Computational Design + Fabrication: 1D Geometry

Jonathan Bachrach

EECS UC Berkeley

September 17, 2015

Today

- news
- section review
- bottom-up 1d->3d – pre-rationalized
  - stick figures
- top-down 3d->1d – post-rationalized
  - meshes – flattening pieces
  - slicing – interlocking
  - piercing – weaving
  - space filling curves
- joinery + assembly
- review lab 0

News

- invention lab training
- wire bending training
- section next week in soda 373 2-3p
- teams of one ok for labs

- transformations
- generator -> get_generator
- pictures of layers and flows

Transformations

- solid transformations are based on openscad
- polyline transformations are a bit broken because of overloading collision

```
pl *= transformation_matrix
```

- can move between digifab and solidpython

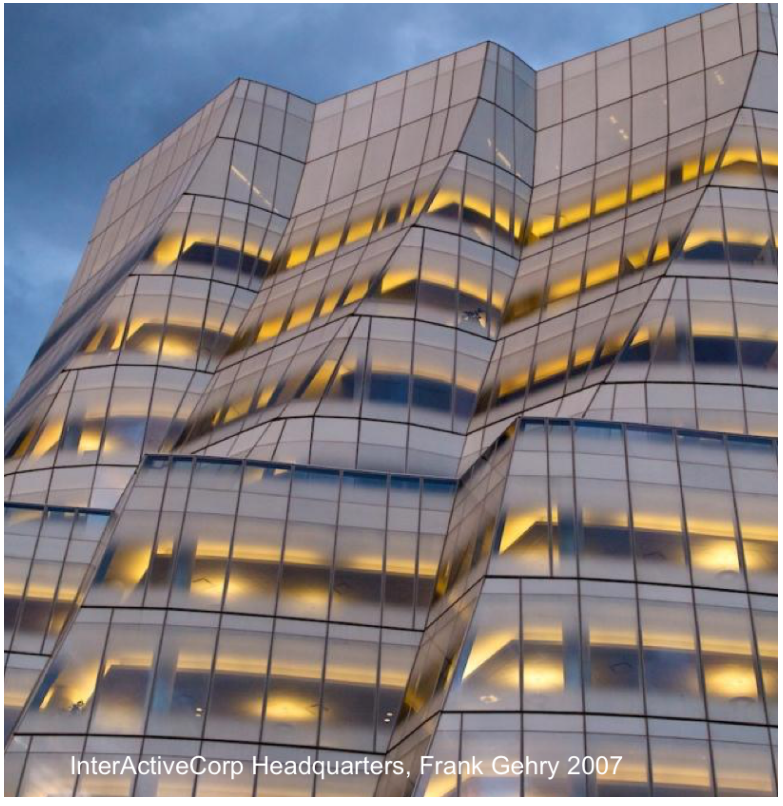
```
ss  = solid.square(10)
ps  = Polyline(generator=ss)
sps = ps.get_generator()
```

Layers of API

- digifab produces solidpython shapes
- solidpython produces openscad expressions

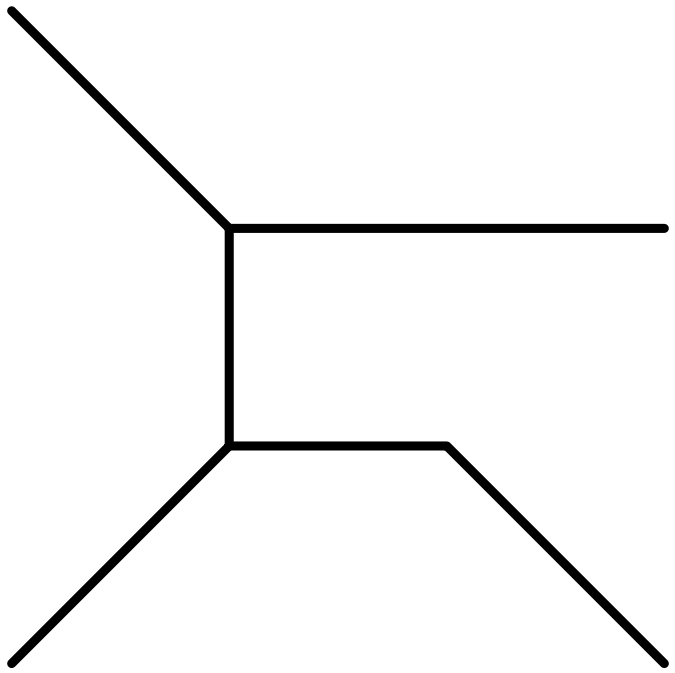
Pre and Post Rationalization

pre rationalization
computationalist first
foster

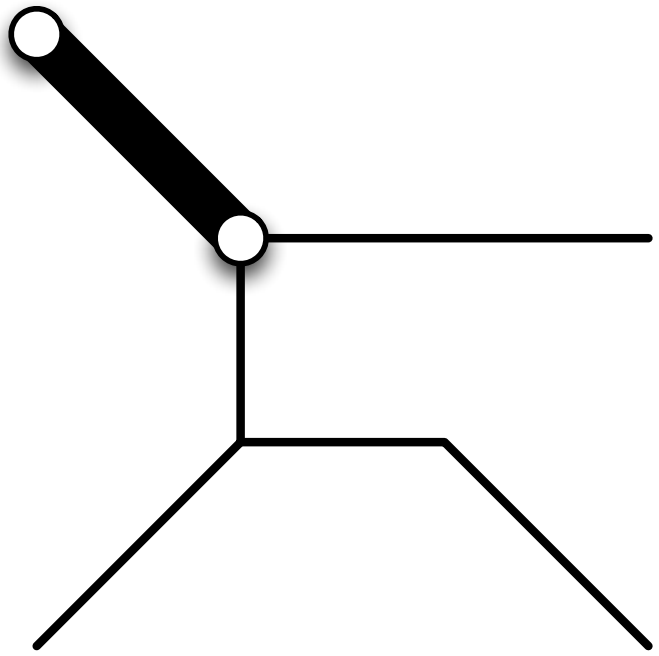


post rationalization
designer first
gehry

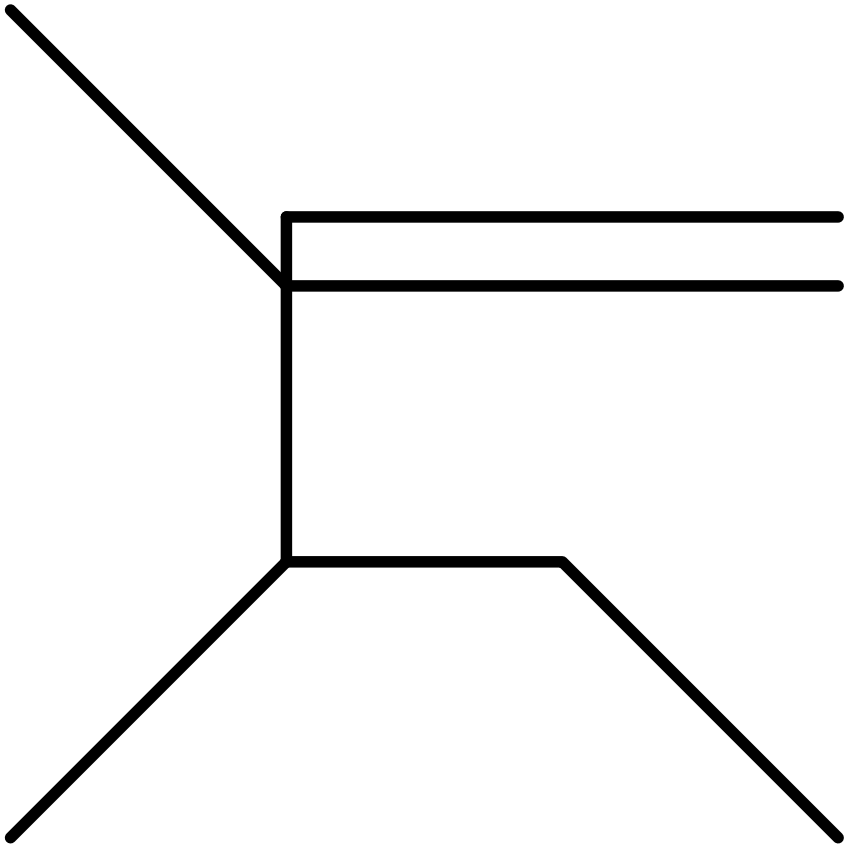
- stick figure

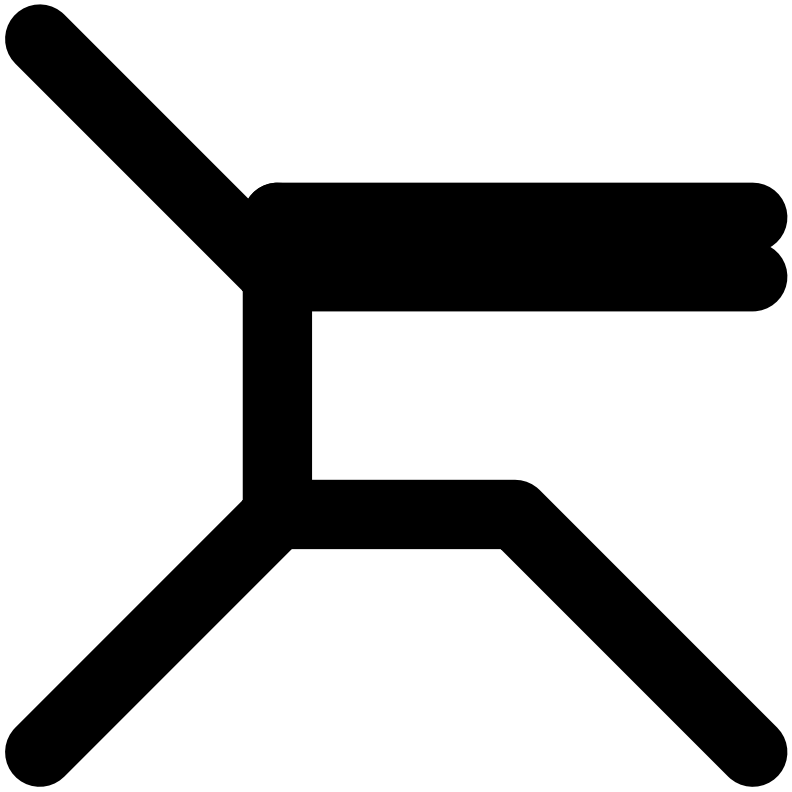


- point to sphere
- line to cylinder
- convex hull



- does thickening invalidate structure
- self intersection



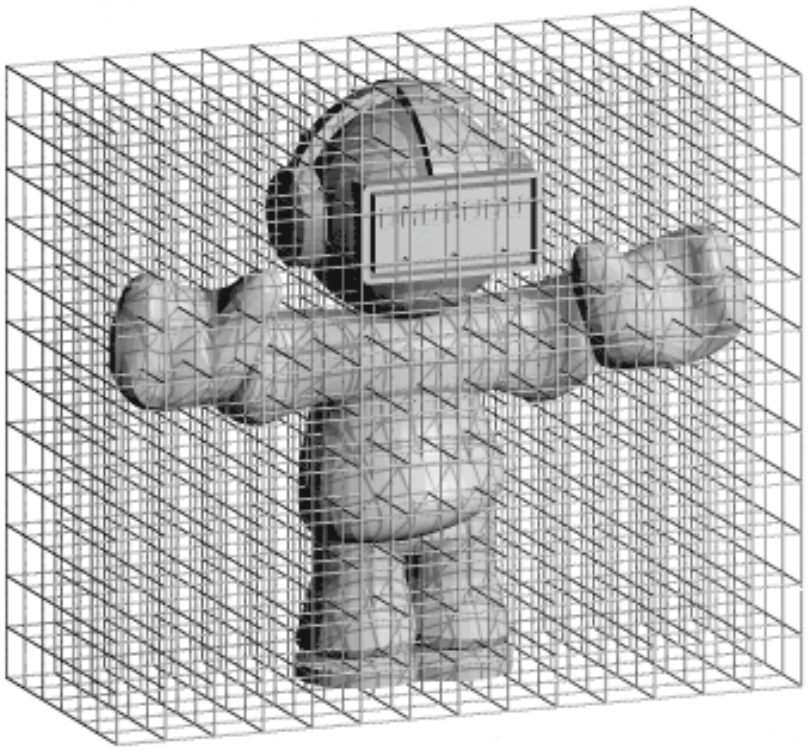


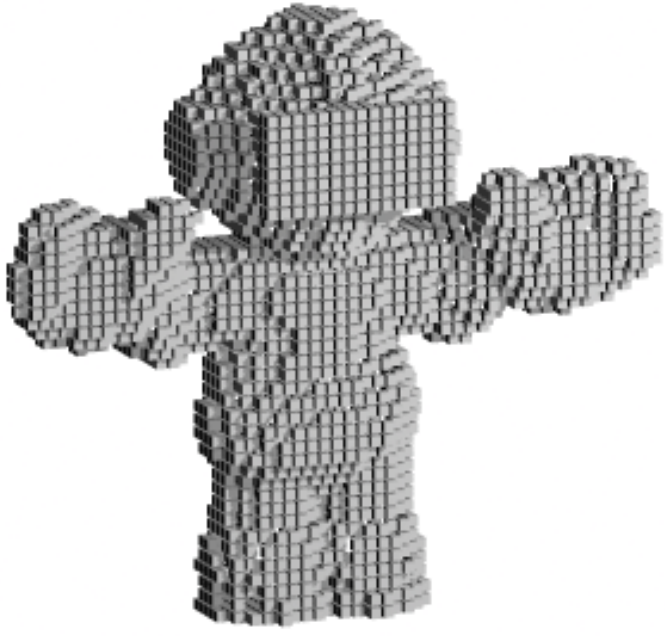
Post Rationalization

- decouple design from fabrication
- rationalize into parts after design
- potentially have to reject design

- 3D to 2D to 1D
- mesh -> polygons
- slice -> polygons
- lines -> segments
- space filling curves

