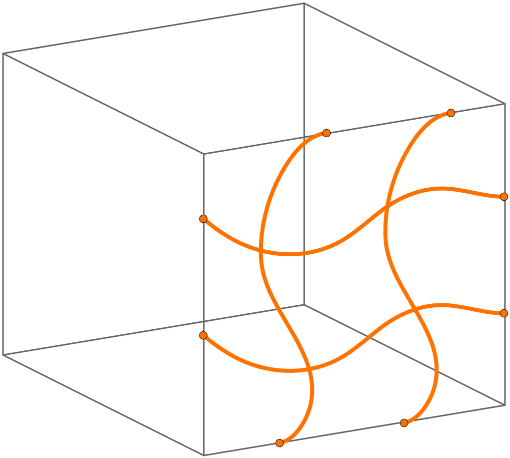
But the shadows of σ and σ' cannot both be τ. Contradiction!

The shadows of a 3D cycle cannot all be paths



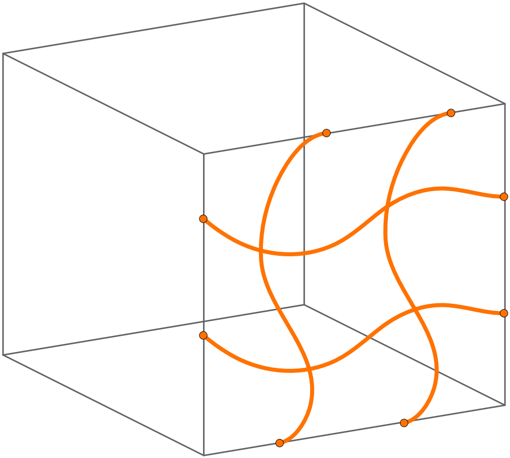
Hence a shadow has two (internally disjoint) vertical strands.

The shadows of a 3D cycle cannot all be paths



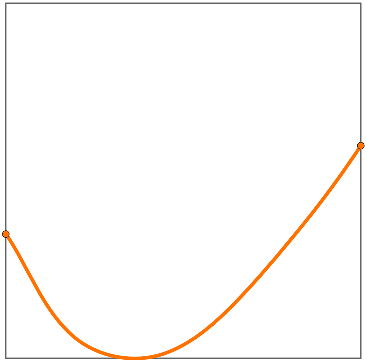
By a symmetric argument, it also has two horizontal strands.

The shadows of a 3D cycle cannot all be paths



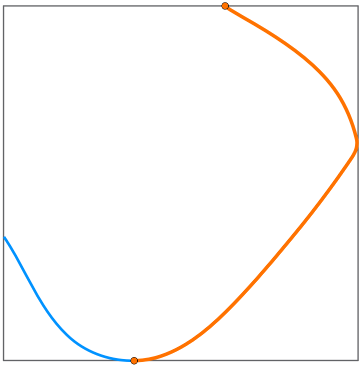
Claim: no 2D path has two vertical and two horizontal strands.

The shadows of a 3D cycle cannot all be paths



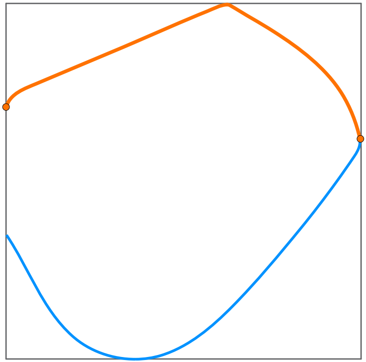
Start with a horizontal strand.

The shadows of a 3D cycle cannot all be paths



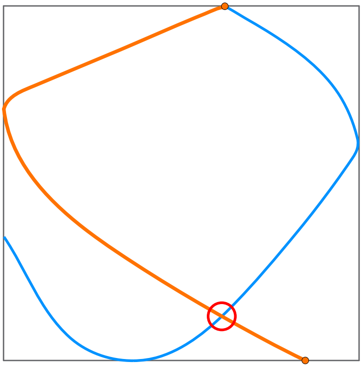
Extend it to a vertical strand.

The shadows of a 3D cycle cannot all be paths



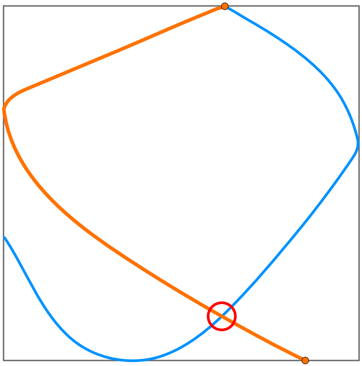
Extend it with a second horizontal strand.

The shadows of a 3D cycle cannot all be paths



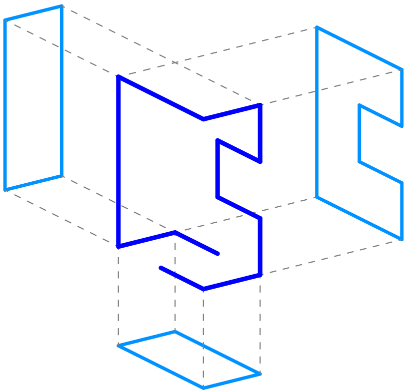
If a second vertical strand is drawn, the curve self-intersects.

The shadows of a 3D cycle cannot all be paths



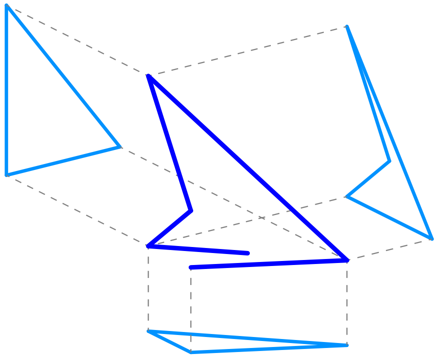
The other cases are similar...

The shadows of a 3D path can be all cycles



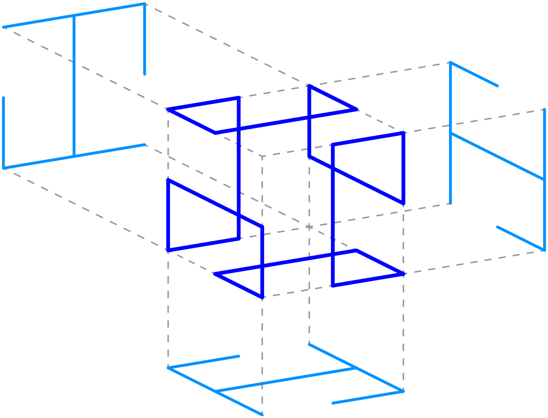
An orthogonal chain whose shadows are polygons.

The shadows of a 3D path can be all cycles



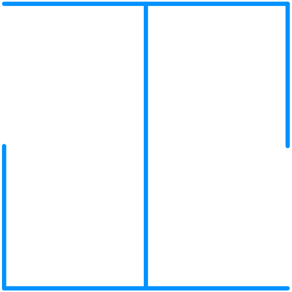
A 5-segment chain whose shadows are polygons.
(Smallest possible!)

Generalizing Rickard's curve to higher dimensions



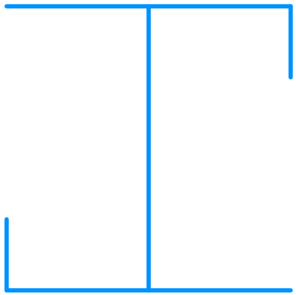
What does it mean to generalize Rickard's curve?

Generalizing Rickard's curve to higher dimensions



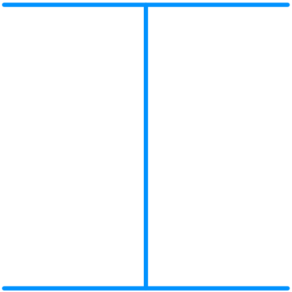
Note that the shadows of Rickard's curve are *contractible*
(i.e., they deformation-retract to a point)...

Generalizing Rickard's curve to higher dimensions



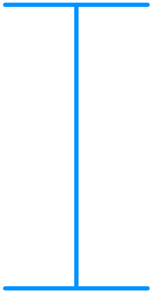
Note that the shadows of Rickard's curve are *contractible*
(i.e., they deformation-retract to a point)...

Generalizing Rickard's curve to higher dimensions



Note that the shadows of Rickard's curve are *contractible*
(i.e., they deformation-retract to a point)...

Generalizing Rickard's curve to higher dimensions



Note that the shadows of Rickard's curve are *contractible*
(i.e., they deformation-retract to a point)...

Generalizing Rickard's curve to higher dimensions

Note that the shadows of Rickard's curve are *contractible*
(i.e., they deformation-retract to a point)...

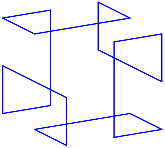
Generalizing Rickard's curve to higher dimensions

Note that the shadows of Rickard's curve are *contractible*
(i.e., they deformation-retract to a point)...

Generalizing Rickard's curve to higher dimensions

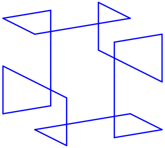
Note that the shadows of Rickard's curve are *contractible*
(i.e., they deformation-retract to a point)...

Generalizing Rickard's curve to higher dimensions



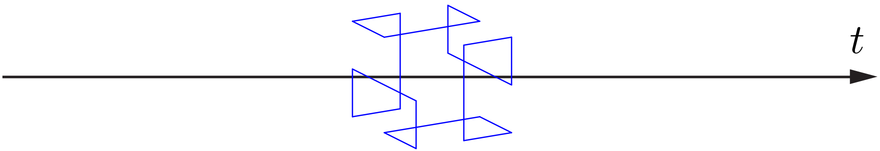
...While Rickard's curve, being a 1-sphere, is not contractible.

Generalizing Rickard's curve to higher dimensions



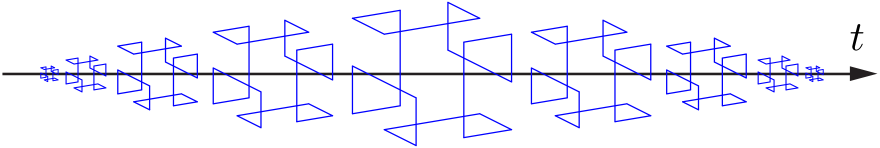
Claim: there is a 2-sphere in $\mathbb{R}^4$ whose shadows are contractible.

Generalizing Rickard's curve to higher dimensions



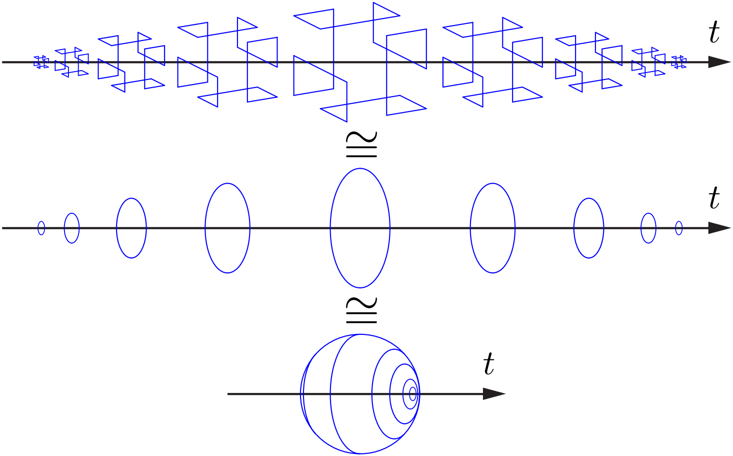
Think of the 4-dimensional space as a function of time.

Generalizing Rickard's curve to higher dimensions



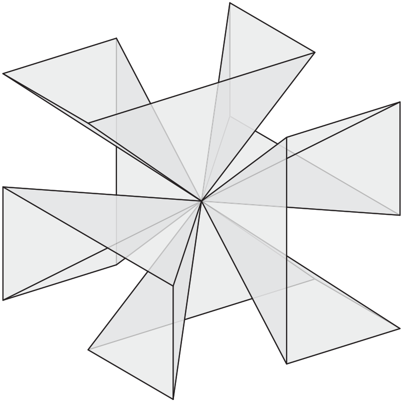
In each 3D frame, put a scaled copy of Rickard's curve...

Generalizing Rickard's curve to higher dimensions



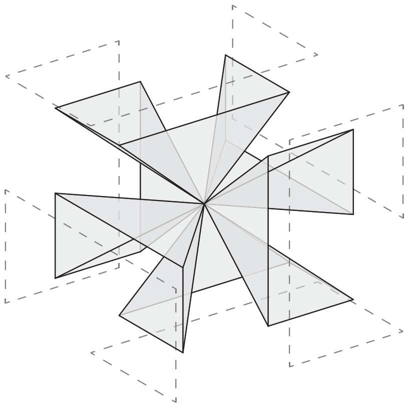
...So that the union of all frames is homeomorphic to a 2-sphere.

Generalizing Rickard's curve to higher dimensions



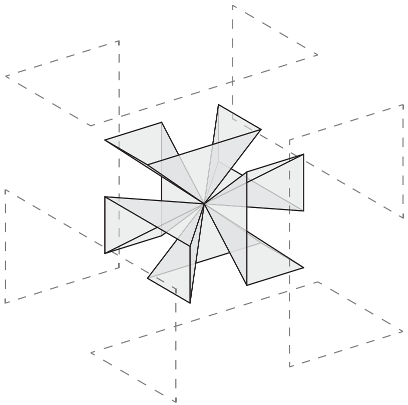
The t-orthogonal shadow is the superimposition of all frames...

Generalizing Rickard's curve to higher dimensions



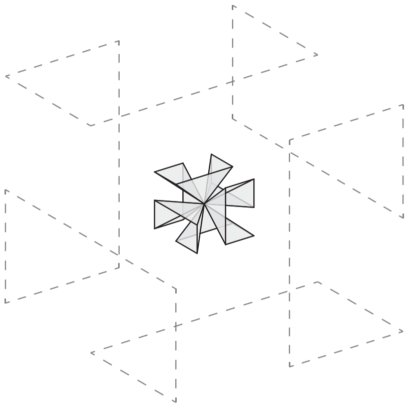
...Which is contractible.