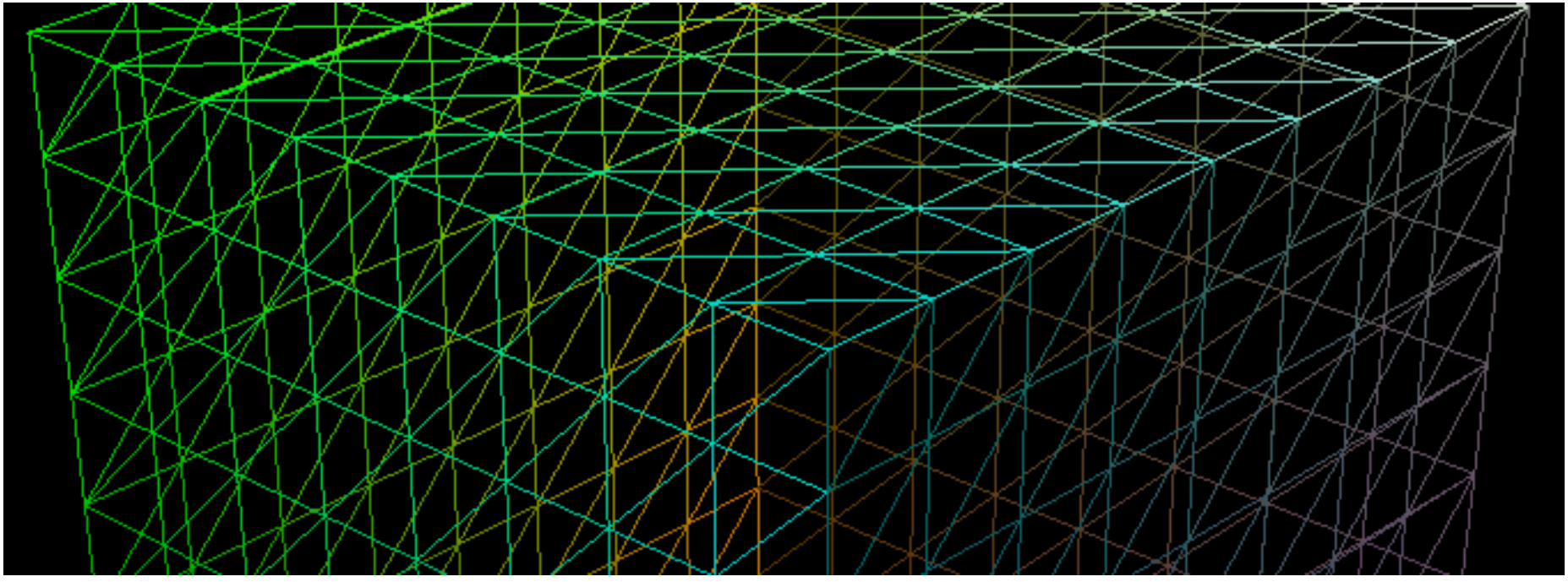
- overlay grid on solid
- represent grid cells called voxels





Space Frames Construction

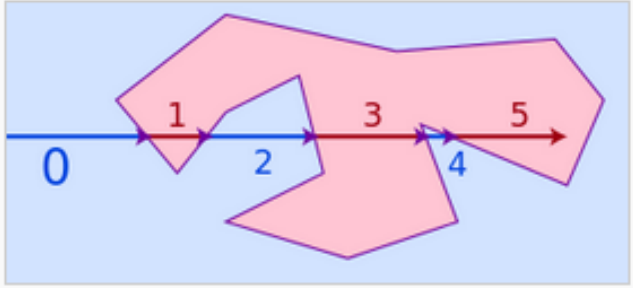
- voxelize
- build out borders of voxels
- adjust for space frame vertices



by giawa

- inside outside check
- line triangle intersections

```
int count = 0
for each tri:Triangle in mesh
  count += isIntersect(tri, ray)
val isInside = (count % 2) == 1
```



The number of intersections for a ray passing from the exterior of the polygon to any point; if odd, it shows that the point lies inside the polygon. If it is even, the point lies outside the polygon; this test also works in three dimensions.

Mesh To Polygons

- split mesh into faces
- print out face polygons
- connect

Rotate Faces to Plane

- mesh consists of faces with points in clockwise order
- know normal of triangle using right hand rule and cross product
- find rotation to place triangle in z=0 plane

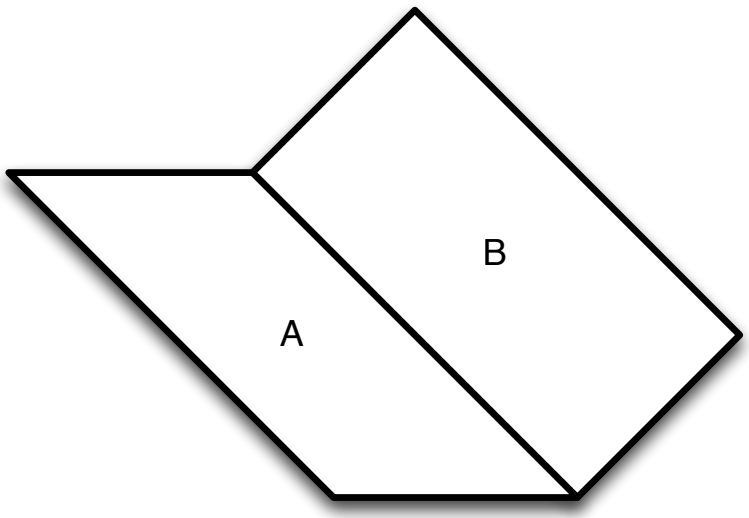
Dihedral Angle

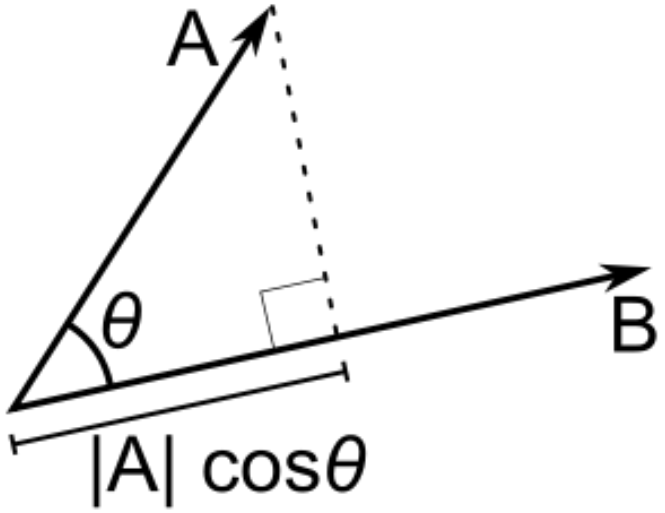
angle

- between two planes A and B looking down intersecting line
- by which plane A must be rotated to align it with plane B
- between their two normal vectors

$$N_A \cdot N_B = |N_A||N_B| \cos \theta_{AB} \tag{1}$$

$$\theta_{AB} = \arccos\left(\frac{N_A \cdot N_B}{|N_A||N_B|}\right) \tag{2}$$



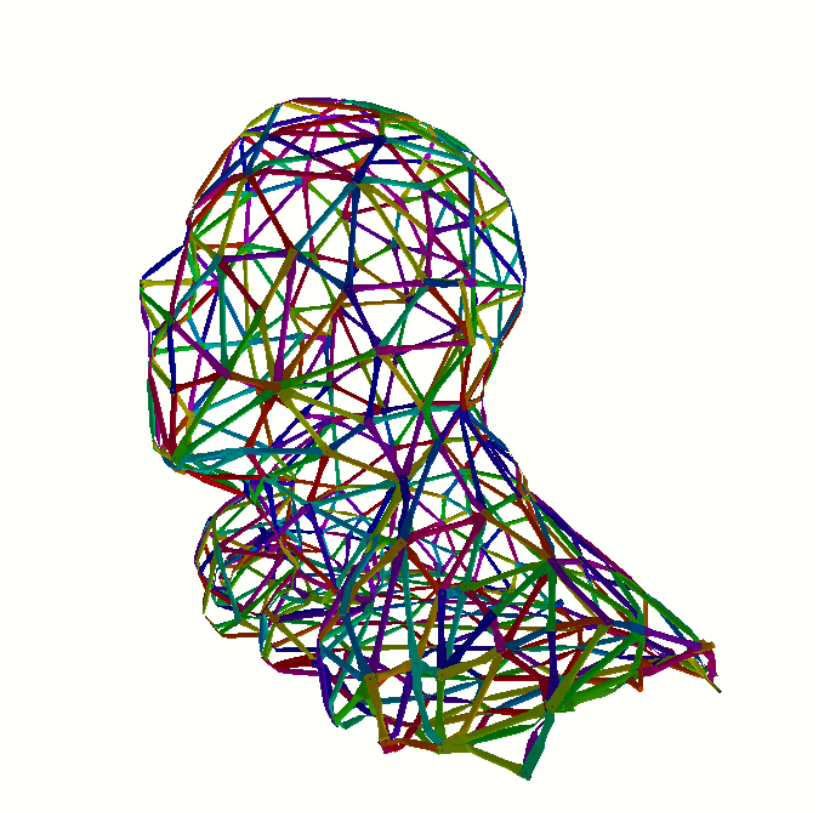


Mesh To Polygons Optimizations

- simplify mesh
- reduce shared edges if coplanar
- more to come ...

Print Out Mesh Edges

- split mesh into edges
- print out edges
- produce joinery to join edges at vertices

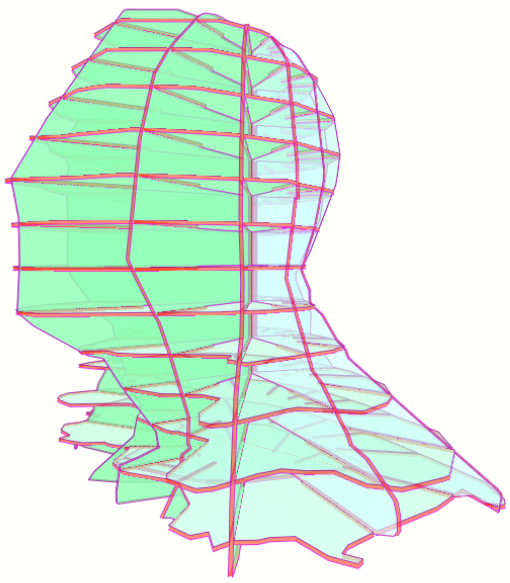


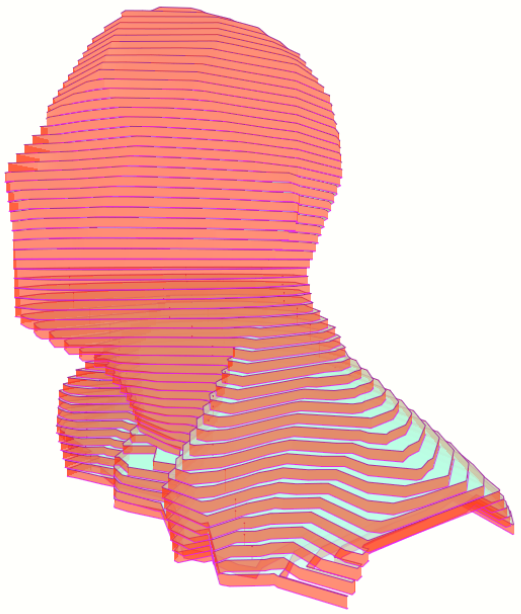
Elephant Play Structure

Elephant Play Structure

Shape To Slices

- slice one way
- slice radially
- slice orthogonal

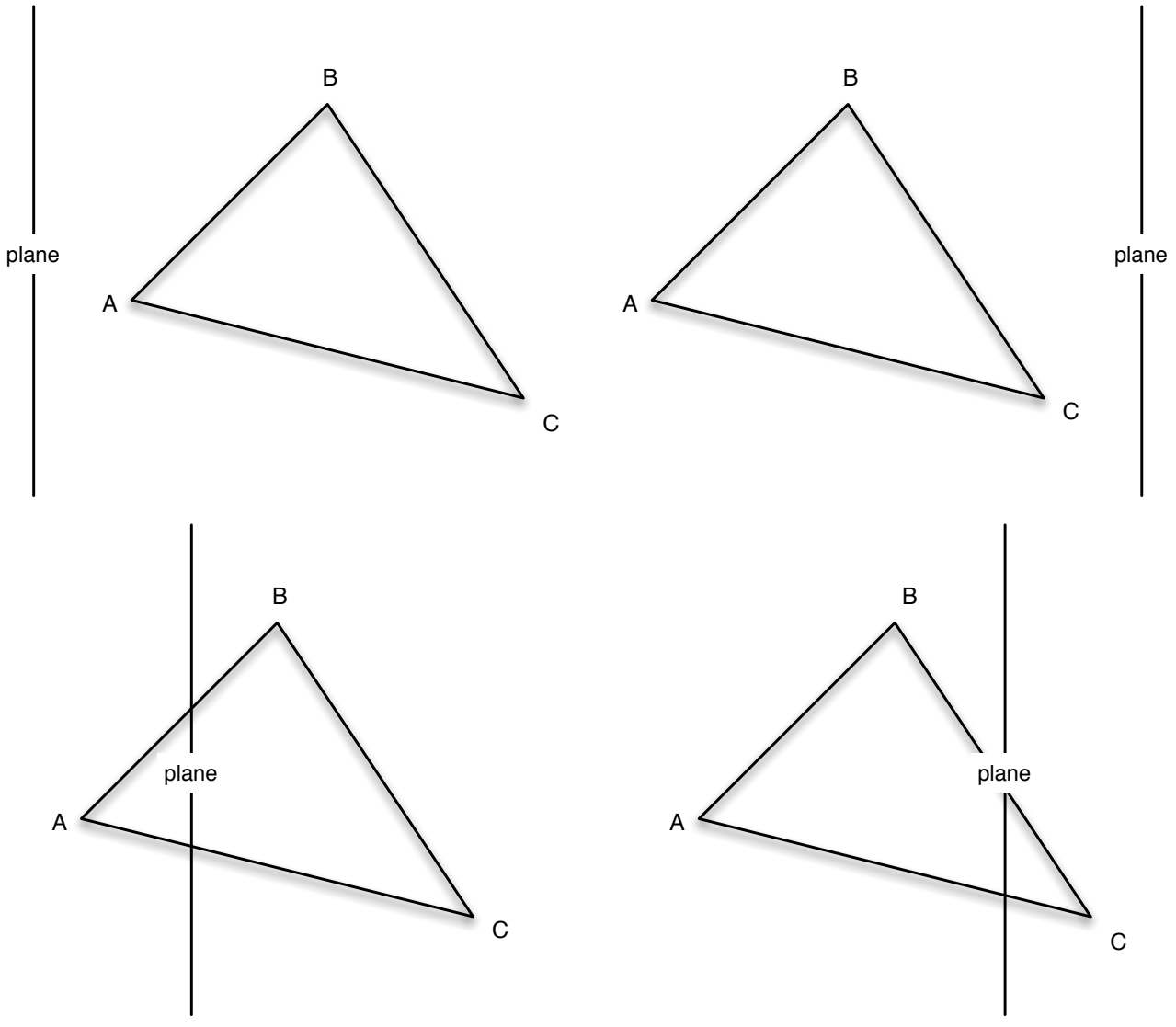




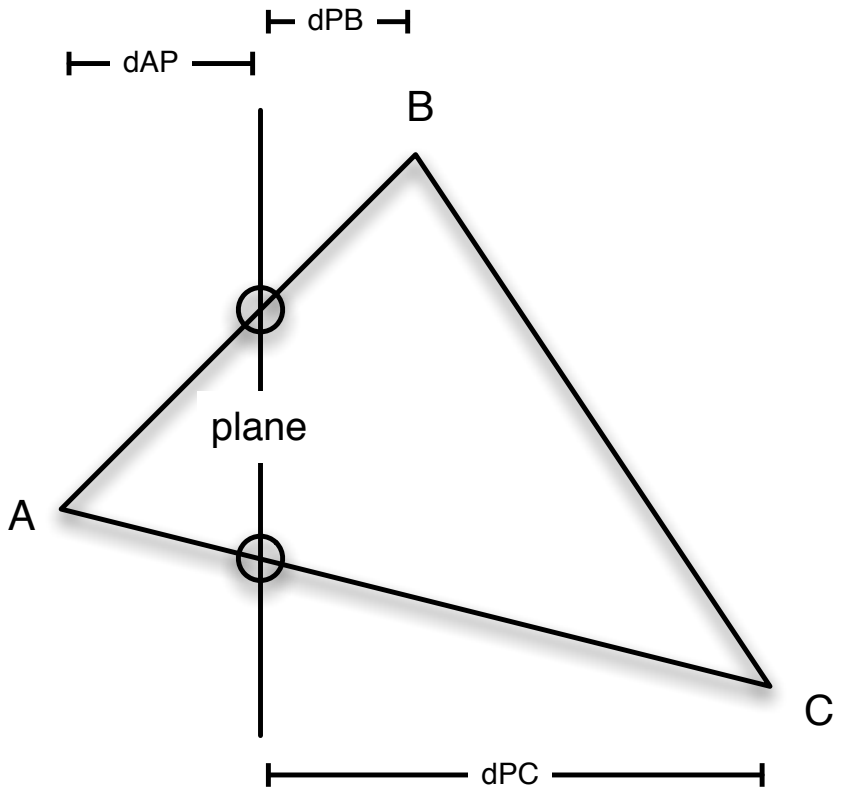
Slicing


```
val polys = Vector<PolyLine>()
for each slice:Plane in slices
  val poly:PolyLine
  for each tri:Triangle in mesh
    poly += intersect(tri, slice)
  polys += poly
```

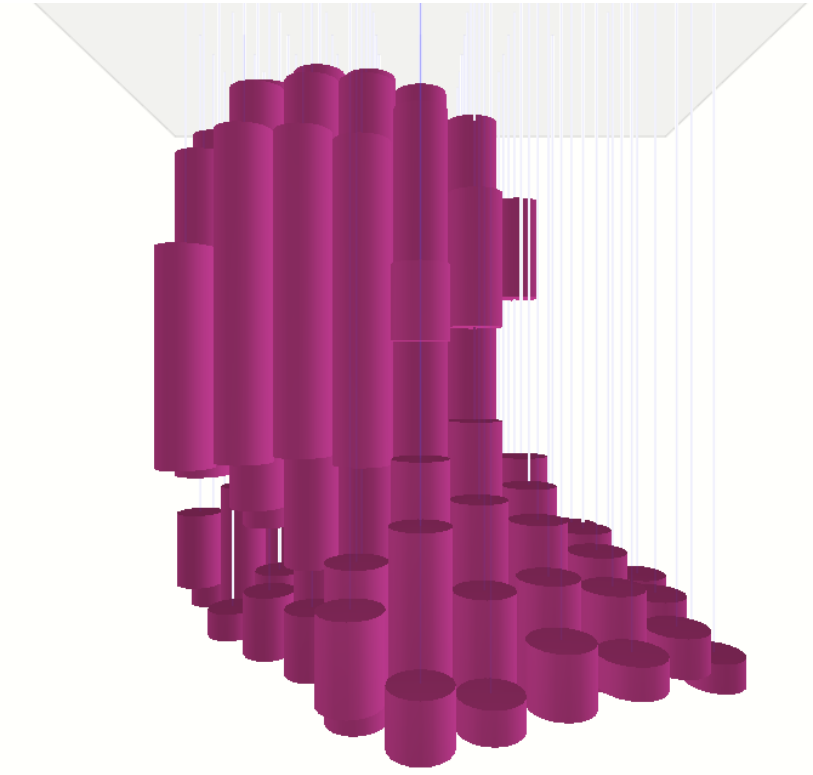
Plane / Triangle Intersection Cases



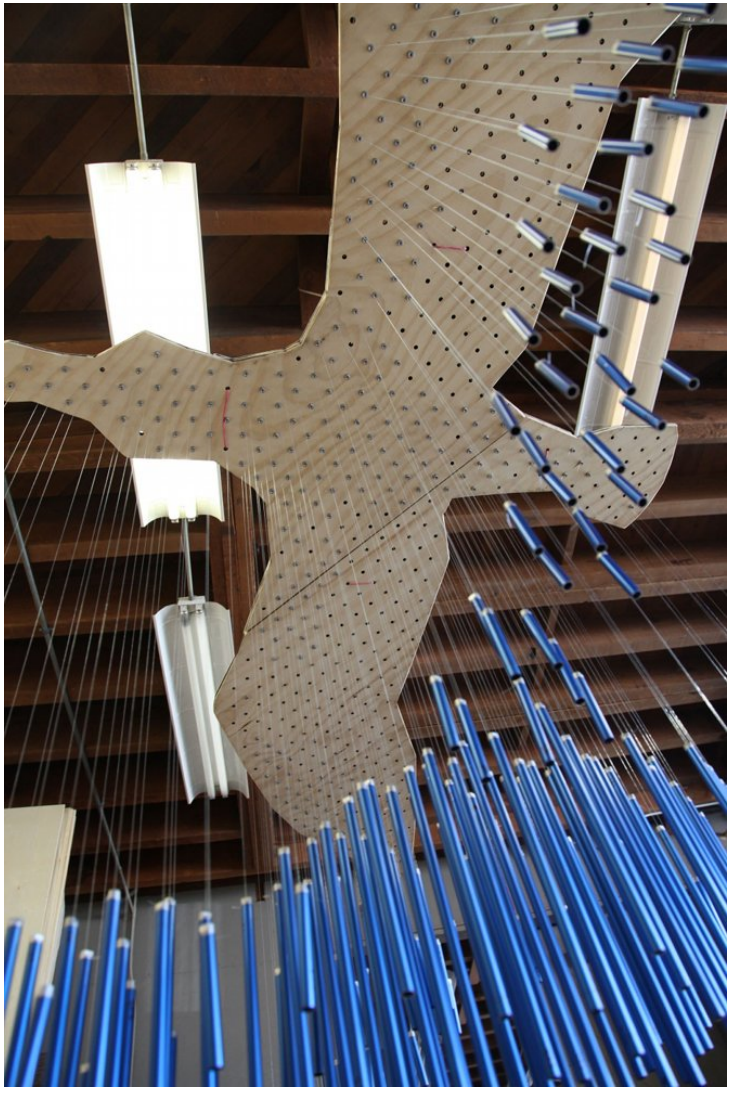
- use $d0 * d1 < 0$ to determine sign and insideness
- interpolate to find point between p0 and p1



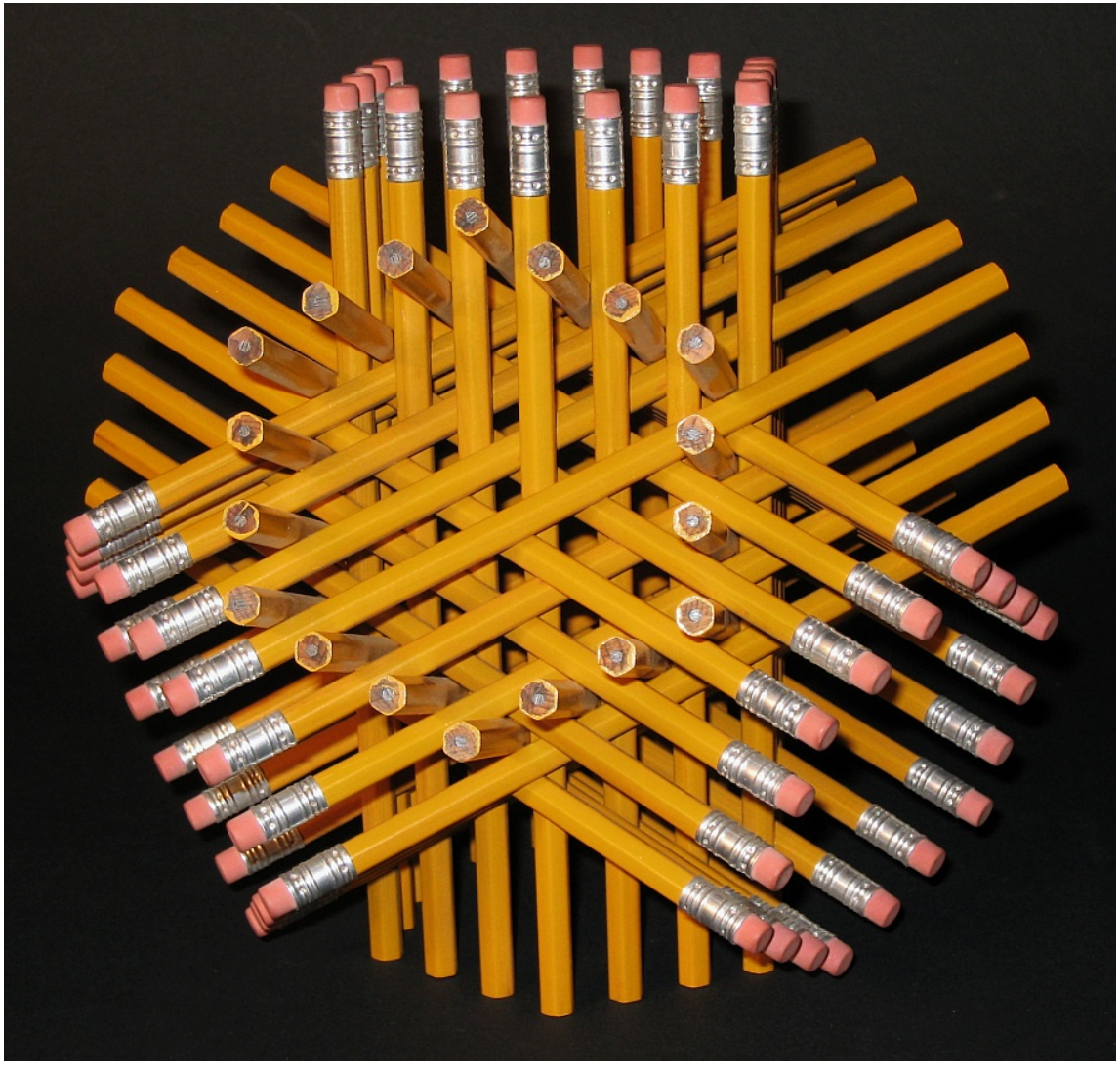
- find line intersections with mesh triangles and chop into segments
- perhaps do this from multiple angles and weave