Generalizing Rickard's curve to higher dimensions



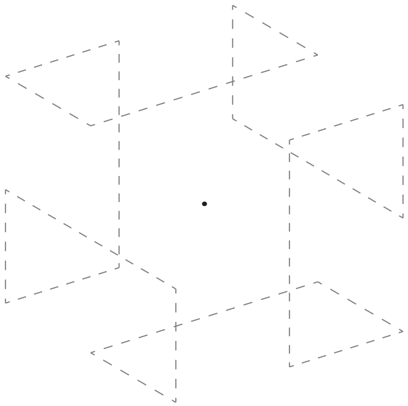
...Which is contractible.

Generalizing Rickard's curve to higher dimensions



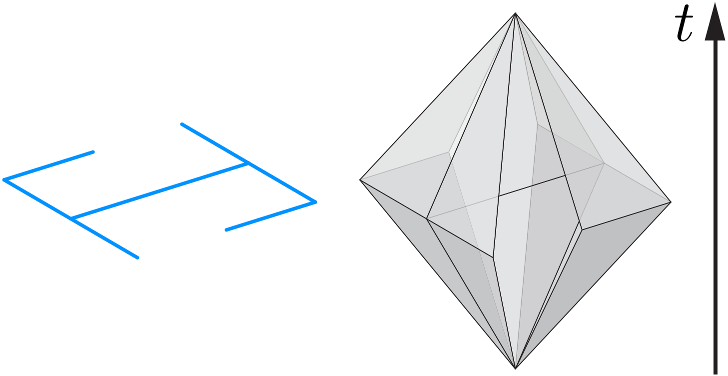
...Which is contractible.

Generalizing Rickard's curve to higher dimensions



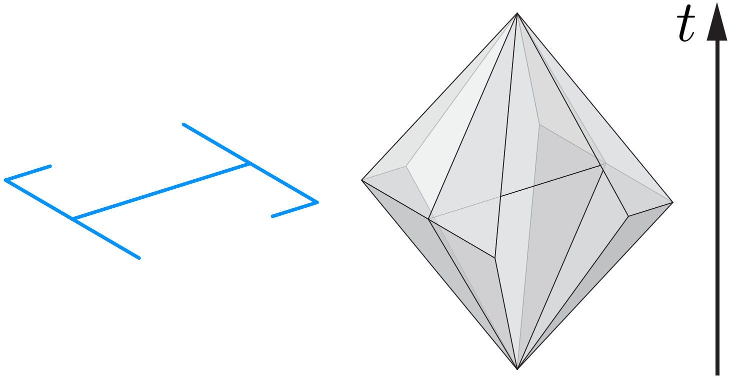
...Which is contractible.

Generalizing Rickard's curve to higher dimensions



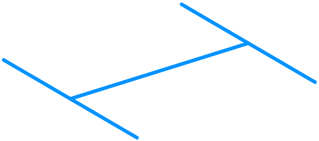
In the other three shadows, each t-orthogonal slice is a scaled copy of a shadow of Rickard's curve.

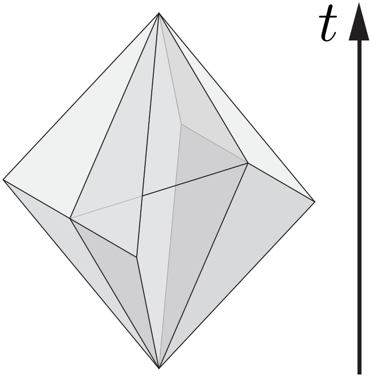
Generalizing Rickard's curve to higher dimensions



To contract it, first contract all slices simultaneously...

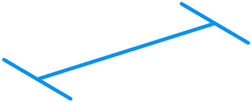
Generalizing Rickard's curve to higher dimensions

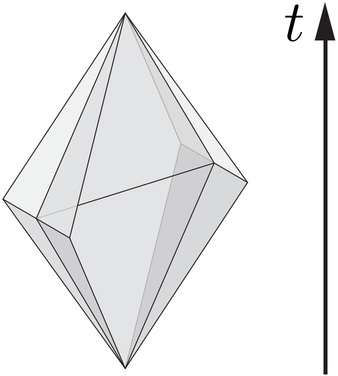




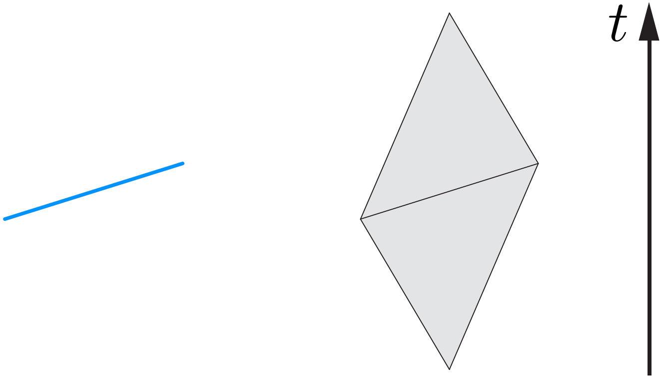
To contract it, first contract all slices simultaneously...

Generalizing Rickard's curve to higher dimensions



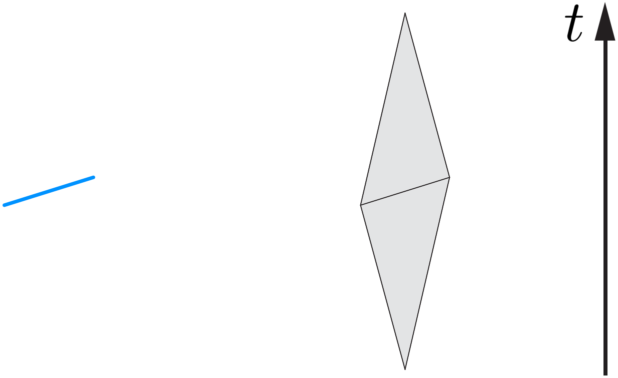


To contract it, first contract all slices simultaneously...



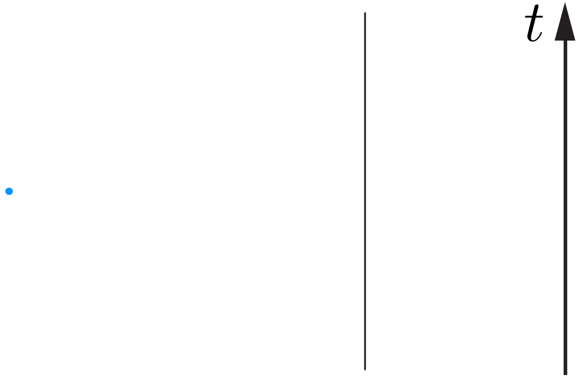
To contract it, first contract all slices simultaneously...

Generalizing Rickard's curve to higher dimensions



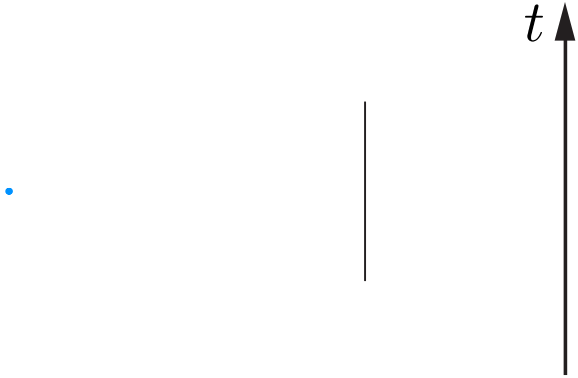
To contract it, first contract all slices simultaneously...

Generalizing Rickard's curve to higher dimensions



...Then contract the resulting segment.

Generalizing Rickard's curve to higher dimensions



...Then contract the resulting segment.

…Then contract the resulting segment.

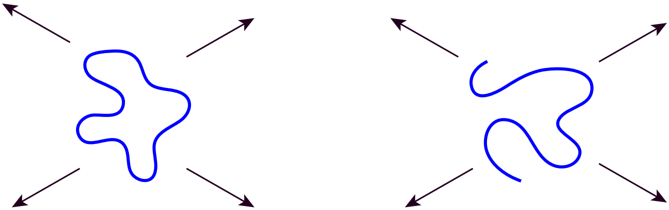
Generalizing Rickard's curve to higher dimensions

By induction, for every $d > 0$, we can construct a d-sphere in $\mathbb{R}^{d+2}$ with contractible shadows.

Summary

- The shadows of a cycle in $\mathbb{R}^3$:
  - can be all trees
  - cannot be all paths
- The shadows of a path in $\mathbb{R}^3$:
  - can be all cycles
  - cannot be all convex cycles
- The shadows of a d-sphere in $\mathbb{R}^{d+2}$:
  - can be all contractible

Open problem



Problem: what if the curves cast four shadows instead of three?
Can the four shadows of a cycle be trees?
Can the four shadows of a path be cycles?

Algorithms for Designing Pop-Up Cards

STACS 2013

Joint work with Z. Abel, E. D. Demaine,

M. L. Demaine, S. Eisenstat, A. Lebiw,

A. Schulz, D. L. Souvaine, and A. Winslow

Pop-up cards

- Pop-up cards (or books) are 3D paper models that fold flat with one degree of freedom.

(The Jungle Book: A Pop-Up Adventure, by Matthew Reinhart)

Pop-up cards

- Pop-up cards (or books) are 3D paper models that fold flat with one degree of freedom.
- *Can every possible shape be modeled as a pop-up card, and how efficiently?*

(The Jungle Book: A Pop-Up Adventure, by Matthew Reinhart)

Pop-up cards

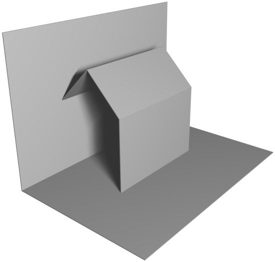
- Pop-up cards (or books) are 3D paper models that fold flat with one degree of freedom.
- *Can every possible shape be modeled as a pop-up card, and how efficiently?*
- We are not concerned with practical realizability (e.g., paper thickness, feature size).

(The Jungle Book: A Pop-Up Adventure, by Matthew Reinhart)

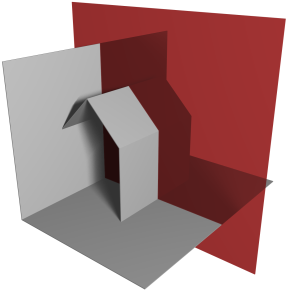
Outline

- 2D orthogonal polygon pop-ups, $O(n)$ links.
- 2D general polygon pop-ups, $O(n^2)$ links.
- 3D orthogonal polyhedron pop-ups, $O(n^3)$ links.

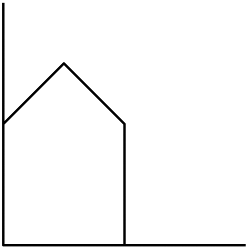
2D model for pop-ups



Desired card



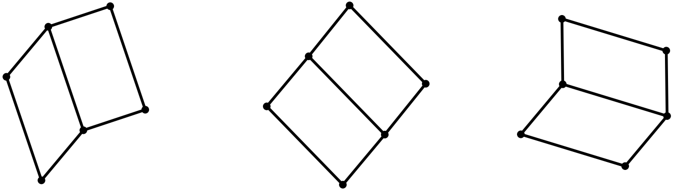
Cross section



2D model

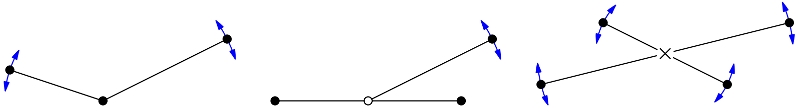
Linkages

- Linkages are formed by rigid *bars* and flexible *joints*.
- If bars intersect only at joints, the linkage configuration is called *non-crossing*.



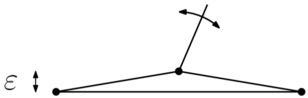
Three non-crossing configurations of a 7-bar linkage.

More general joints

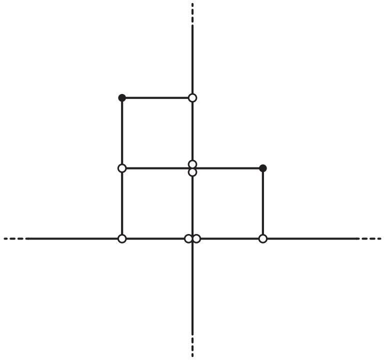


Common joint: $\bullet$ Flap: $\circ$ Sliceform: $\times$

Everything is a joint



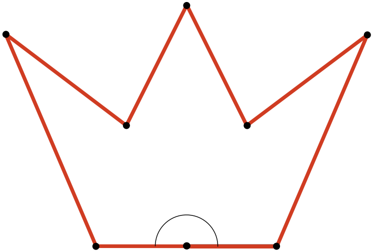
Flap made of joints



Sliceform made of flaps + joints

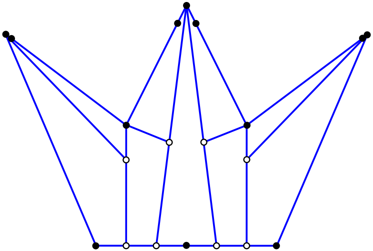
Problem formulation

- **Input:** 2D polygon P (unfolded shape):
  n vertices in total, one distinguished vertex (the "fold").



Problem formulation

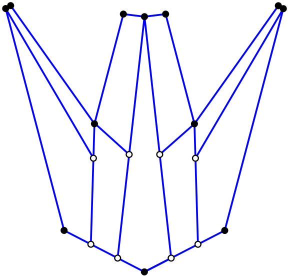
- **Input:** 2D polygon P (unfolded shape):
  n vertices in total, one distinguished vertex (the "fold").
- **Output:** linkage structure L with external boundary P.
  - L folds around the distinguished vertex to form a line,
  - L is non-crossing throughout,
  - L has one degree of freedom.