Hanging Sculpture

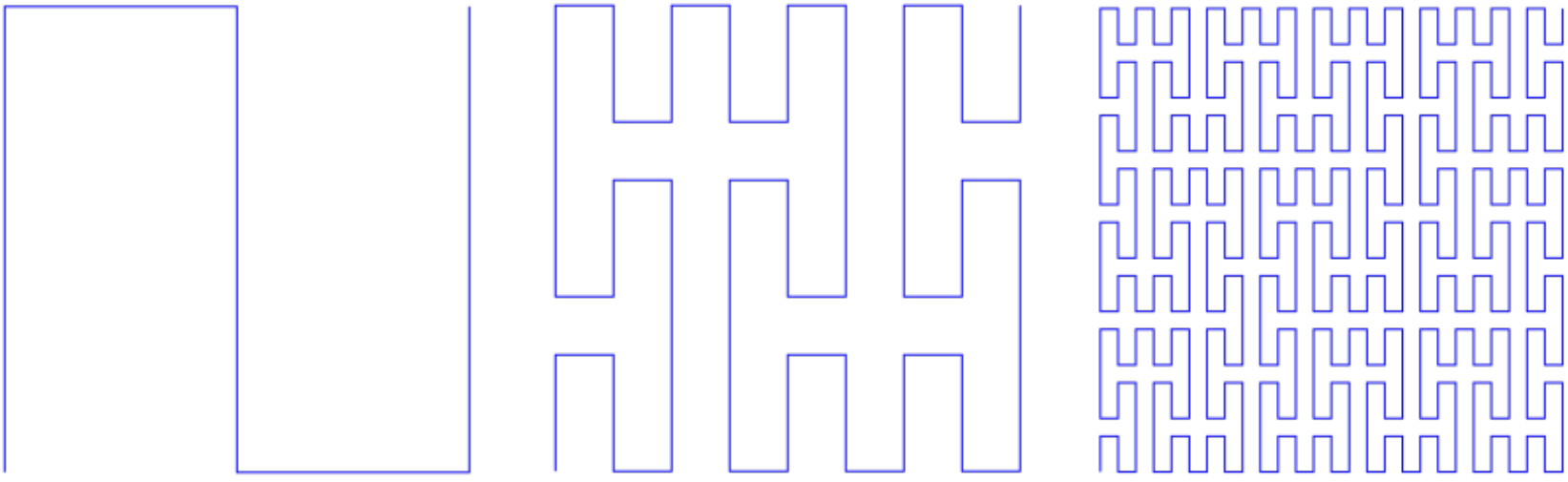


Pencils

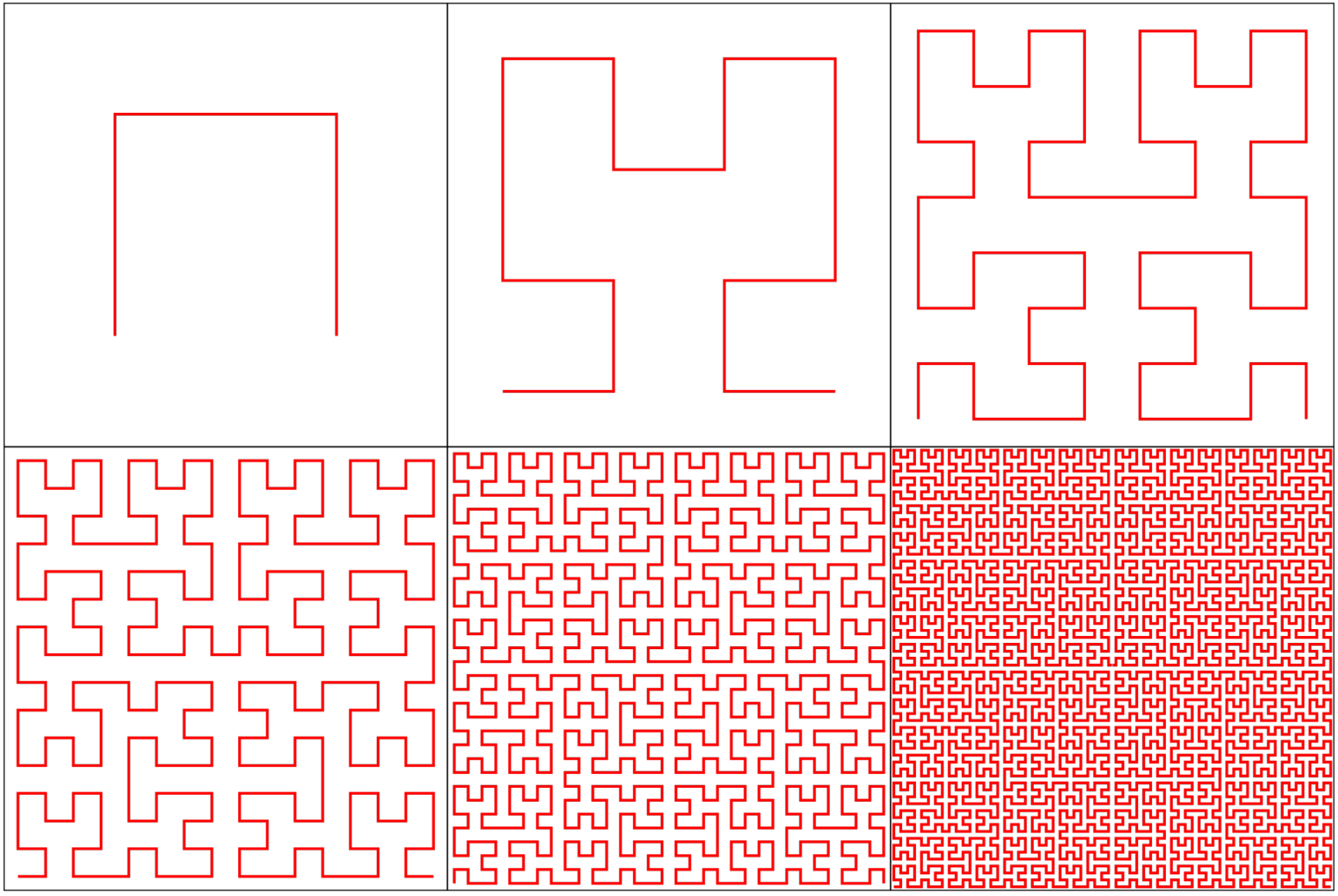


- tour through space
- recursive construction
- how to fill arbitrary target volumes?

Peano Curve



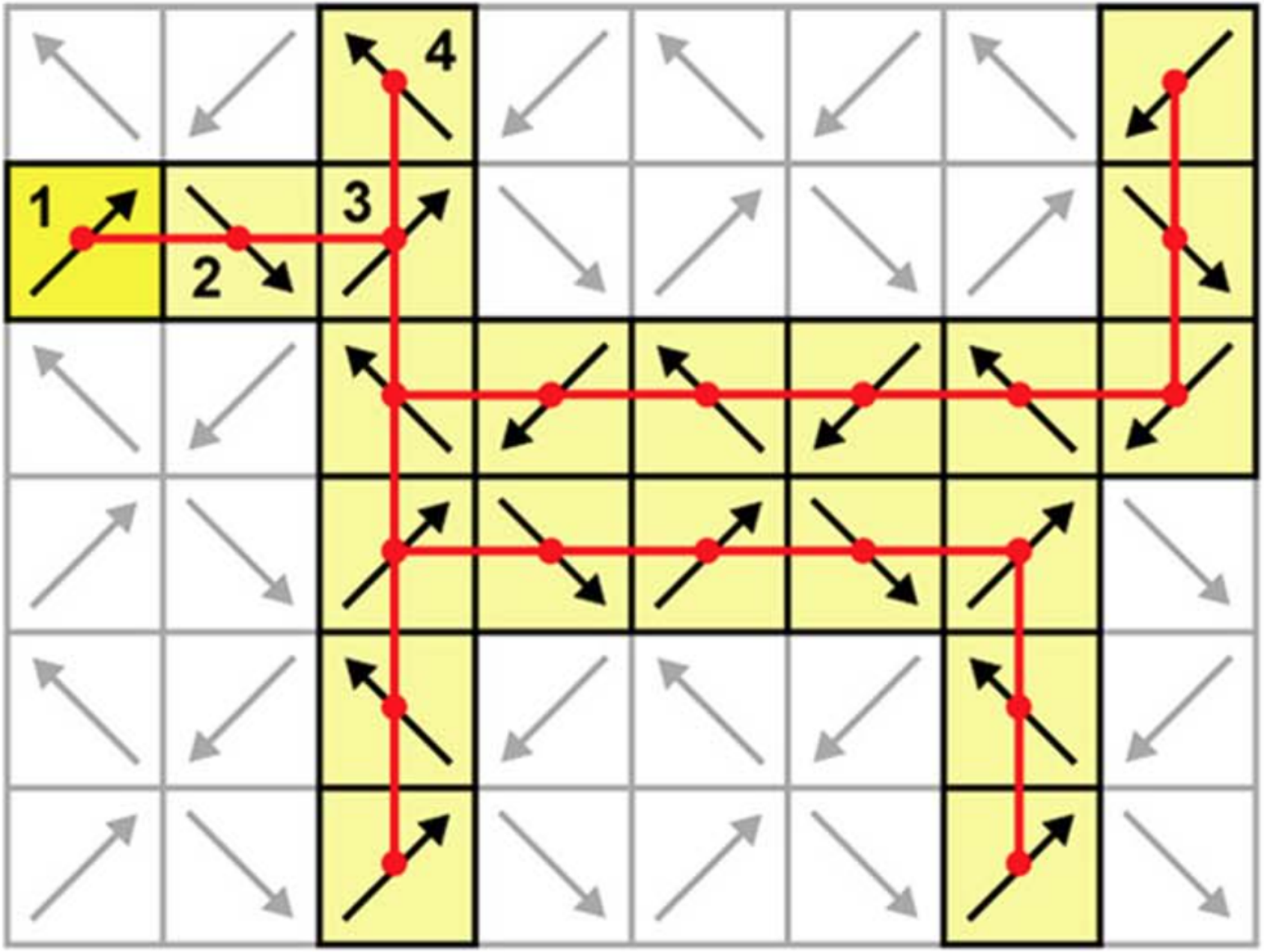
Hilbert Curve



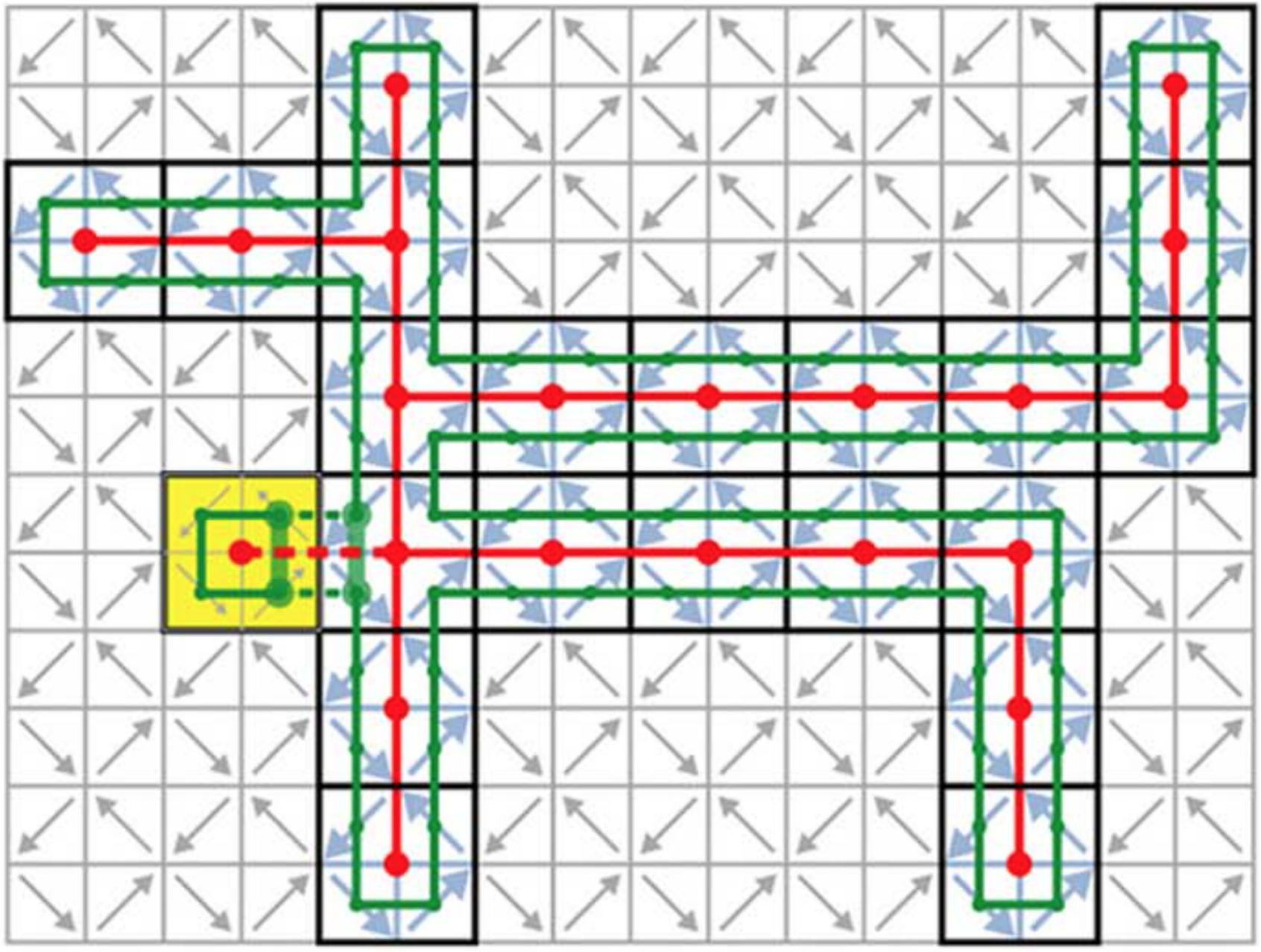
IEEE TRANSACTIONS ON ROBOTICS

Programmable Assembly With Universally Foldable Strings (Moteins)

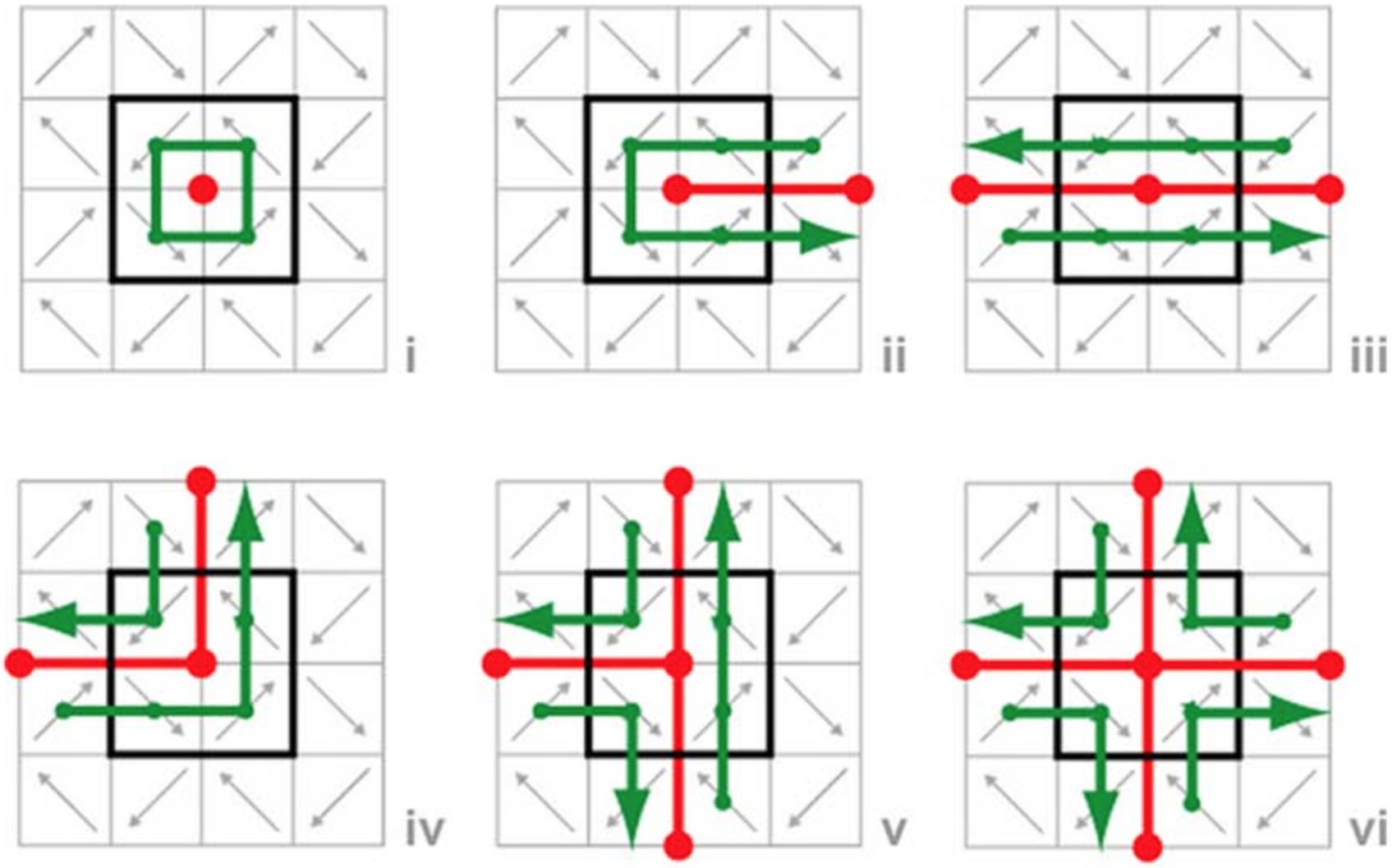
Kenneth C. Cheung, Erik D. Demaine, Jonathan R. Bachrach, and Saul Griffith