(Pop-up designed using algorithm by Hara and Sugihara, 2009)

Problem formulation

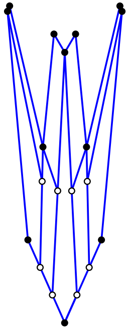
- **Input:** 2D polygon P (unfolded shape):
  n vertices in total, one distinguished vertex (the "fold").
- **Output:** linkage structure L with external boundary P.
  - L folds around the distinguished vertex to form a line,
  - L is non-crossing throughout,
  - L has one degree of freedom.



(Pop-up designed using algorithm by Hara and Sugihara, 2009)

Problem formulation

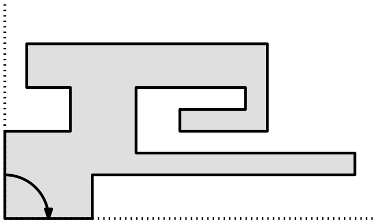
- **Input:** 2D polygon P (unfolded shape):
  n vertices in total, one distinguished vertex (the "fold").
- **Output:** linkage structure L with external boundary P.
  - L folds around the distinguished vertex to form a line,
  - L is non-crossing throughout,
  - L has one degree of freedom.



(Pop-up designed using algorithm by Hara and Sugihara, 2009)

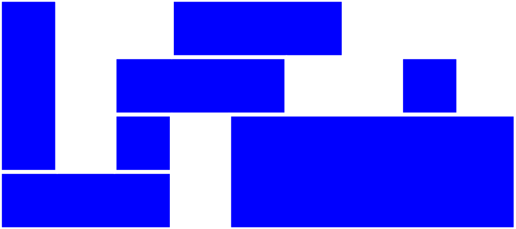
Orthogonal polygons

- *P* orthogonal: every edge is either vertical or horizontal.



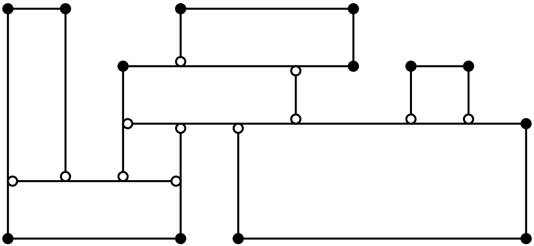
- Opening angle is 90° (angles 180°, 270°, and 360° are discussed later).
- **Strategy:** preserve parallelism throughout the motion (i.e., *shearing* motion).

3-step construction



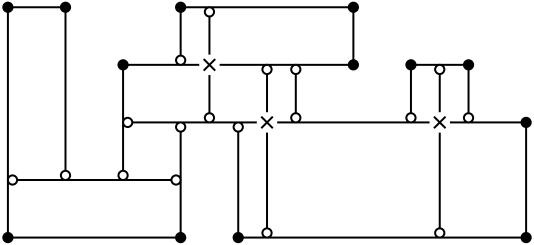
- Subdivide P into horizontal stripes.

3-step construction



- Subdivide P into horizontal stripes.
- Model all degree-3 vertices as flaps.

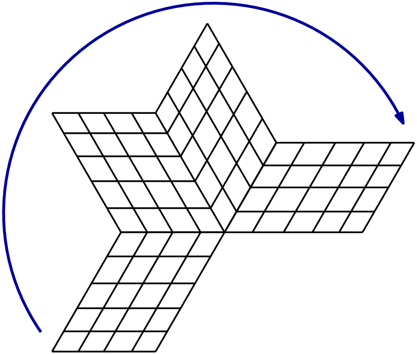
3-step construction



- Subdivide P into horizontal stripes.
- Model all degree-3 vertices as flaps.
- Enforce a 1-dof motion by adding vertical bars connected by sliceforms.

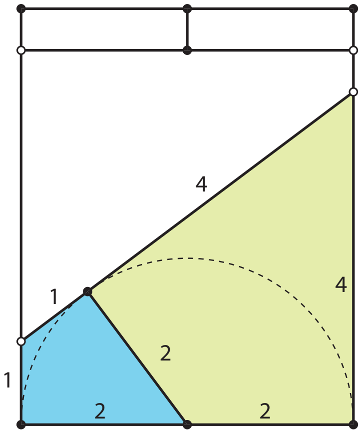
Larger opening angle

- **Strategy:** combine the 90° shearing motions.
- Need to “reflect” the shear.



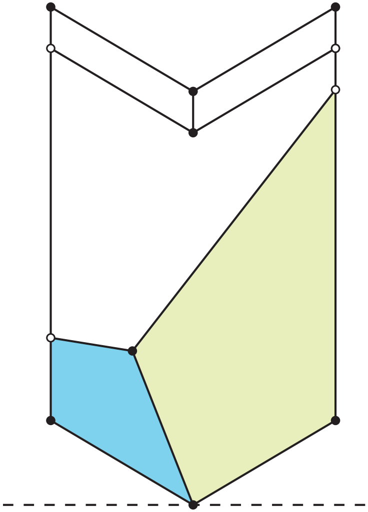
Reflector gadget

- The top part keeps the vertical lines parallel.
- The two kites are similar and force the left and right halves to move symmetrically.

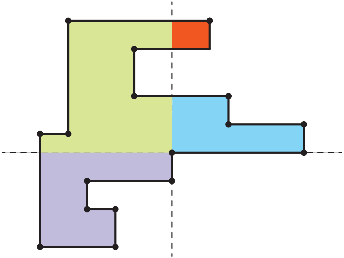


Reflector gadget

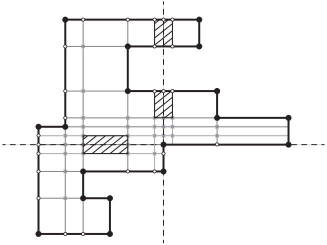
- The top part keeps the vertical lines parallel.
- The two kites are similar and force the left and right halves to move symmetrically.



Synchronizing shears



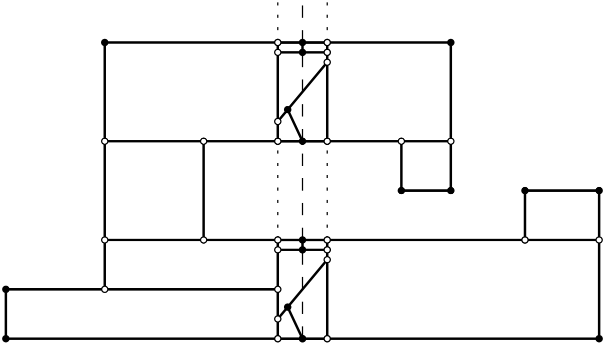
Cut P along the x and y axes.



Reconnect the 90° solutions via reflector gadgets.

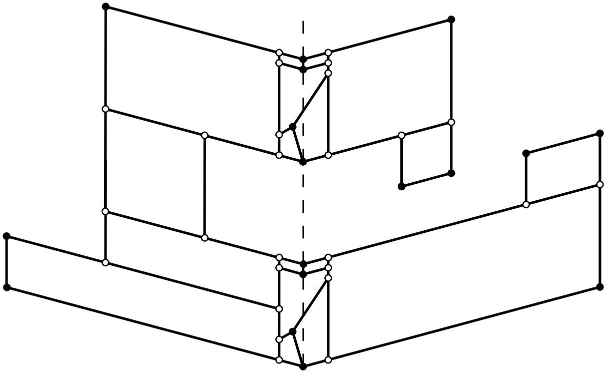
Result

- The resulting structure has complexity $O(n)$.



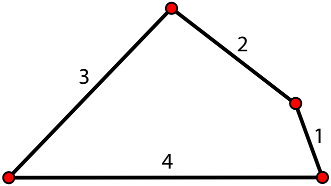
Result

- The resulting structure has complexity $O(n)$.



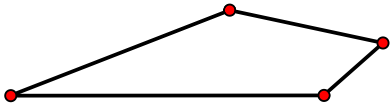
General polygons: V-folds

- **Outward V-fold:**
  $1 + 4 = 2 + 3$.



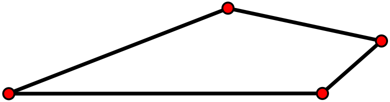
General polygons: V-folds

- **Outward V-fold:**
  $1 + 4 = 2 + 3.$

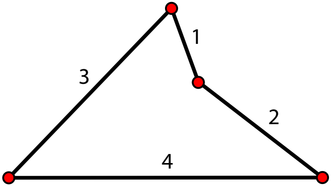


General polygons: V-folds

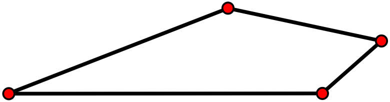
- **Outward V-fold:**
  $1 + 4 = 2 + 3.$



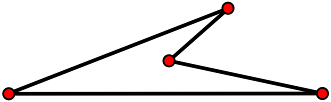
- **Inward V-fold:**
  $3 - 1 = 4 - 2.$



- **Outward V-fold:**
  $1 + 4 = 2 + 3.$



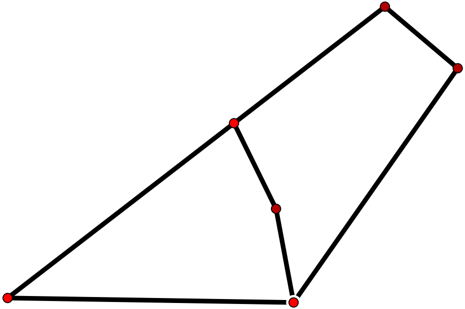
- **Inward V-fold:**
  $3 - 1 = 4 - 2.$



Nested V-folds

Lemma

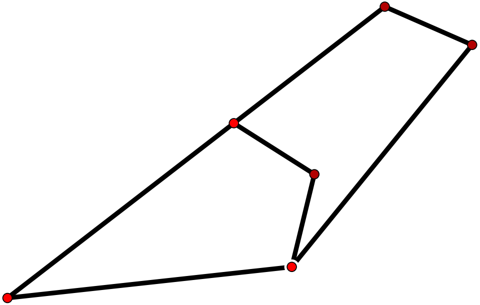
The closing motion of nested outward (resp. inward) V-folds intersects only in the end configuration.



Nested V-folds

Lemma

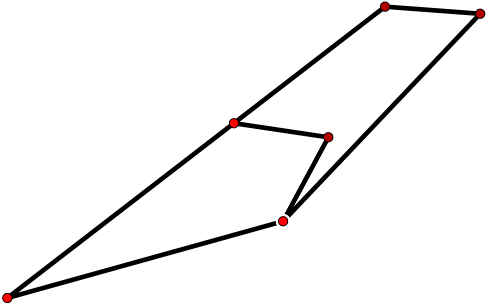
The closing motion of nested outward (resp. inward) V-folds intersects only in the end configuration.



Nested V-folds

Lemma

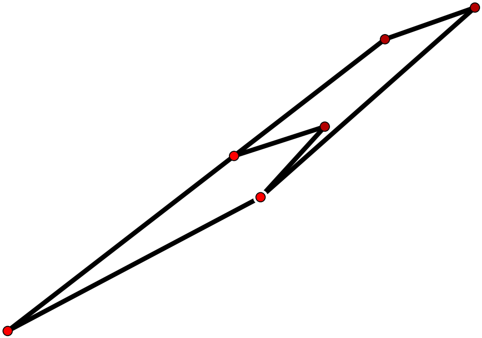
The closing motion of nested outward (resp. inward) V-folds intersects only in the end configuration.



Nested V-folds

Lemma

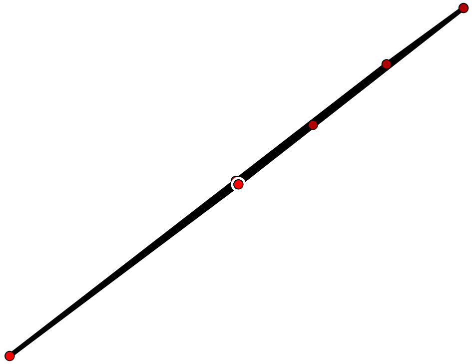
The closing motion of nested outward (resp. inward) V-folds intersects only in the end configuration.



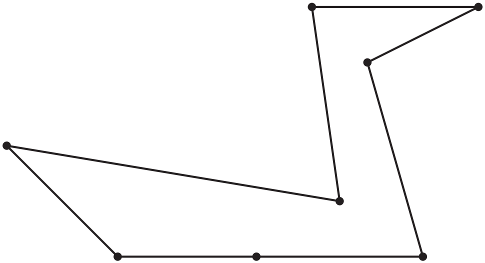
Nested V-folds

Lemma

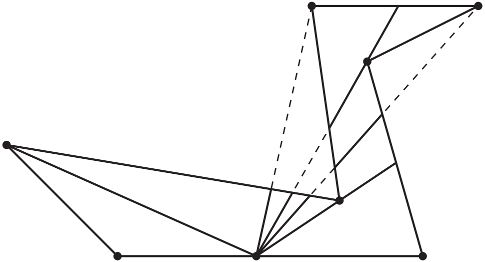
The closing motion of nested outward (resp. inward) V-folds intersects only in the end configuration.



Cell decomposition

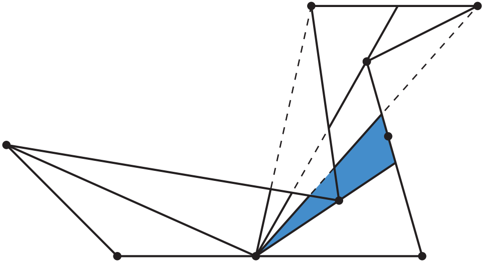


Cell decomposition



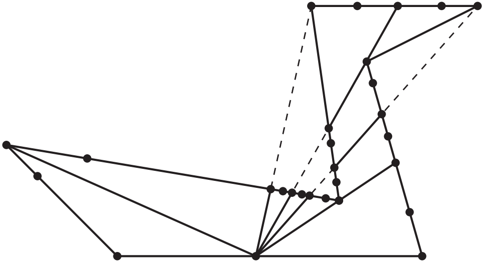
- Draw a ray from the fold to every vertex of P.

Cell decomposition



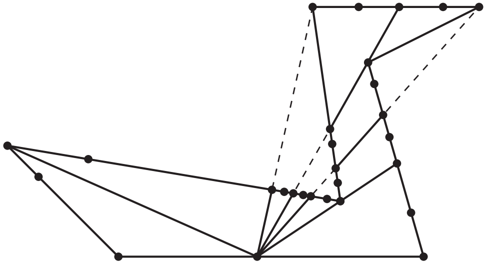
- Draw a ray from the fold to every vertex of P.
- Make outward V-folds for all edges between rays.