1
2
3
4

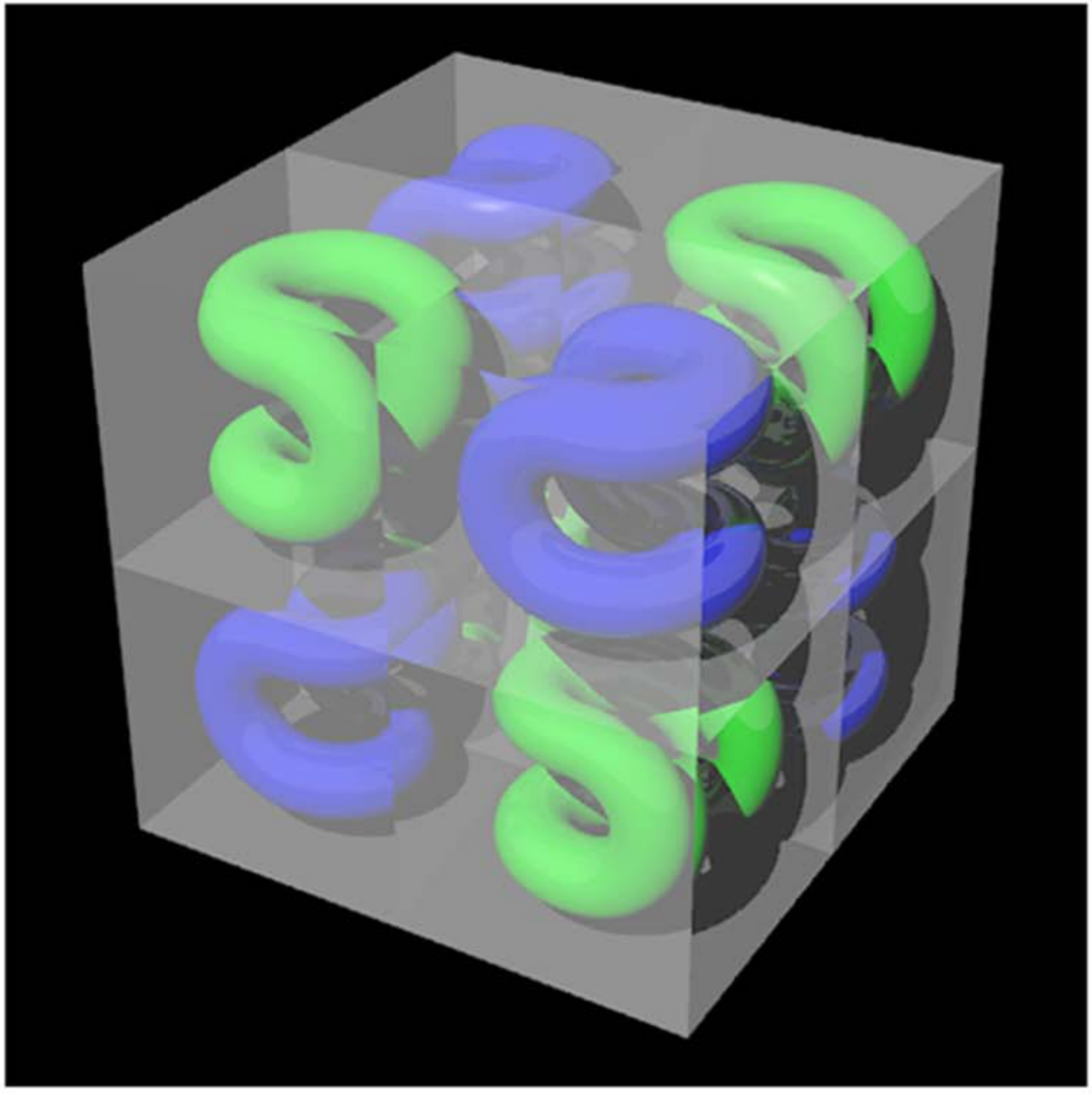
Pixel Subdivision

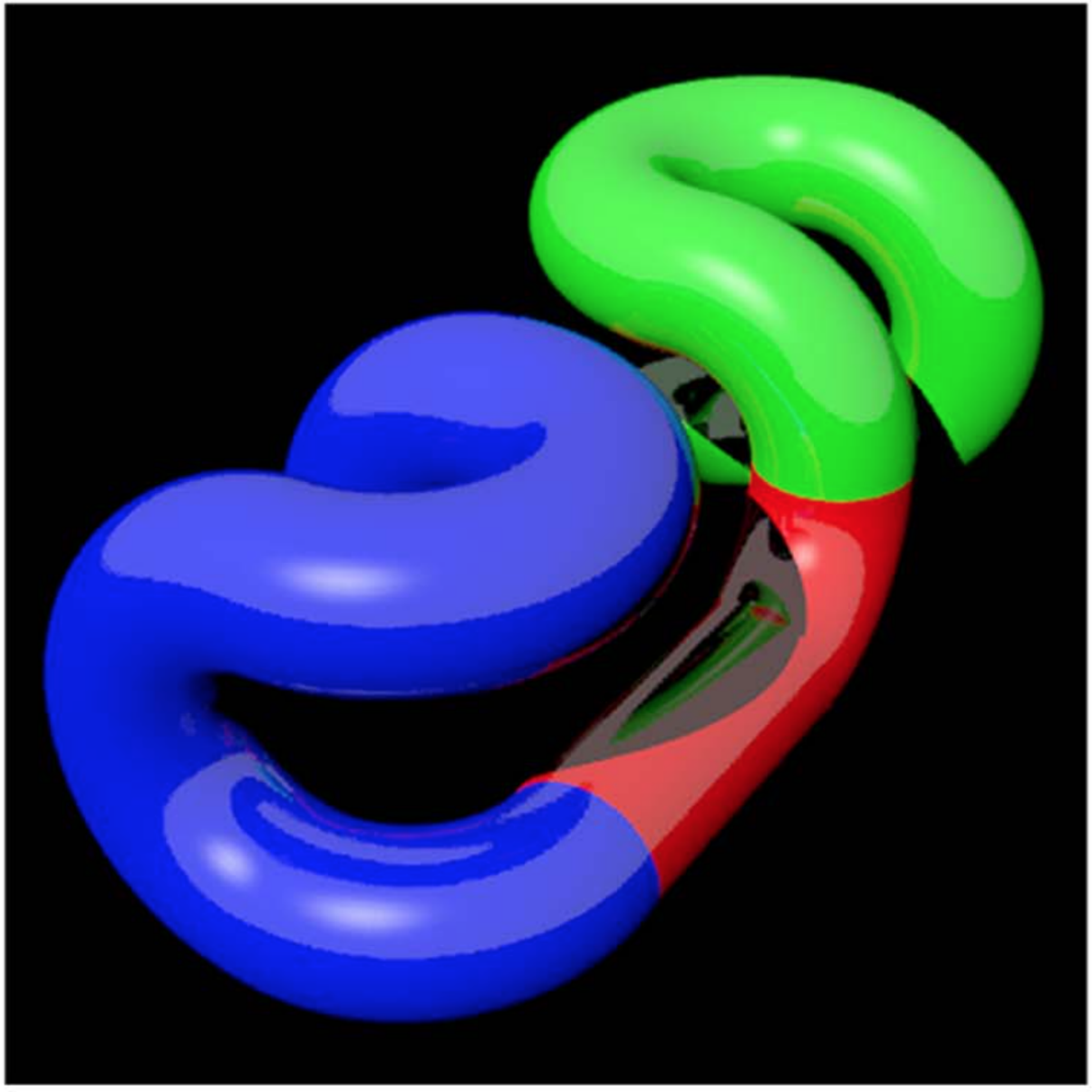


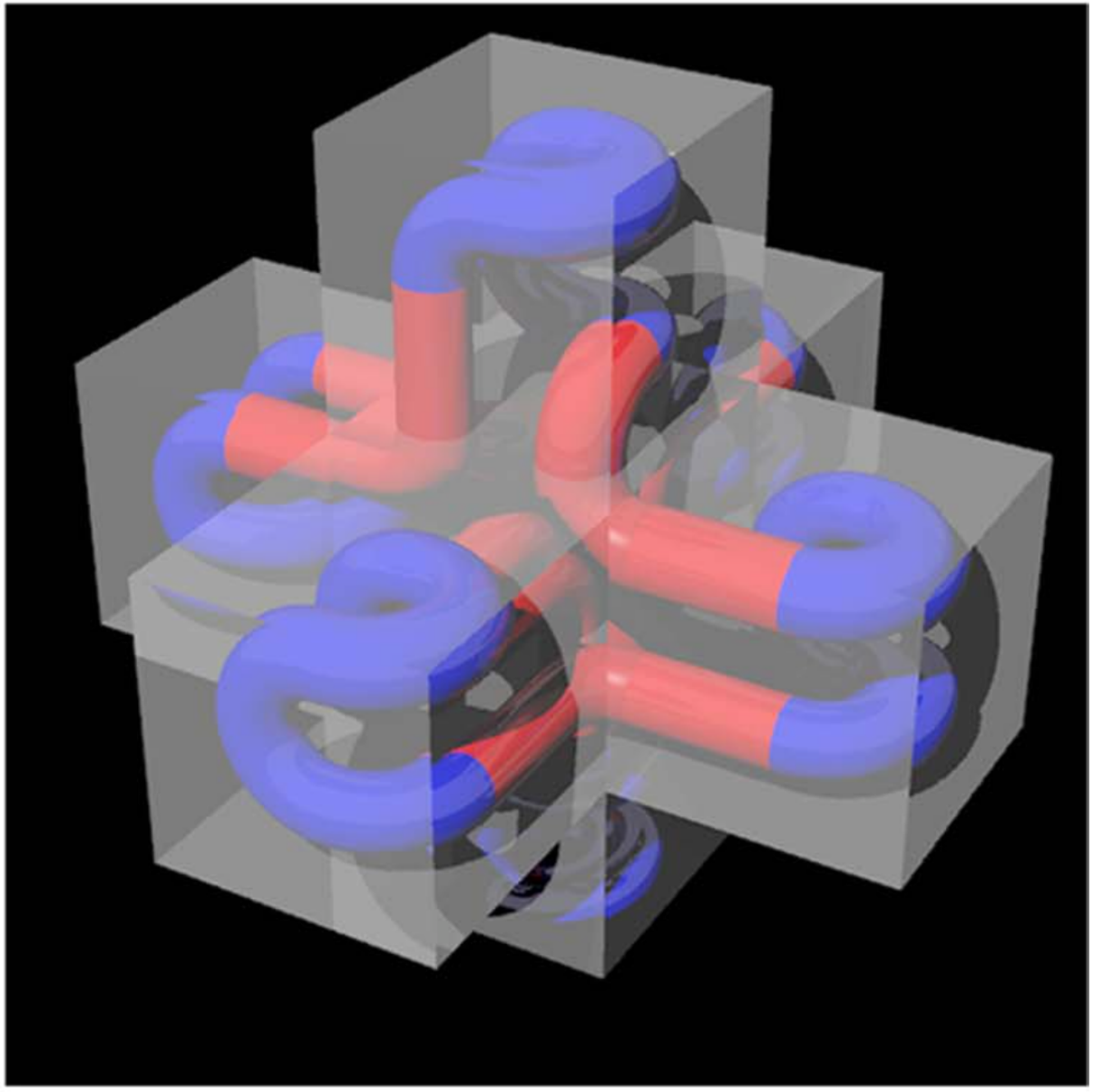
Pixel Connectivity

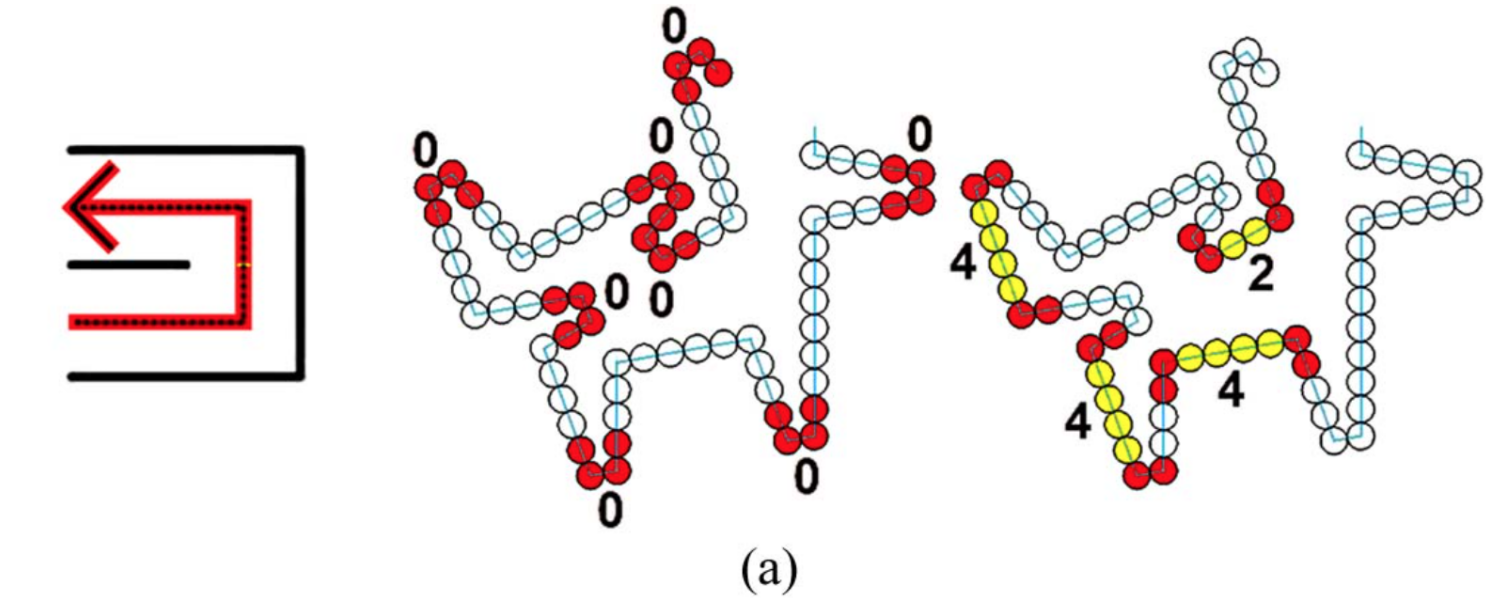


Voxel Subdivision

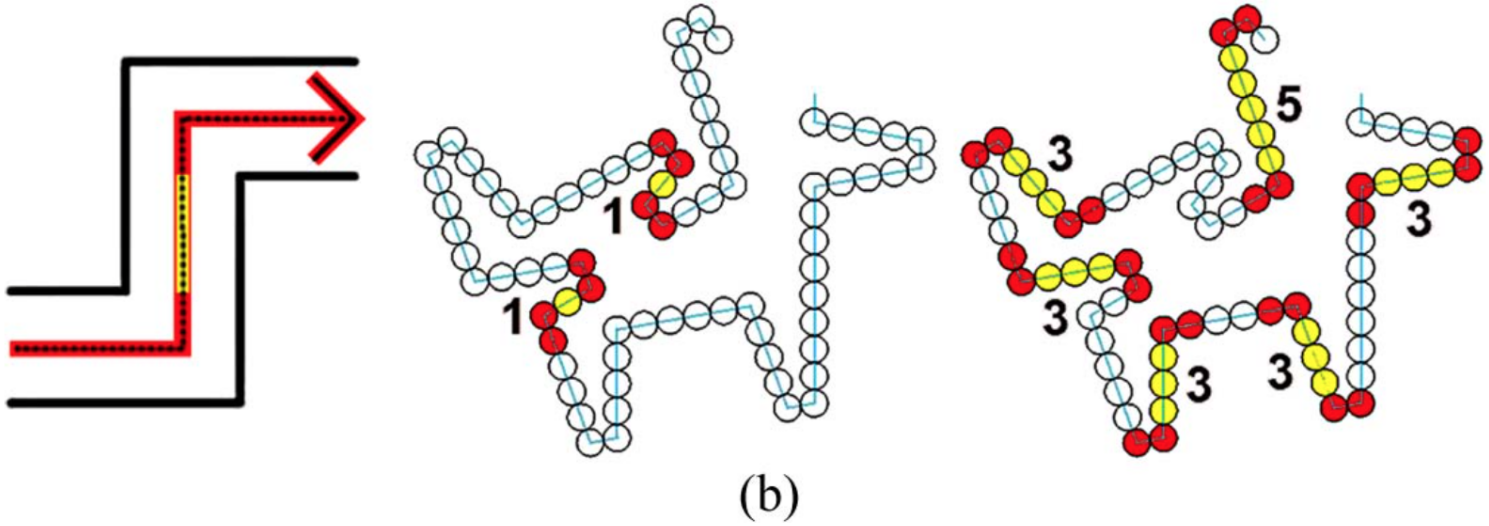




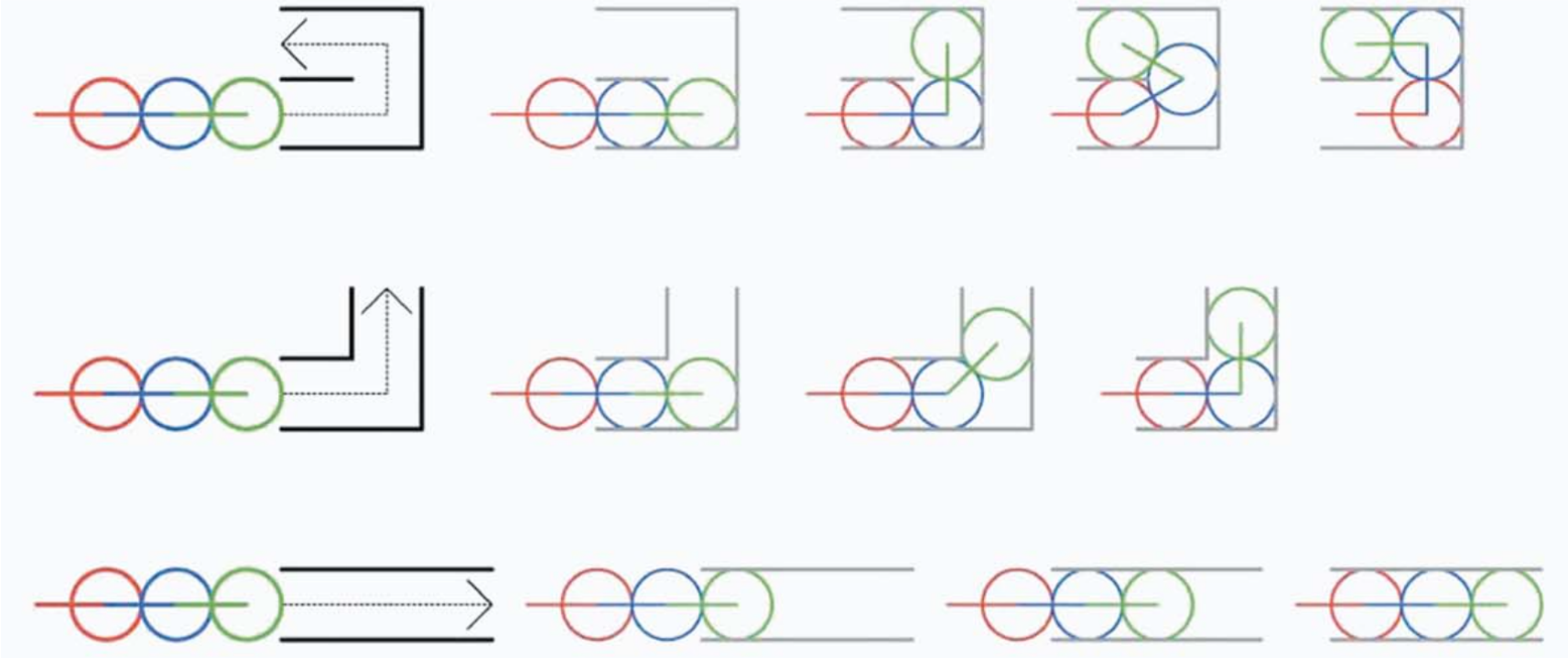


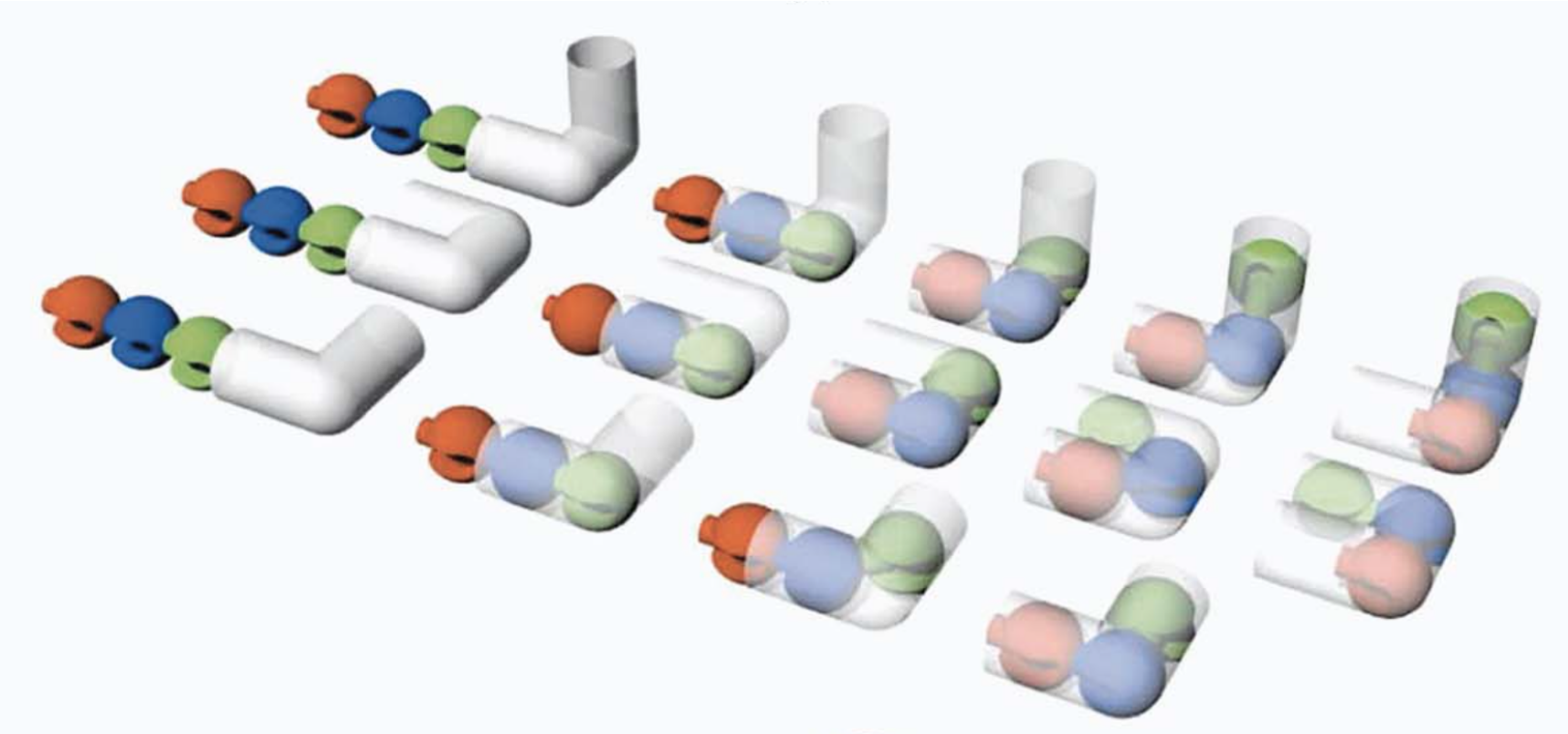


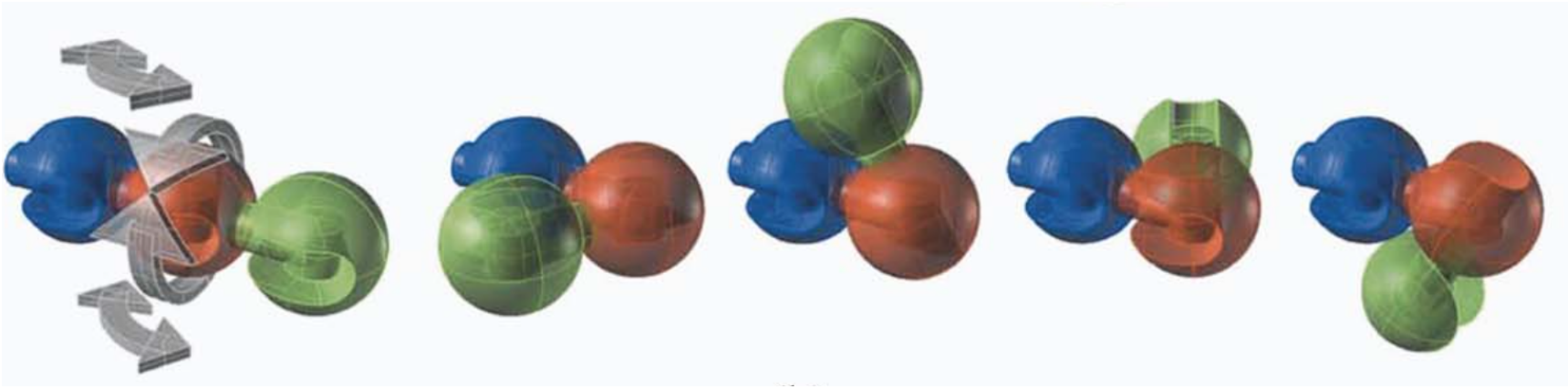
(a)

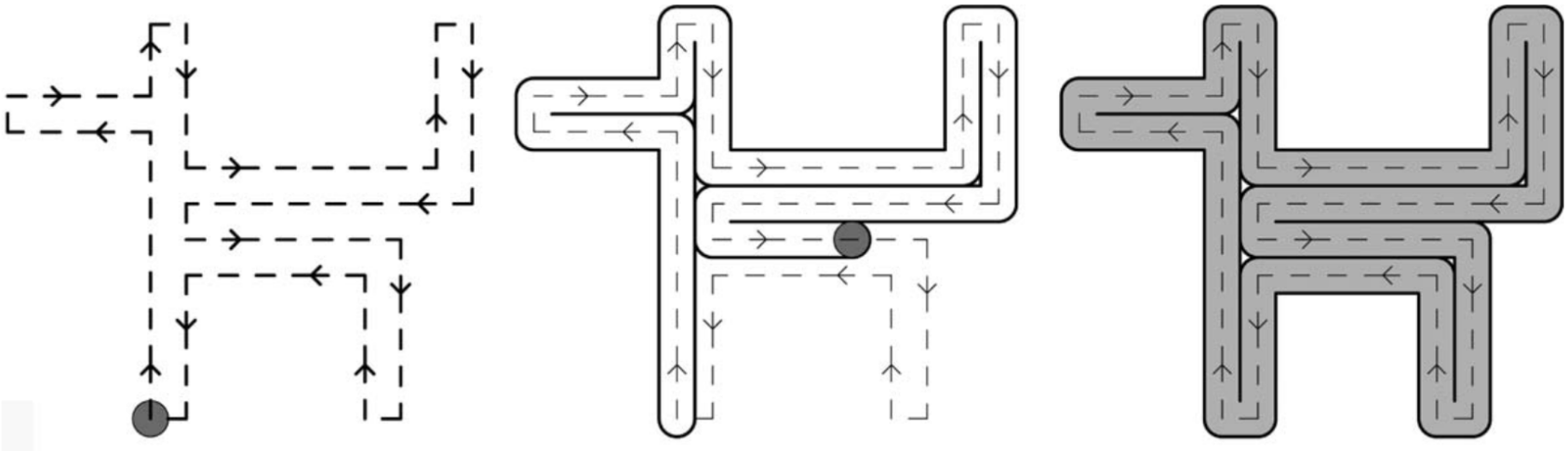


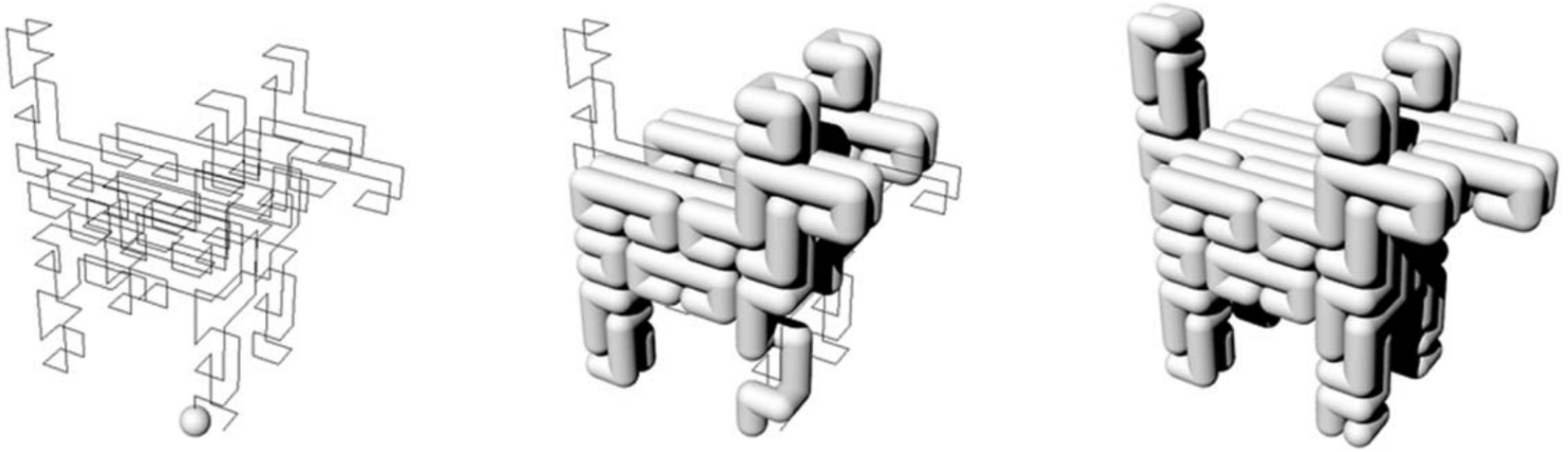
(b)

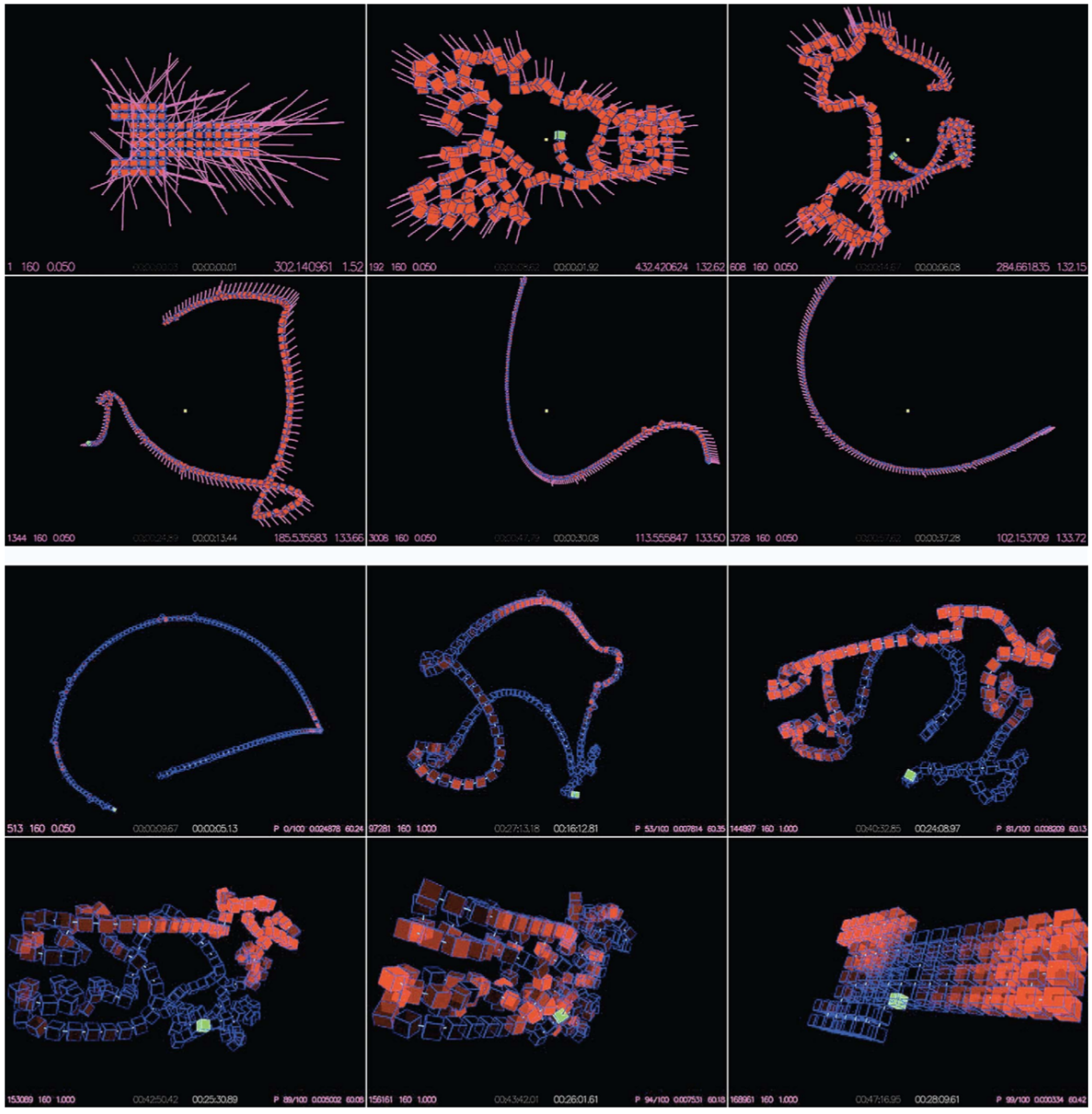




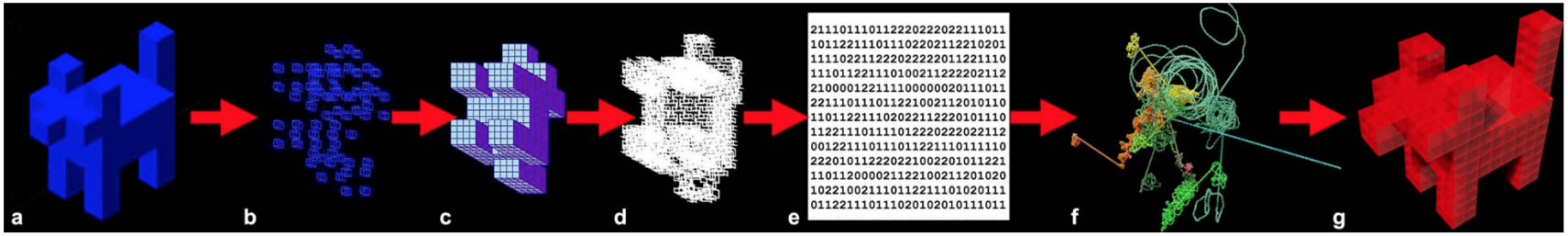







1 160 0.050 00:00:00.01 302.140961 1.52
192 160 0.050 00:00:01.92 432.420624 132.62
608 160 0.050 00:00:06.08 284.661835 132.15
1344 160 0.050 00:00:13.44 185.535583 133.66
3008 160 0.050 00:00:30.08 113.555847 133.50
3728 160 0.050 00:00:37.28 102.153709 133.72
513 160 0.050 00:00:05.13
97281 160 1.000 00:16:12.81
144897 160 1.000 00:24:08.97
153089 160 1.000 00:25:30.89
156161 160 1.000 00:26:01.61
168961 160 1.000 00:28:09.61

Entire Workflow



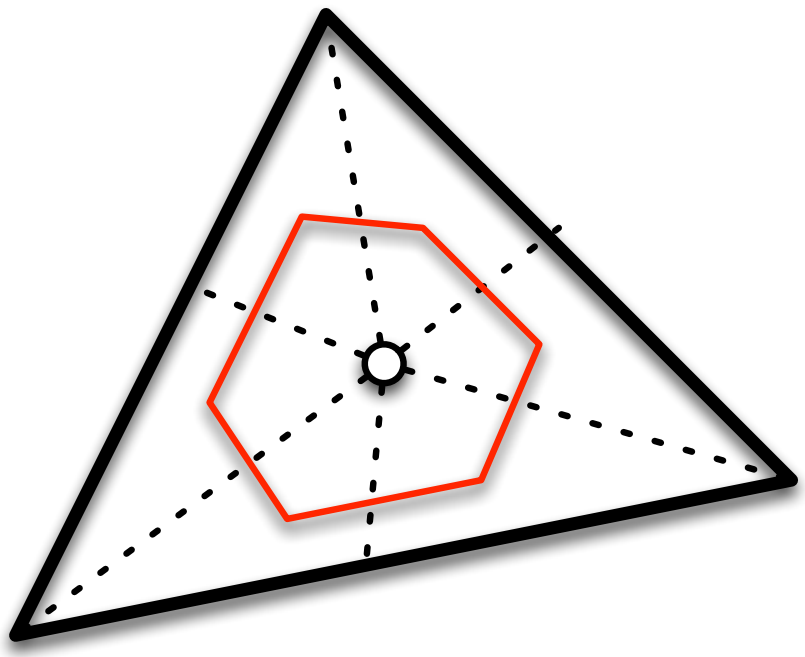
Demo

- stepping
- videos

- voxelize
- place primitive curve in voxels
- merge neighboring voxels

Meta Merging Algorithm On Surface

- triangular mesh
- break face into space fill triangles
- place primitive curve in faces
- merge neighboring faces