Cell decomposition



- Draw a ray from the fold to every vertex of P.
- Make outward V-folds for all edges between rays.

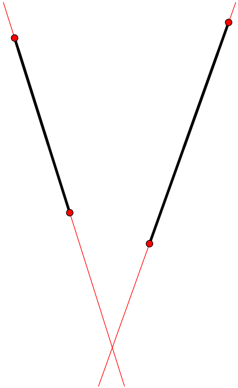
Cell decomposition



- Draw a ray from the fold to every vertex of P.
- Make outward V-folds for all edges between rays.
- Every wedge can be folded flat, but there are too many dof!
  - Want: wall segments rotate around fold.
  - Want: wedge motions be synchronized.

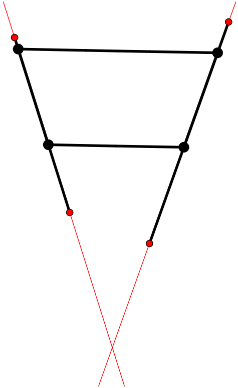
Restricting to rotations

- For each pair of wall segment in an *internal* cell:



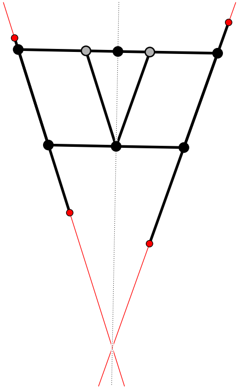
Restricting to rotations

- For each pair of wall segment in an *internal* cell:
  - Add two parallel segments.



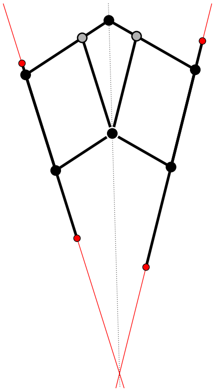
Restricting to rotations

- For each pair of wall segment in an *internal* cell:
  - Add two parallel segments.
  - Add two parallelograms to get two outward V-folds.



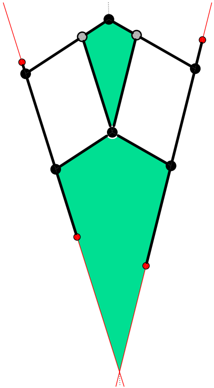
Restricting to rotations

- For each pair of wall segment in an *internal* cell:
  - Add two parallel segments.
  - Add two parallelograms to get two outward V-folds.
- Result: wall segments rotate around the apex, even if they are not connected to it.



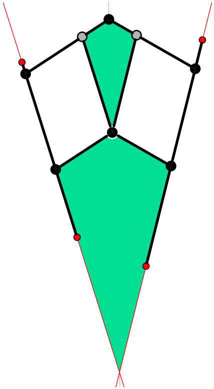
Restricting to rotations

- For each pair of wall segment in an *internal* cell:
  - Add two parallel segments.
  - Add two parallelograms to get two outward V-folds.
- Result: wall segments rotate around the apex, even if they are not connected to it.



Restricting to rotations

- For each pair of wall segment in an *internal* cell:
  - Add two parallel segments.
  - Add two parallelograms to get two outward V-folds.
- Result: wall segments rotate around the apex, even if they are not connected to it.
- (*Leaf* cells are handled separately.)

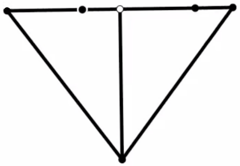


Synchronizing wedges

- **Strategy:** link neighboring cells with with a gadget that synchronizes the independent motions of the wedges.

Synchronizing wedges

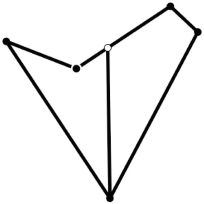
- **Strategy:** link neighboring cells with with a gadget that synchronizes the independent motions of the wedges.



- **Basic sync gadget:**
  inward V-fold + outward V-fold.

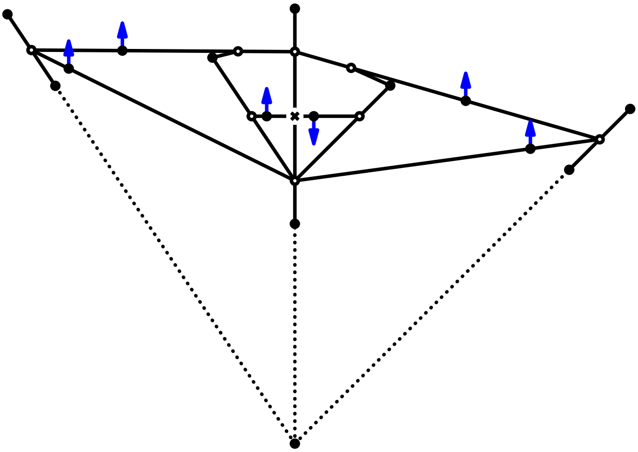
Synchronizing wedges

- **Strategy:** link neighboring cells with with a gadget that synchronizes the independent motions of the wedges.
- **Basic sync gadget:**
  inward V-fold + outward V-fold.



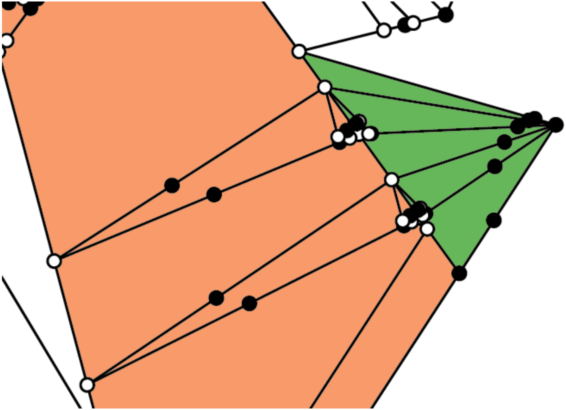
- The basic sync gadget has a 1-dof motion that makes all the cells in the same wedge fold at the same speed.

Fitting the sync gadget



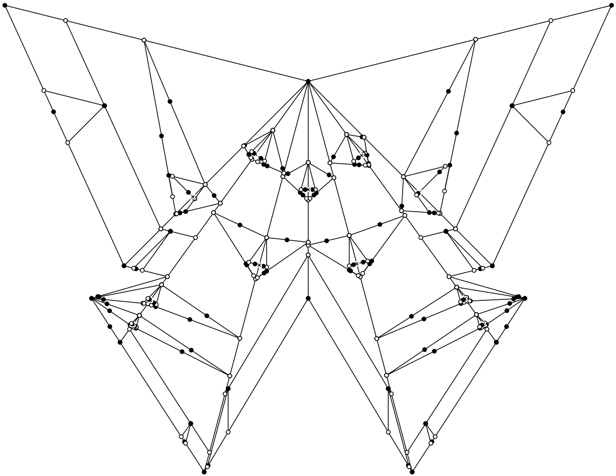
Folding leaf cells

- For cells with only one wall, use two sync gadgets and no rotation gadget.



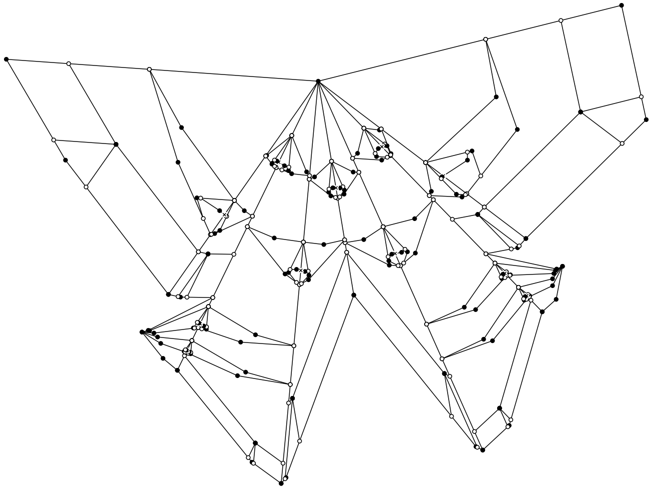
Result

- The resulting structure has complexity $O(n^2)$.



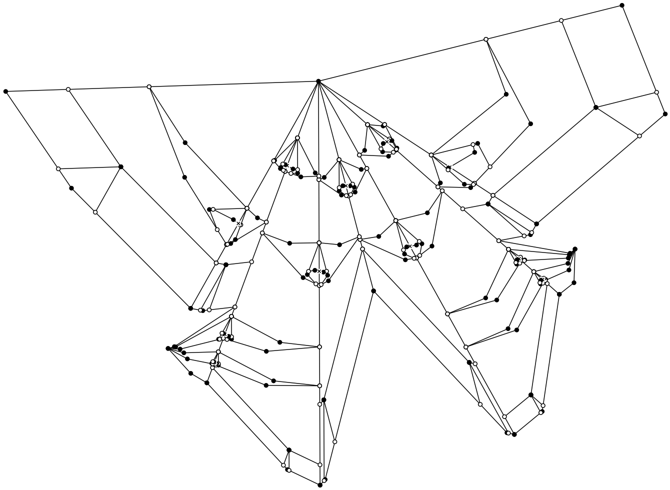
Result

- The resulting structure has complexity $O(n^2)$.



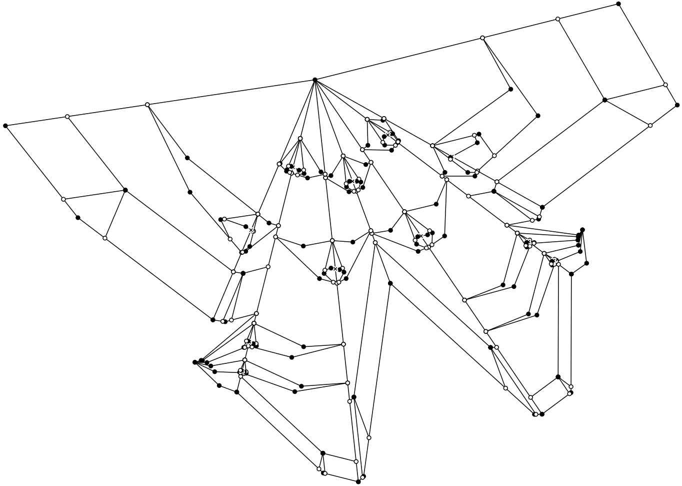
Result

- The resulting structure has complexity $O(n^2)$.



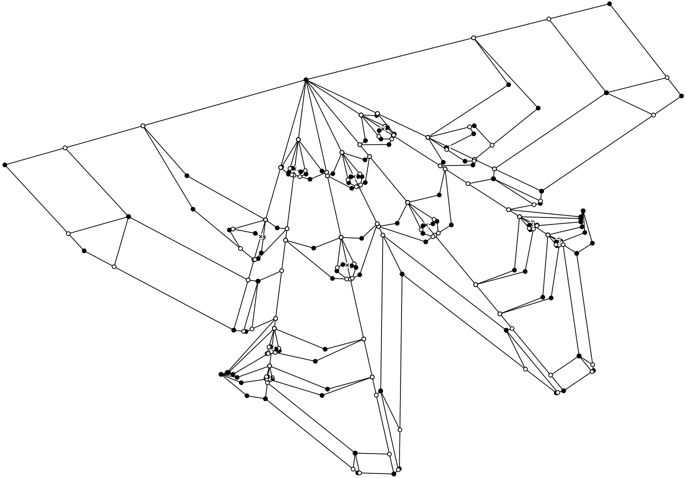
Result

- The resulting structure has complexity $O(n^2)$.



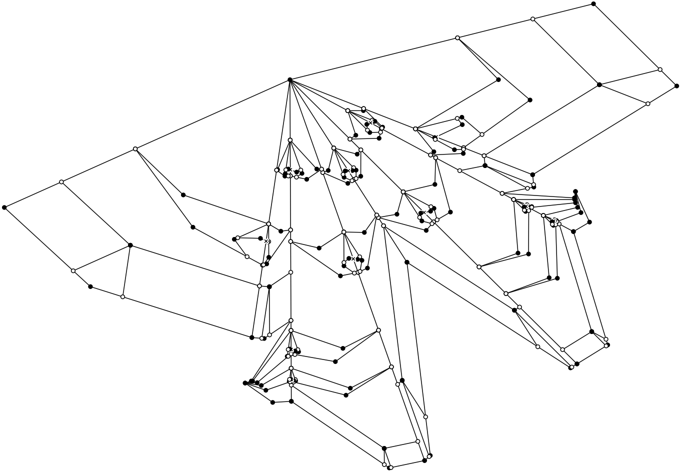
Result

- The resulting structure has complexity $O(n^2)$.



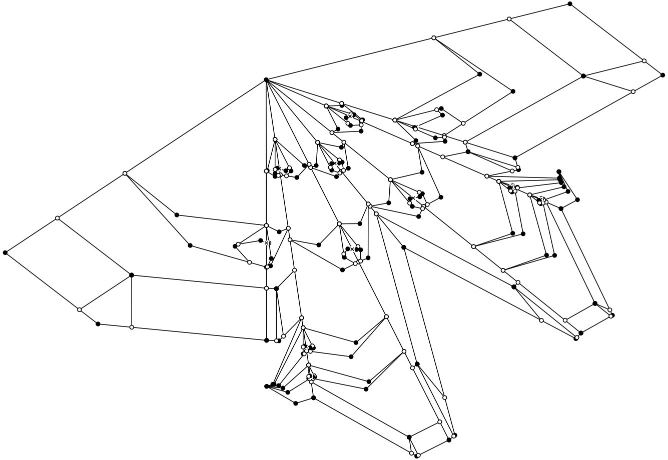
Result

- The resulting structure has complexity $O(n^2)$.



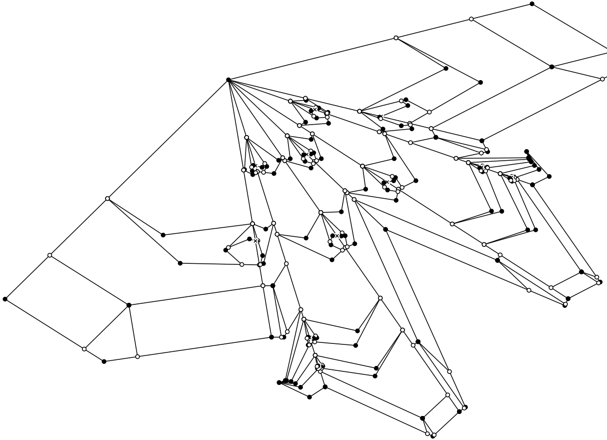
Result

- The resulting structure has complexity $O(n^2)$.



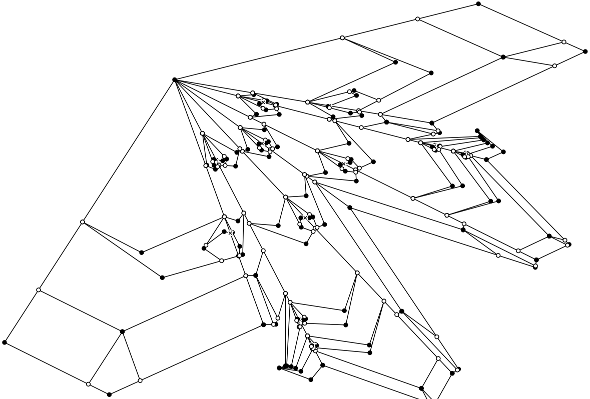
Result

- The resulting structure has complexity $O(n^2)$.



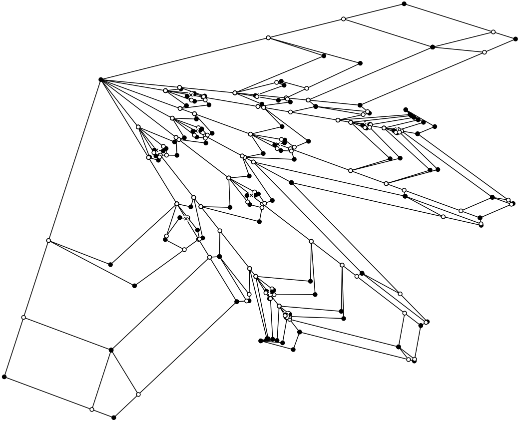
Result

- The resulting structure has complexity $O(n^2)$.



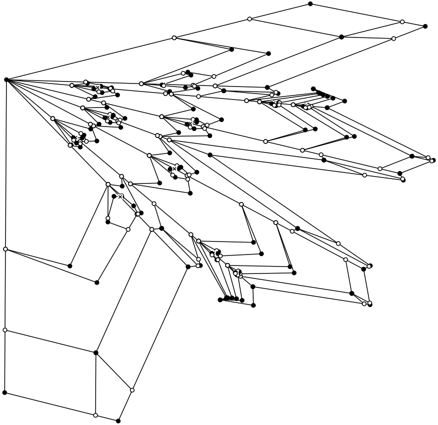
Result

- The resulting structure has complexity $O(n^2)$.



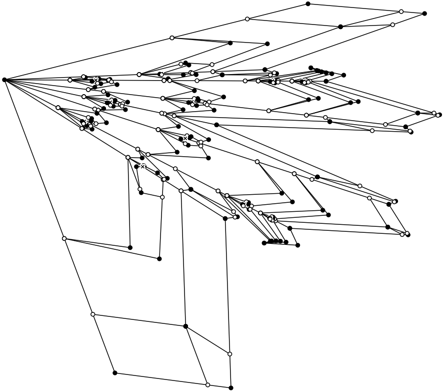
Result

- The resulting structure has complexity $O(n^2)$.



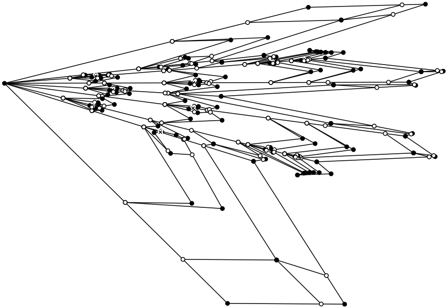
Result

- The resulting structure has complexity $O(n^2)$.



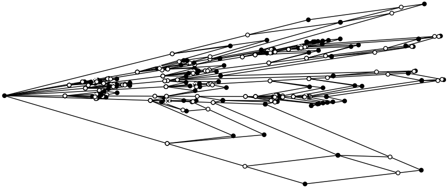
Result

- The resulting structure has complexity $O(n^2)$.



Result

- The resulting structure has complexity $O(n^2)$.



3D (orthogonal) model for pop-ups

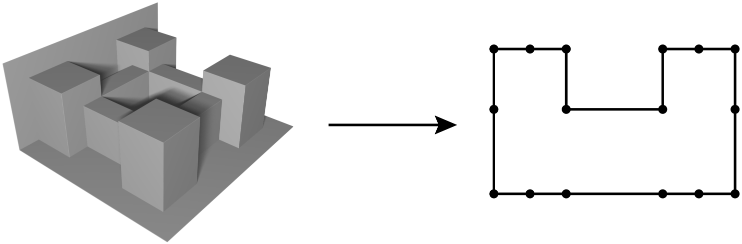
- **Input:** orthogonal polyhedron P, one distinguished edge.
- **Output:** set of *hinged rigid sheets of paper* that folds from P to a flat state with a 1-dof motion.

3D (orthogonal) model for pop-ups

- **Input:** orthogonal polyhedron P, one distinguished edge.
- **Output:** set of *hinged rigid sheets of paper* that folds from P to a flat state with a 1-dof motion.
- **Bellows theorem:** every flexible polyhedron has the same volume in all configurations.
  - We must cut the boundary.

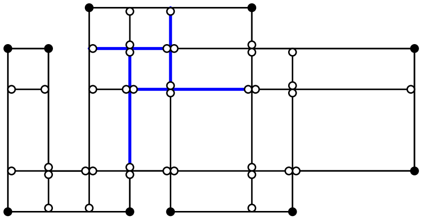
Cutting into slices

- Use the 3D grid induced by the vertices of P.
- Create slices perpendicular to the crease.
  - Each slice is a 2D linkage problem.



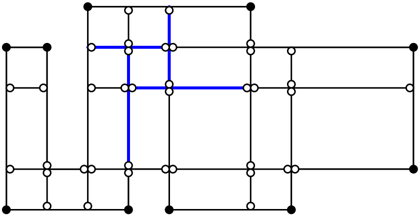
Pinwheel construction

- For each cross section, construct a *pinwheel-pattern* linkage, enforcing a 1-dof shearing motion.

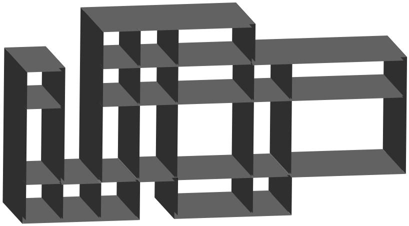


Pinwheel construction

- For each cross section, construct a *pinwheel-pattern* linkage, enforcing a 1-dof shearing motion.

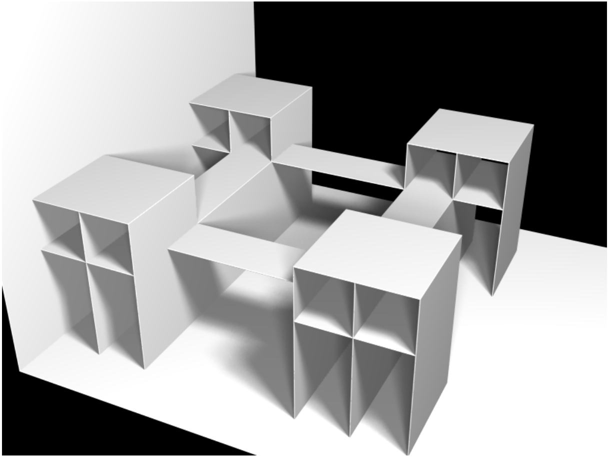


- Extrude each cross section to get a 3D model for a slice of P.



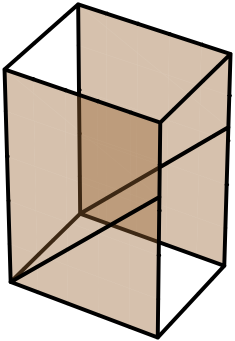
Putting slices together

- Fuse paper in adjacent slices.
  - But we still have holes on the sides...



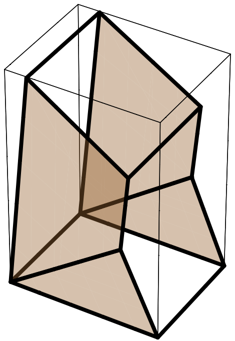
Closing holes

- Add two hinged sheets of paper to close each hole.
- Just the left and bottom sides are hinged to the rest of the structure.



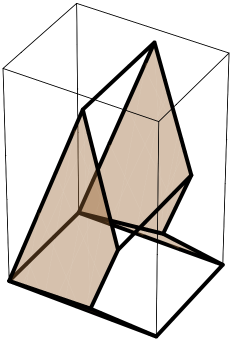
Closing holes

- Add two hinged sheets of paper to close each hole.
- Just the left and bottom sides are hinged to the rest of the structure.



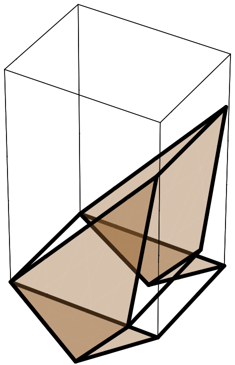
Closing holes

- Add two hinged sheets of paper to close each hole.
- Just the left and bottom sides are hinged to the rest of the structure.



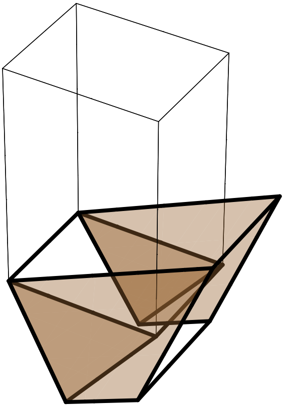
Closing holes

- Add two hinged sheets of paper to close each hole.
- Just the left and bottom sides are hinged to the rest of the structure.



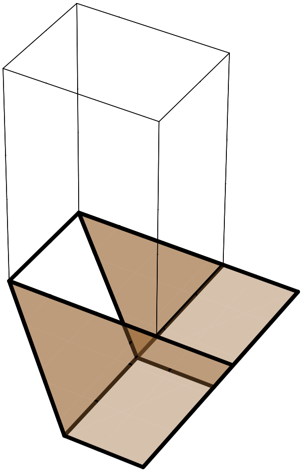
Closing holes

- Add two hinged sheets of paper to close each hole.
- Just the left and bottom sides are hinged to the rest of the structure.



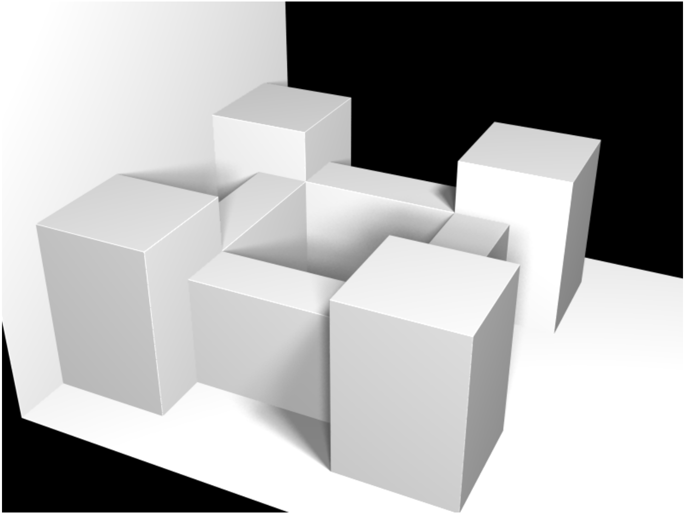
Closing holes

- Add two hinged sheets of paper to close each hole.
- Just the left and bottom sides are hinged to the rest of the structure.



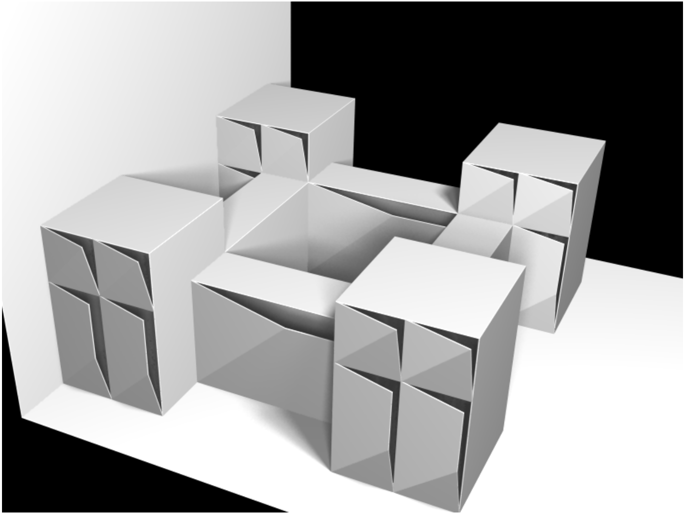
Result

- The resulting structure has complexity $O(n^3)$.



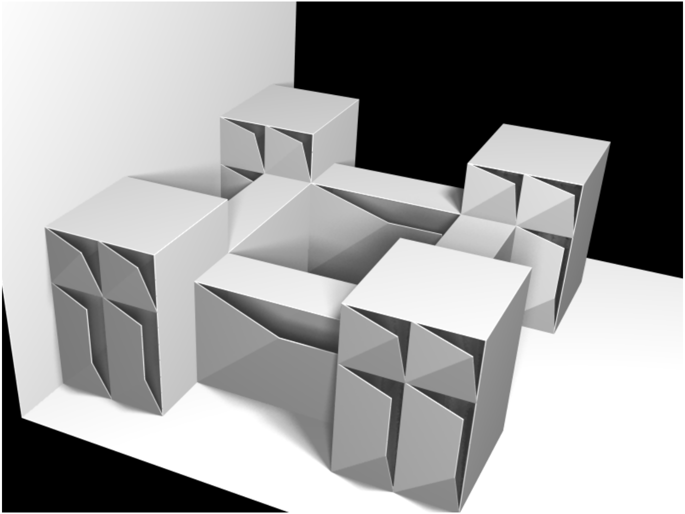
Result

- The resulting structure has complexity $O(n^3)$.