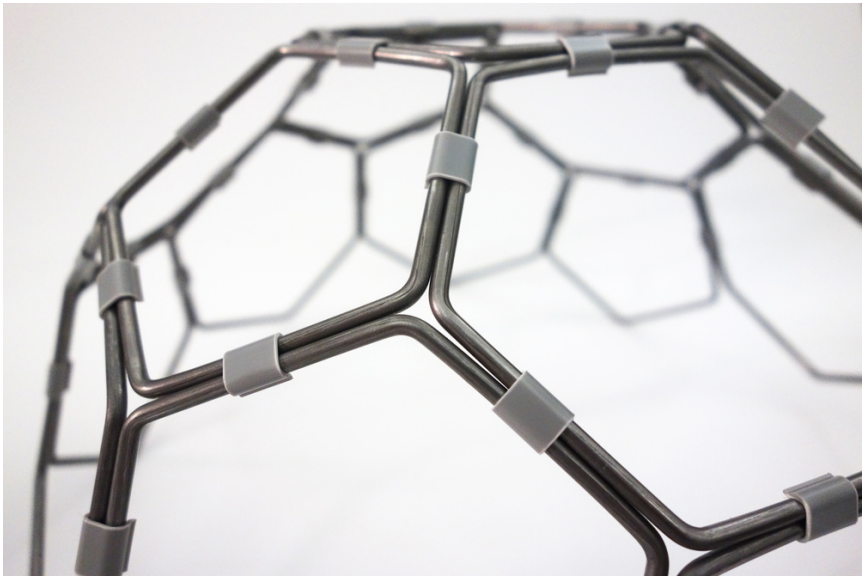


- ties
- clips
- shrink-wrap
- welding
- n way joins
- smashed tubes

Tied Bamboo Pavillion

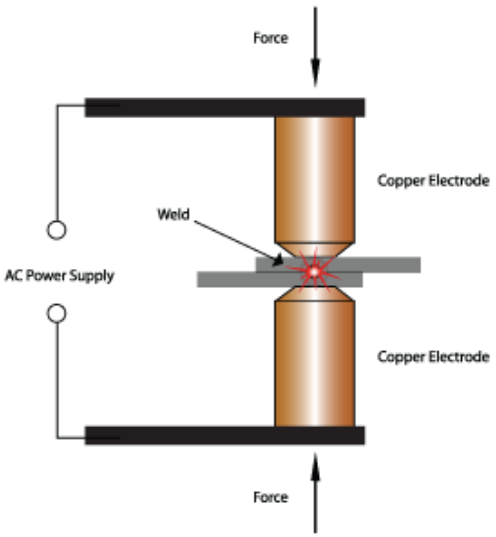
Tied Bamboo

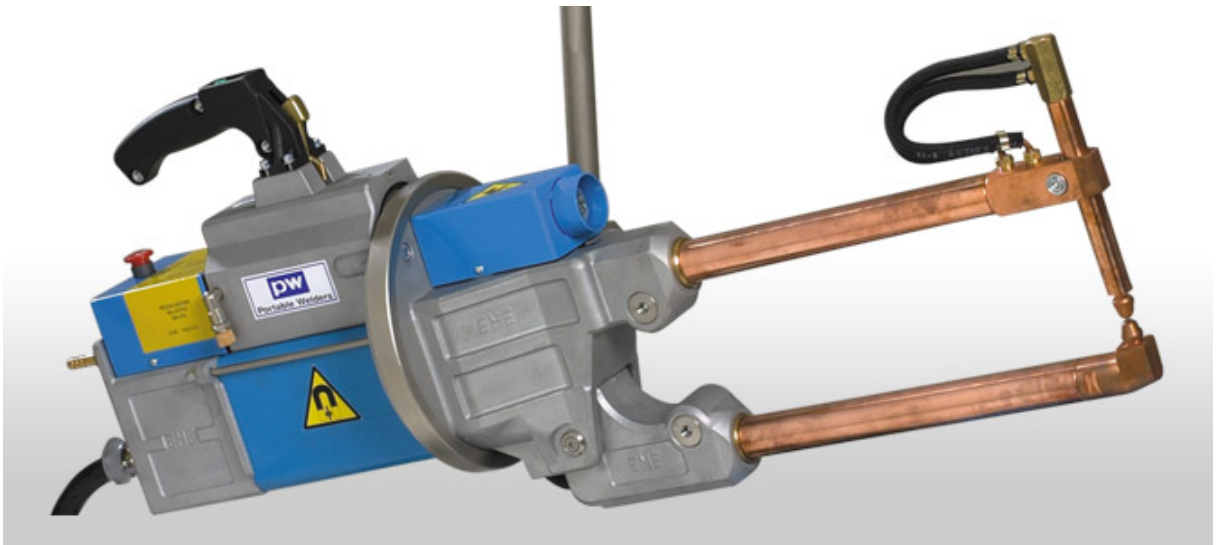
Clips



Shrink Wrap

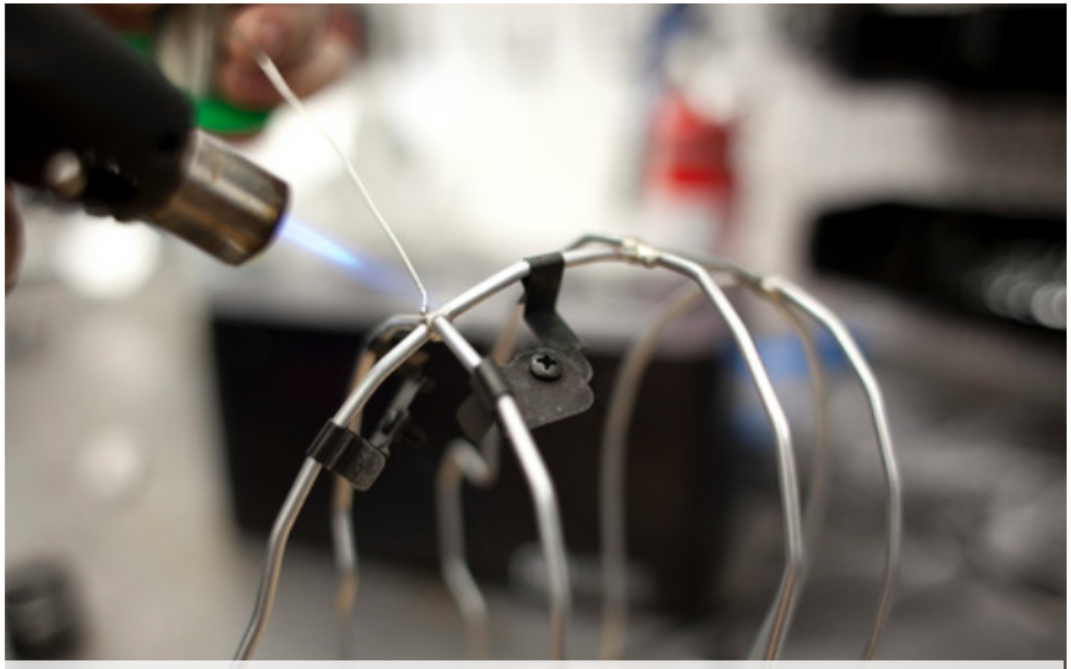
Spot Welding

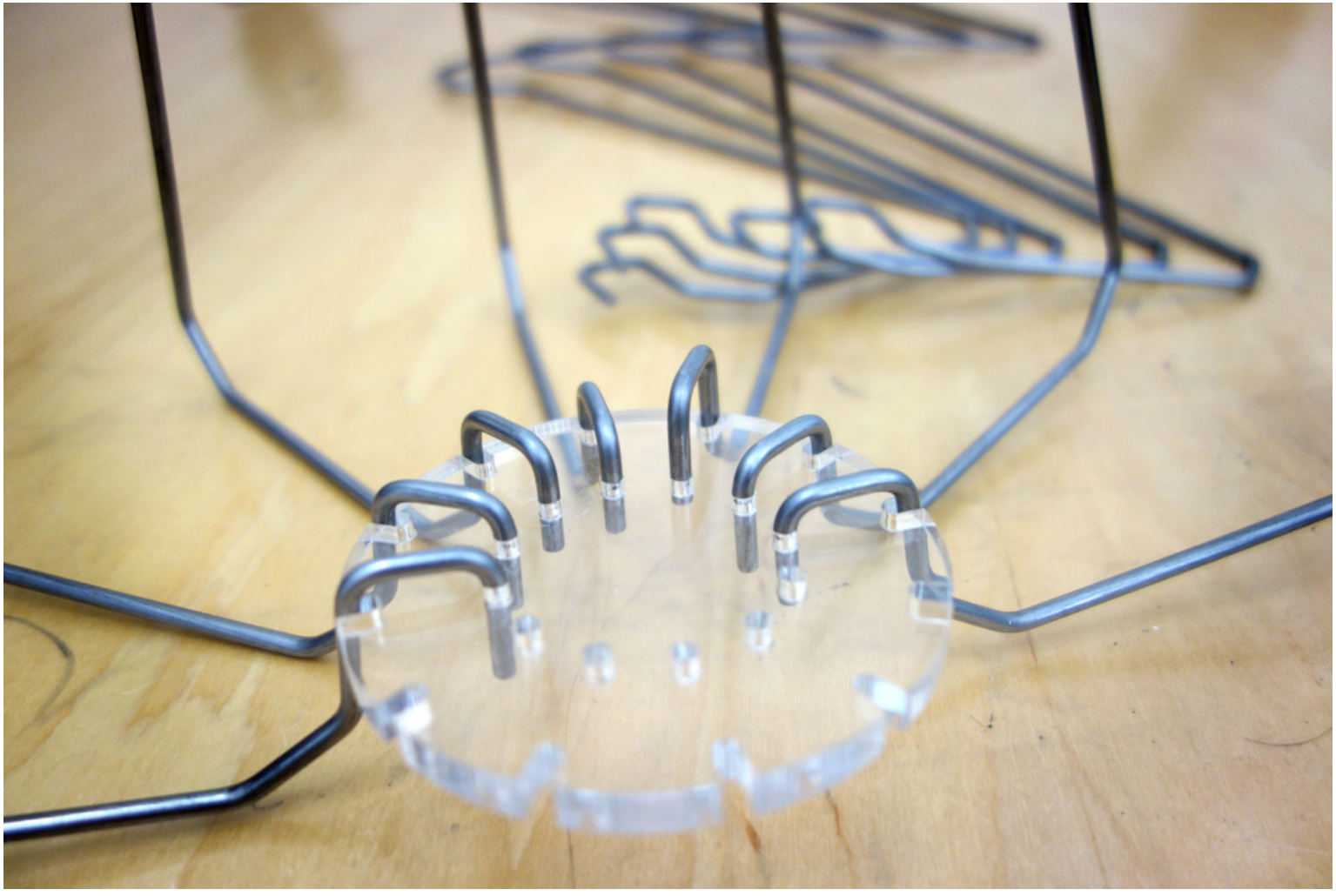




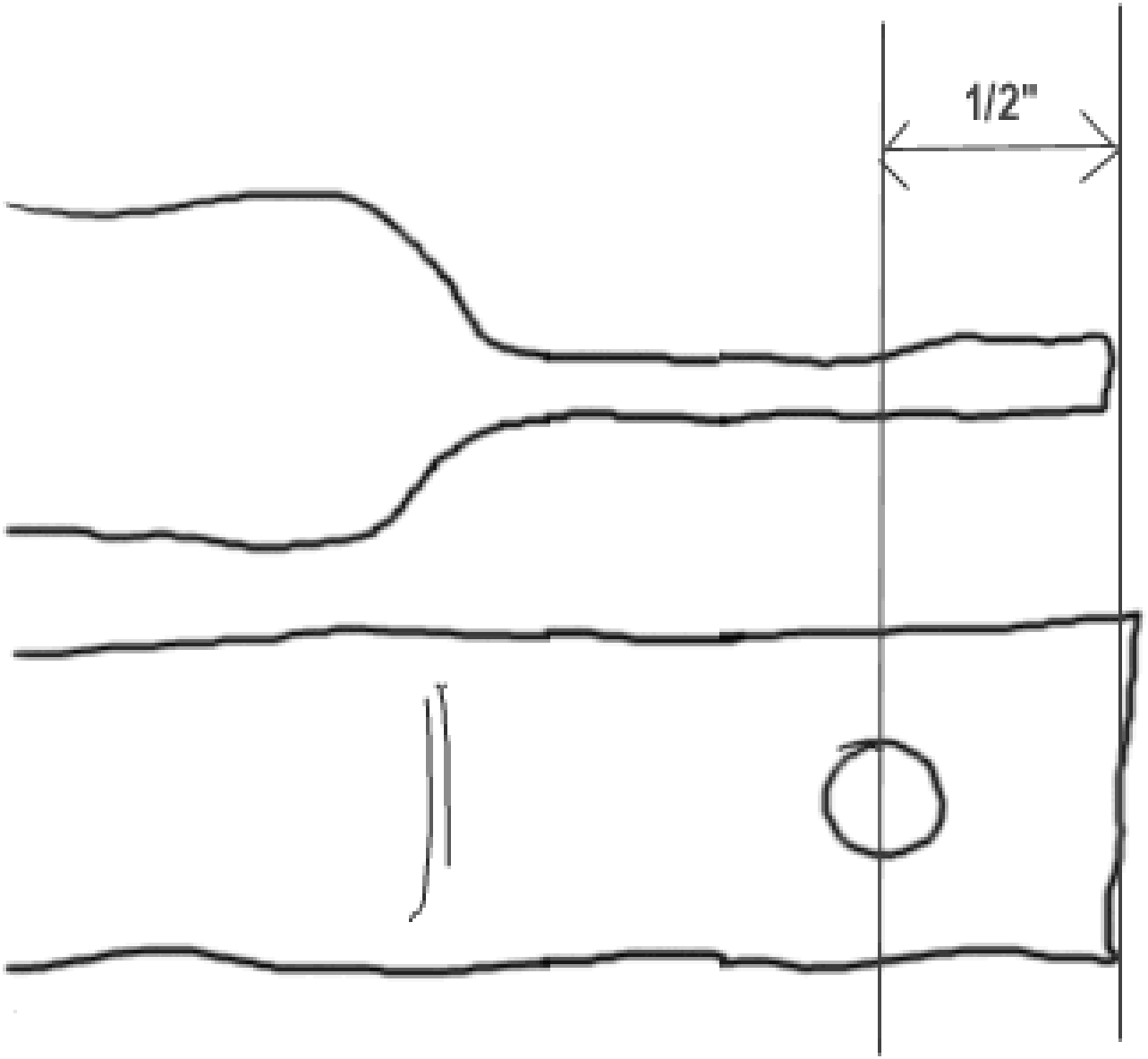
CC

Welding Helper Clips





Smashed Tubes



by waymoot

- step by step
- exploded diagram
- augmented reality
- paper template
- embodied parts

- register parts during construction
- project connection points

Create Template

- create template that could guide construction
- build out of paper or stickers
- add marks to template

- notches
- scoring
- invented code