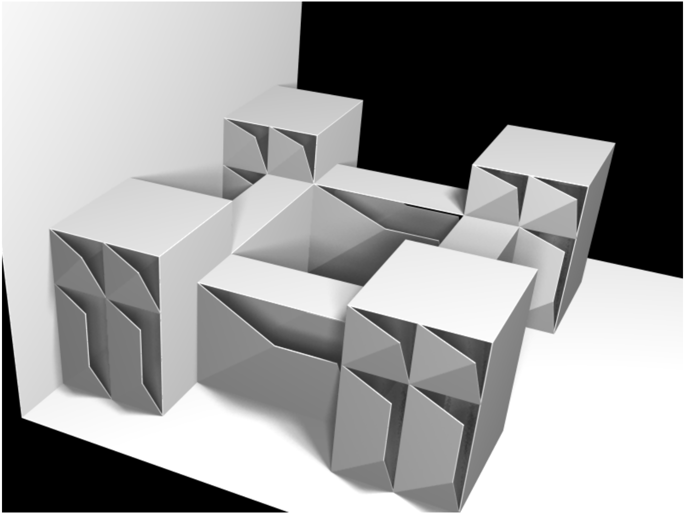
Result

- The resulting structure has complexity $O(n^3)$.



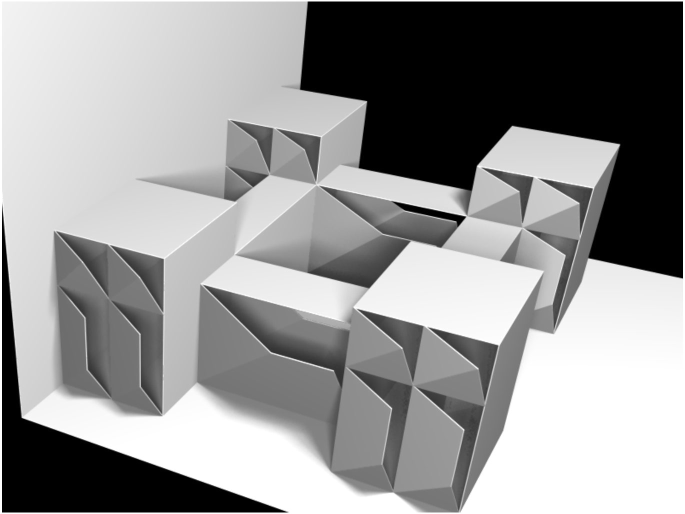
Result

- The resulting structure has complexity $O(n^3)$.



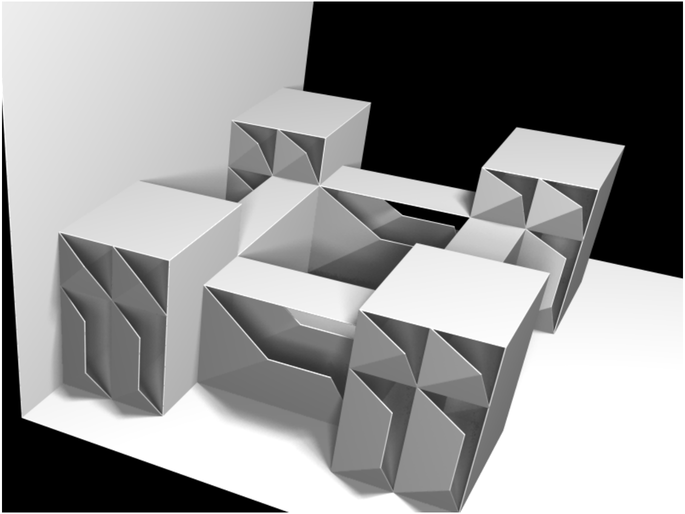
Result

- The resulting structure has complexity $O(n^3)$.



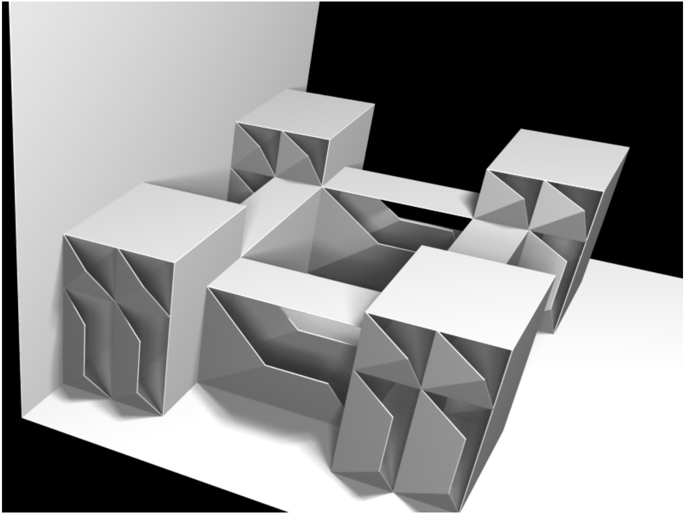
Result

- The resulting structure has complexity $O(n^3)$.



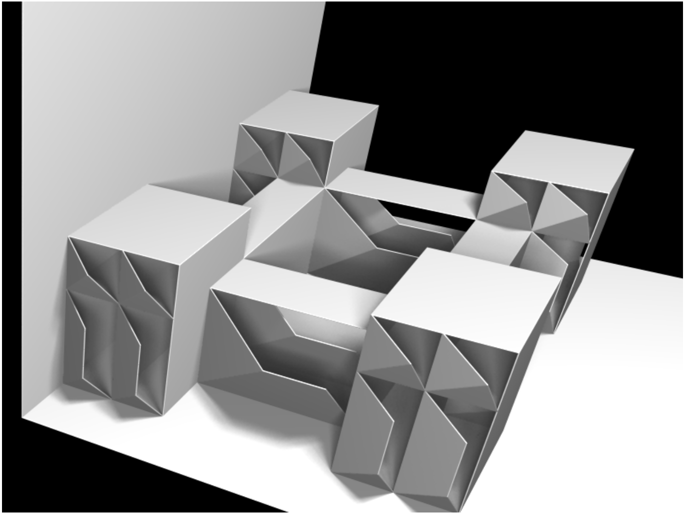
Result

- The resulting structure has complexity $O(n^3)$.



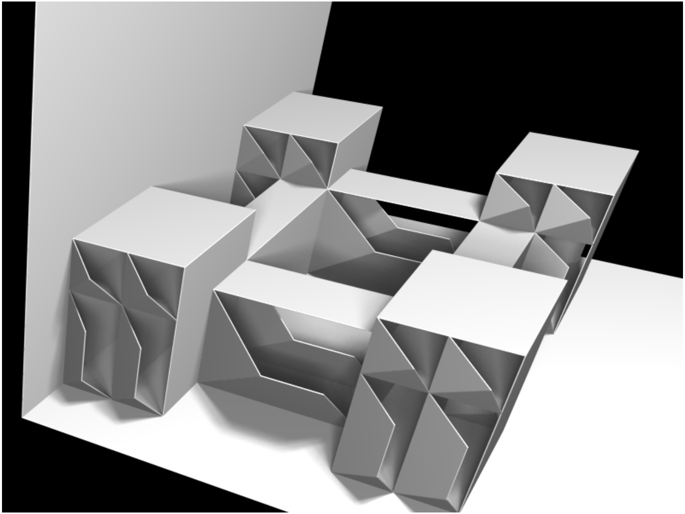
Result

- The resulting structure has complexity $O(n^3)$.



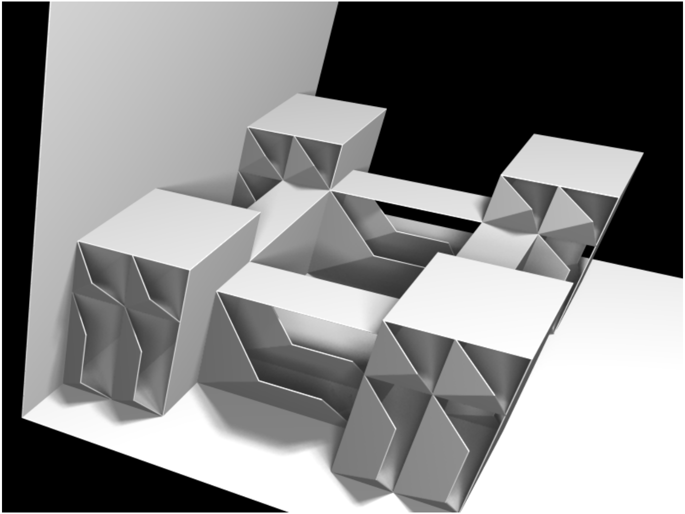
Result

- The resulting structure has complexity $O(n^3)$.



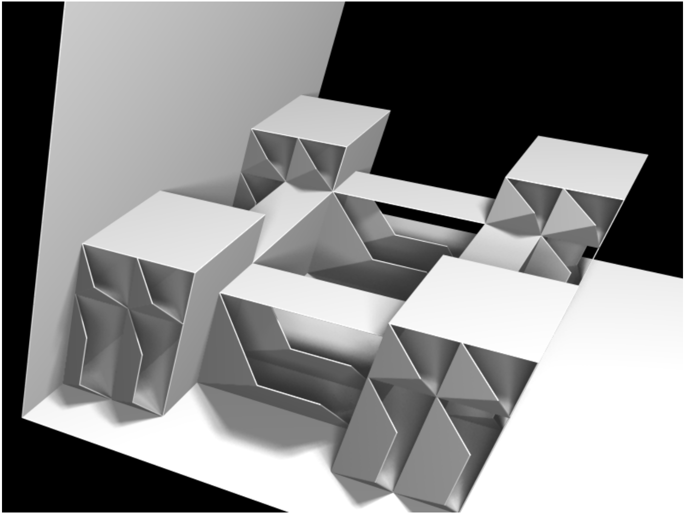
Result

- The resulting structure has complexity $O(n^3)$.



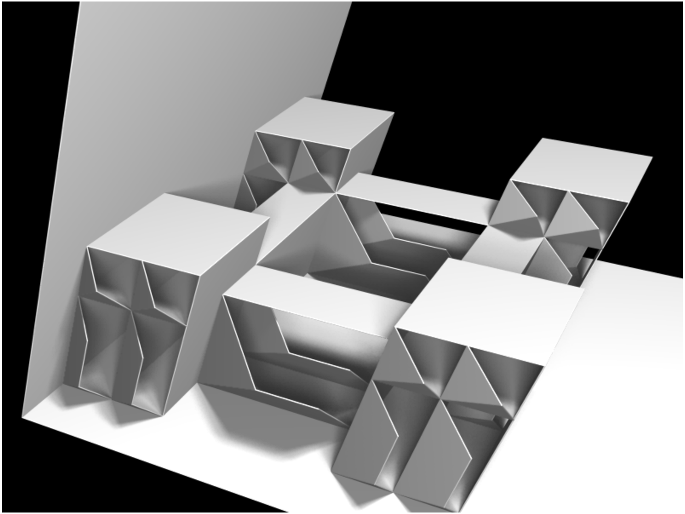
Result

- The resulting structure has complexity $O(n^3)$.



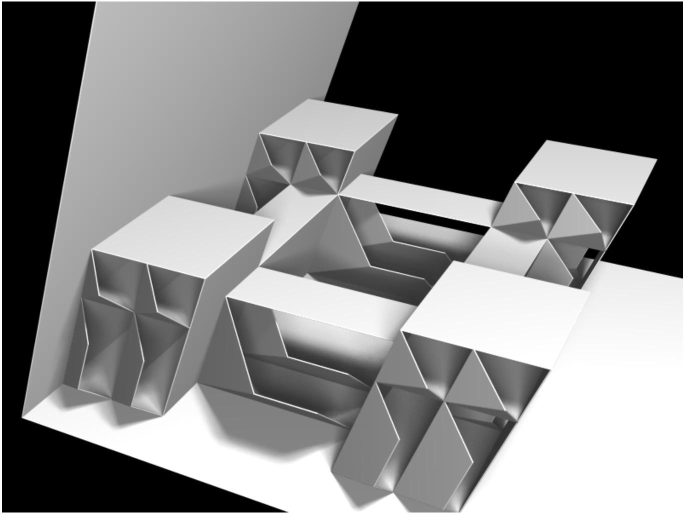
Result

- The resulting structure has complexity $O(n^3)$.



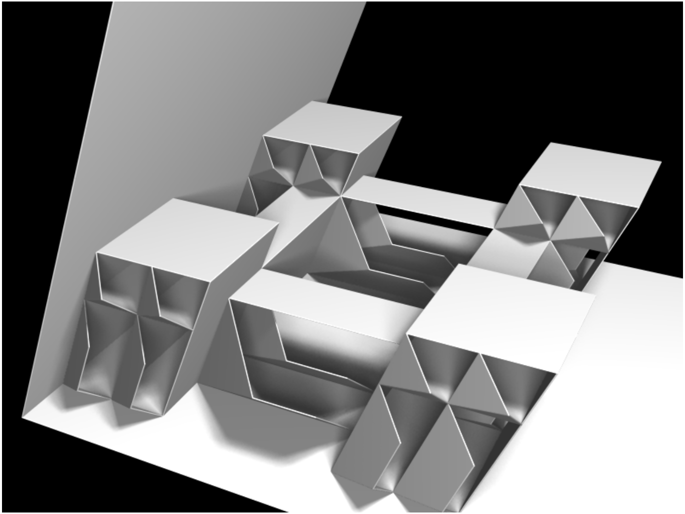
Result

- The resulting structure has complexity $O(n^3)$.



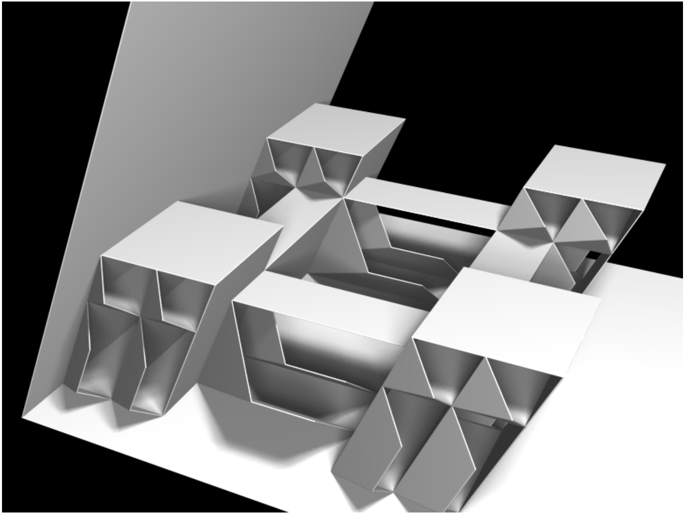
Result

- The resulting structure has complexity $O(n^3)$.



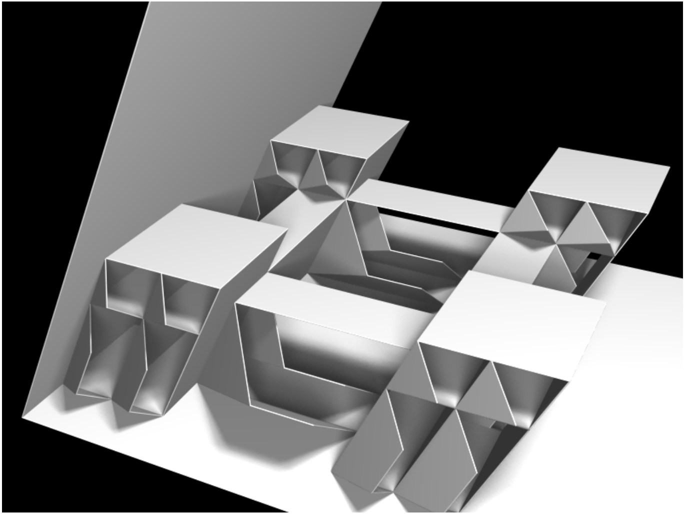
Result

- The resulting structure has complexity $O(n^3)$.



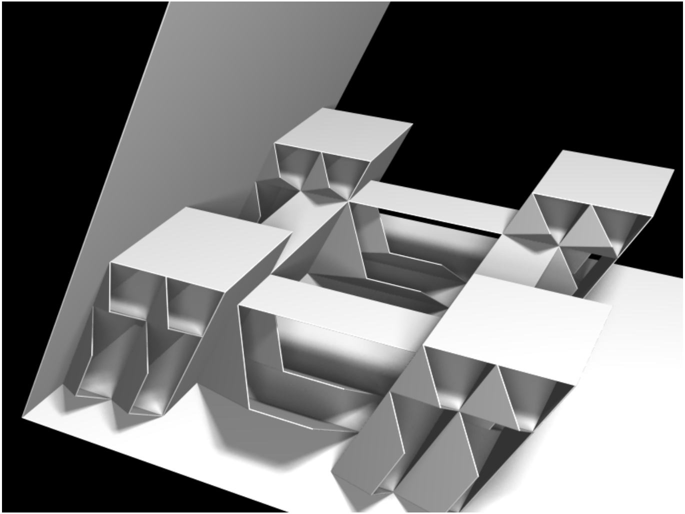
Result

- The resulting structure has complexity $O(n^3)$.



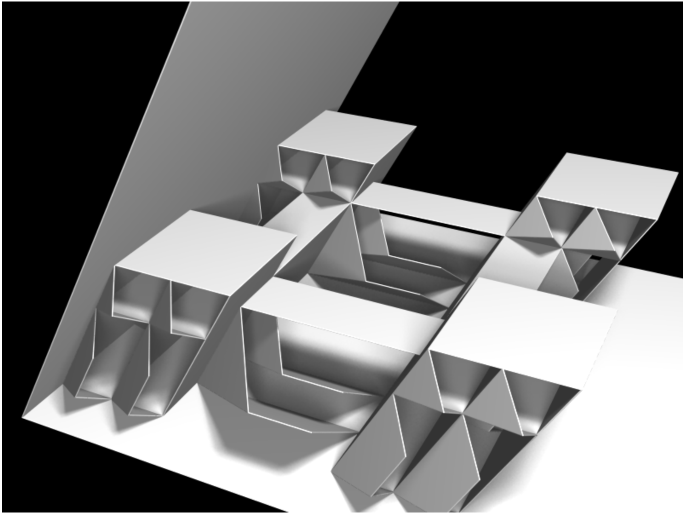
Result

- The resulting structure has complexity $O(n^3)$.



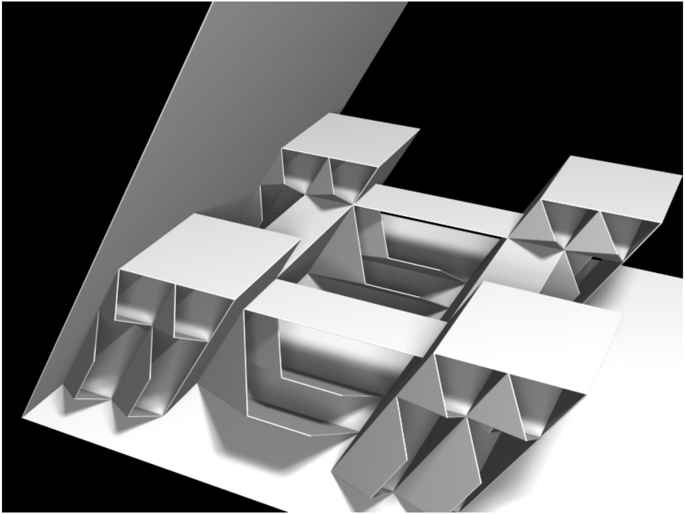
Result

- The resulting structure has complexity $O(n^3)$.



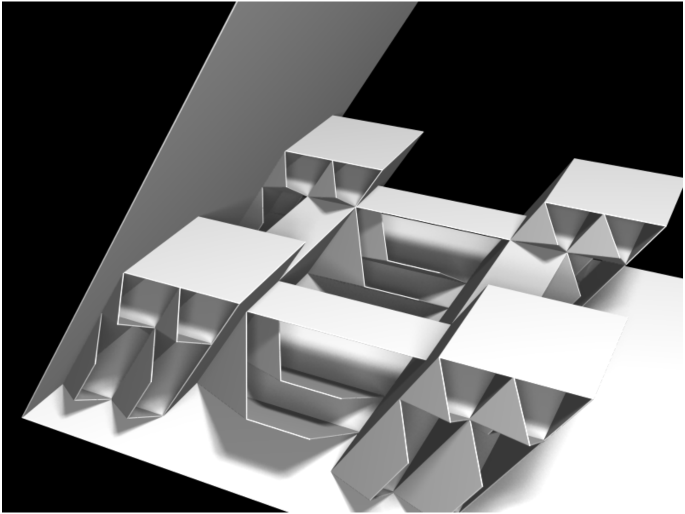
Result

- The resulting structure has complexity $O(n^3)$.



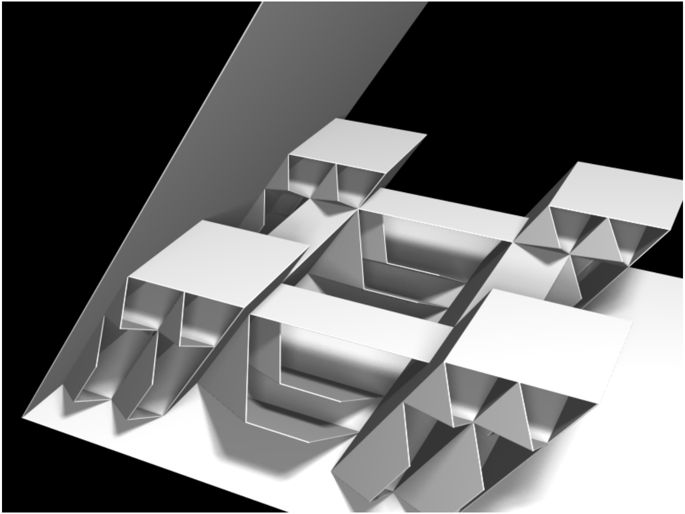
Result

- The resulting structure has complexity $O(n^3)$.



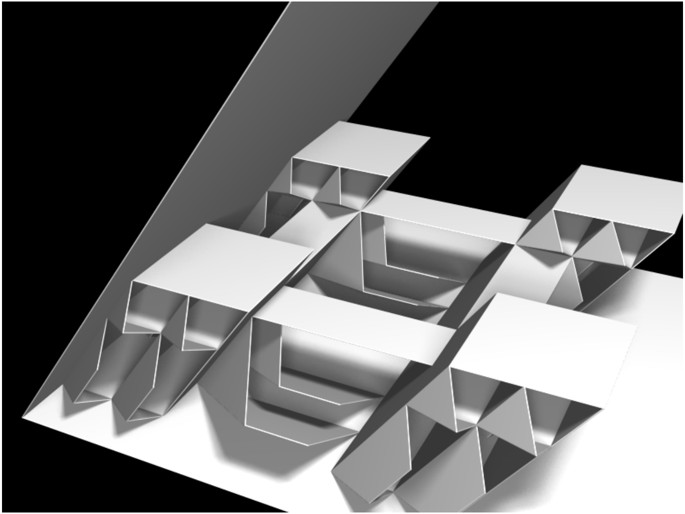
Result

- The resulting structure has complexity $O(n^3)$.



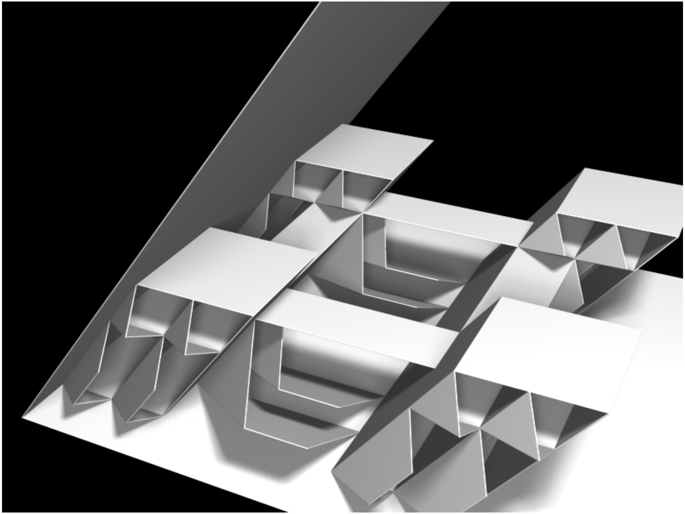
Result

- The resulting structure has complexity $O(n^3)$.



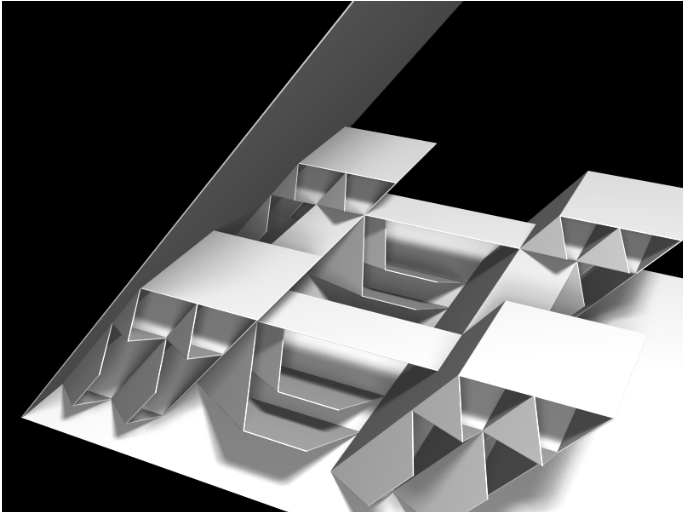
Result

- The resulting structure has complexity $O(n^3)$.



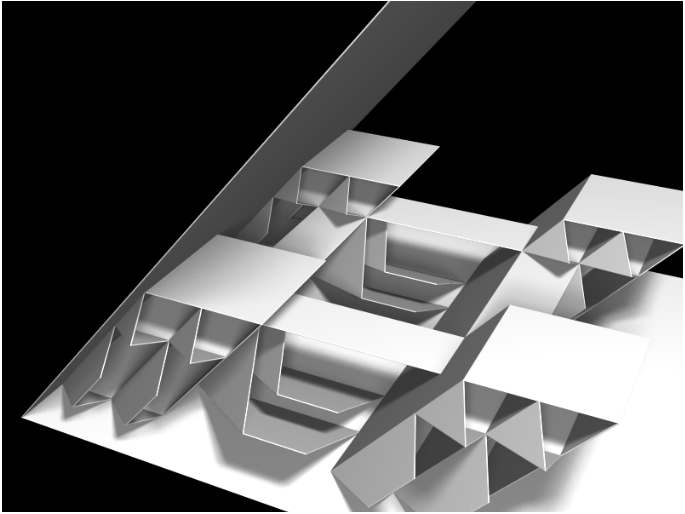
Result

- The resulting structure has complexity $O(n^3)$.



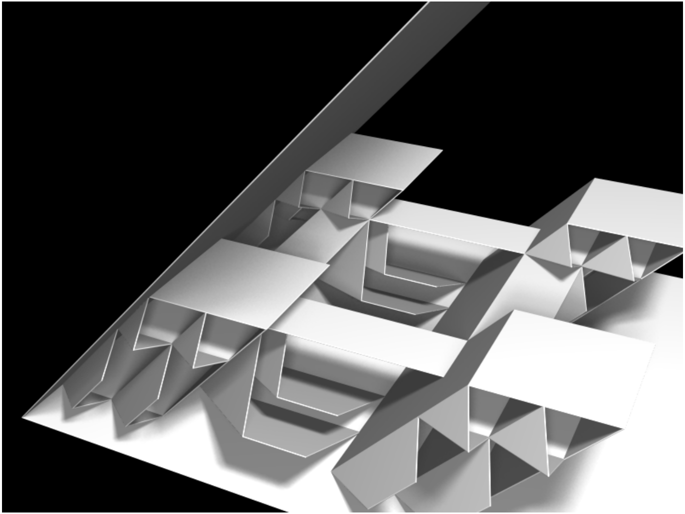
Result

- The resulting structure has complexity $O(n^3)$.



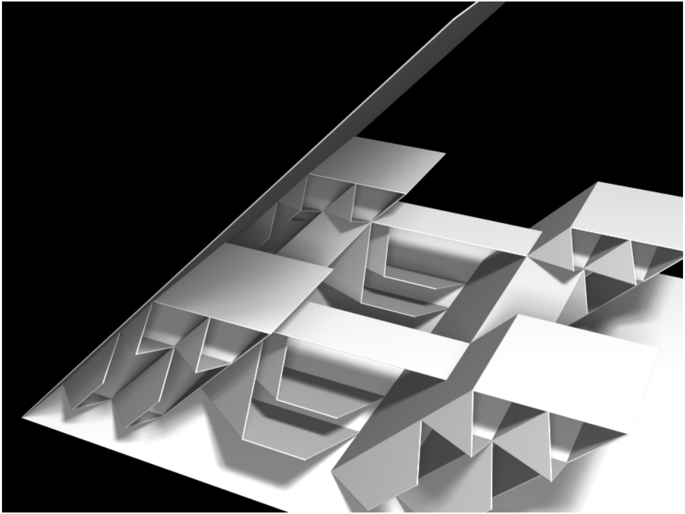
Result

- The resulting structure has complexity $O(n^3)$.



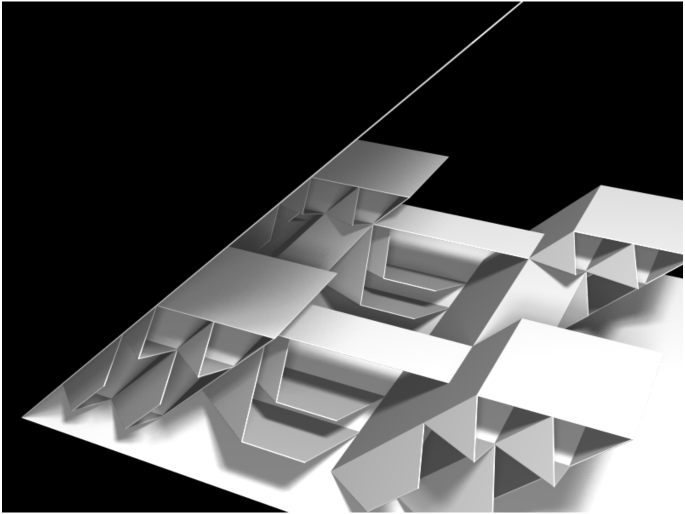
Result

- The resulting structure has complexity $O(n^3)$.



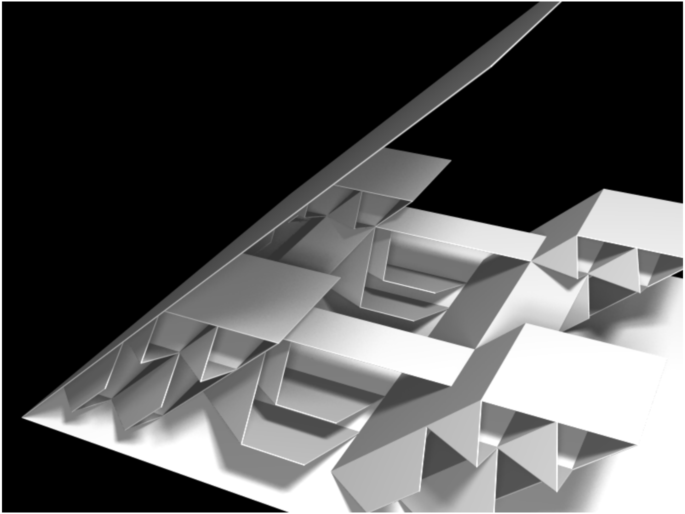
Result

- The resulting structure has complexity $O(n^3)$.



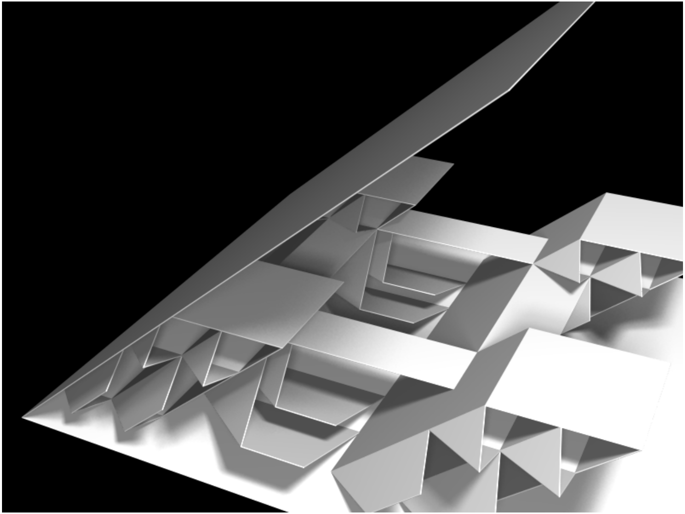
Result

- The resulting structure has complexity $O(n^3)$.



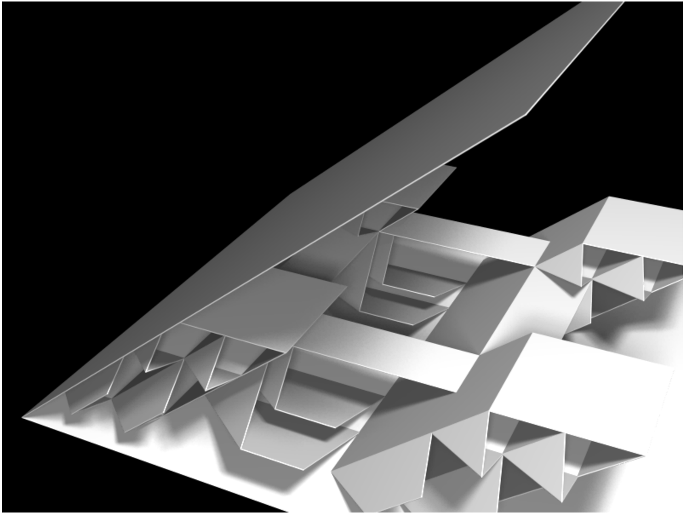
Result

- The resulting structure has complexity $O(n^3)$.



Result

- The resulting structure has complexity $O(n^3)$.



Summary

- $O(n)$ solution for orthogonal polygons.
- $O(n^2)$ solution for general polygons.
- $O(n^3)$ solution for orthogonal polyhedra.

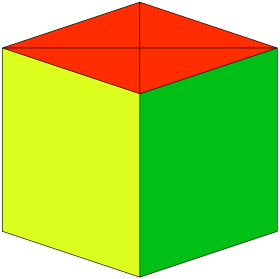
- **Open:** Can every polyhedron be a pop-up?

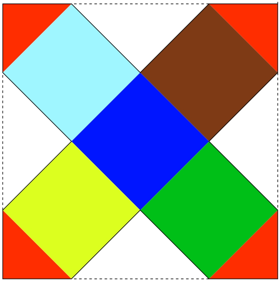
Wrapping a Cube with Rectangular Paper

Work in progress...

Joint work with E. Bardelli and M. Mamino

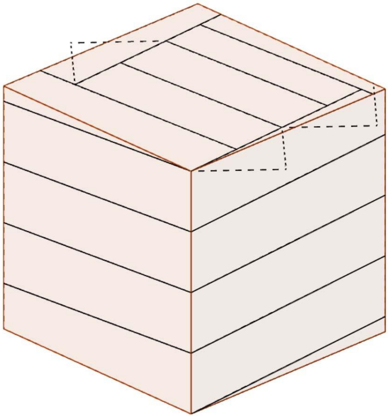
Wrapping a cube with a square





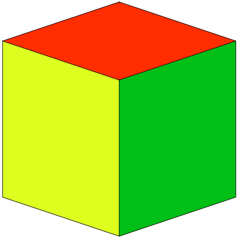
The optimal solution wastes $1/4$ of the paper (Beebee et al., 2001) and is unique (Pan, 2014).

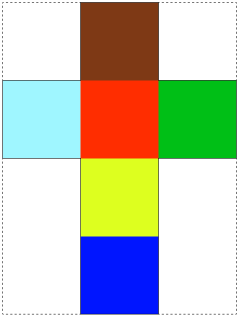
Wrapping a cube with a rectangle



With a long-enough strip, we can be as efficient as we want

(Cole et al., 2013).



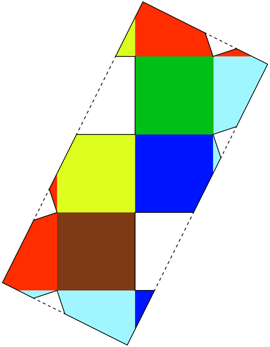


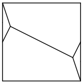
What if we want to avoid overlaps in the wrapping paper?

This corresponds to unfolding a cube into a rectangular region.

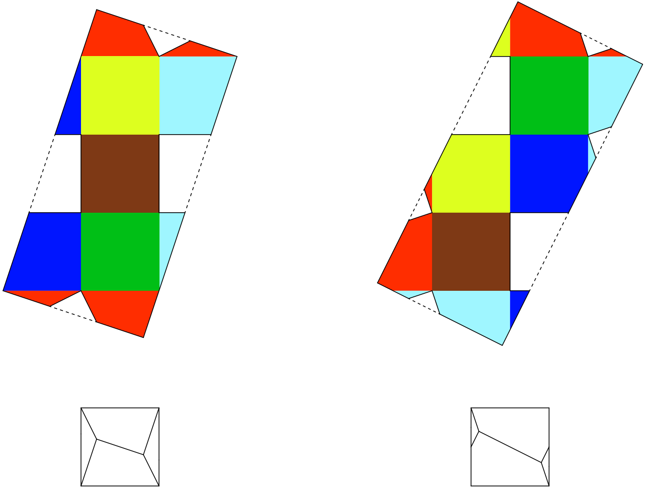
How small can this region be?

Unfolding a cube into a rectangle



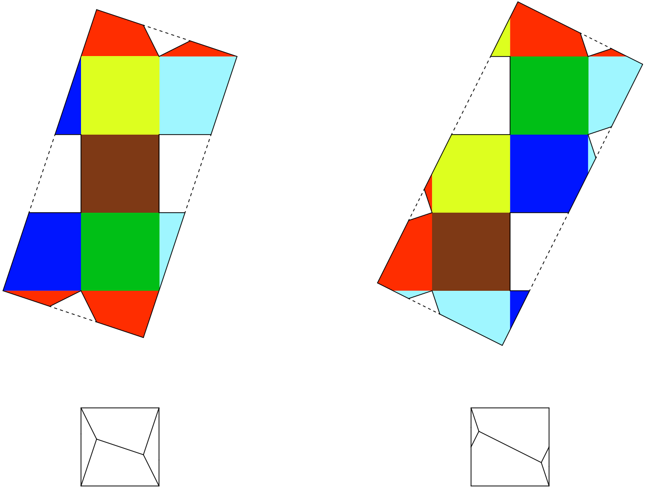


Unfolding a cube into a rectangle



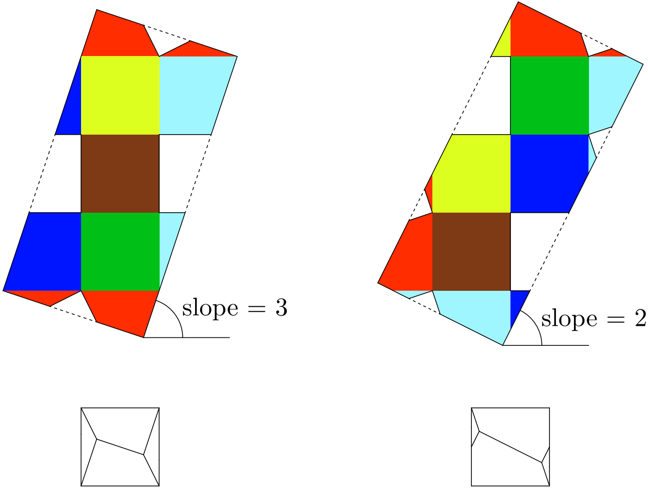
There are two particularly efficient unfoldings.

Unfolding a cube into a rectangle



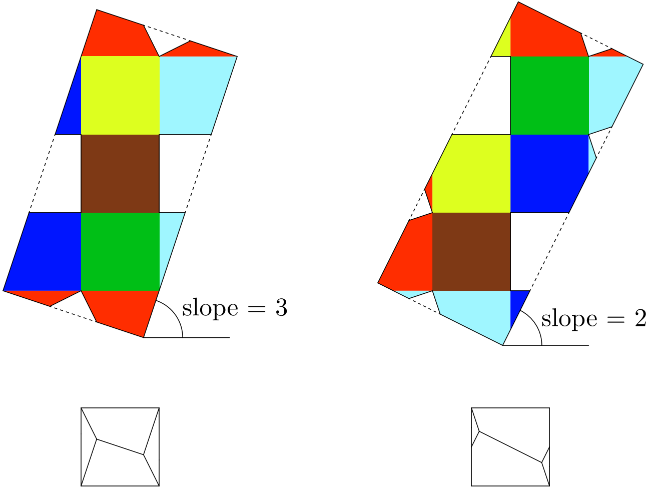
They both waste only 1/6 of the paper.

Unfolding a cube into a rectangle



No other unfoldings that waste $\leq 1/6$ of the paper are known.

Unfolding a cube into a rectangle



Is 1/6 optimal? Are there any other optimal unfoldings?