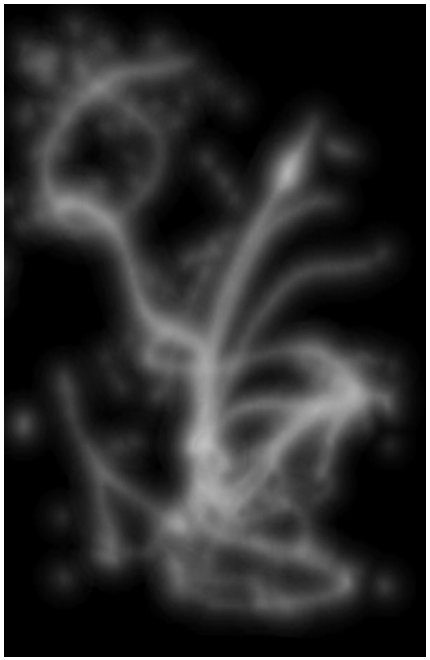
SIGGRAPH ASIA 2020 VIRTUAL
Pre-recorded sessions: From 4 December 2020 | Live sessions: 10 – 13 December 2020
SA2020.SIGGRAPH.ORG
#SIGGRAPHAsia | #SIGGRAPHAsia2020

To cut or to fill: a global optimization approach to topological simplification

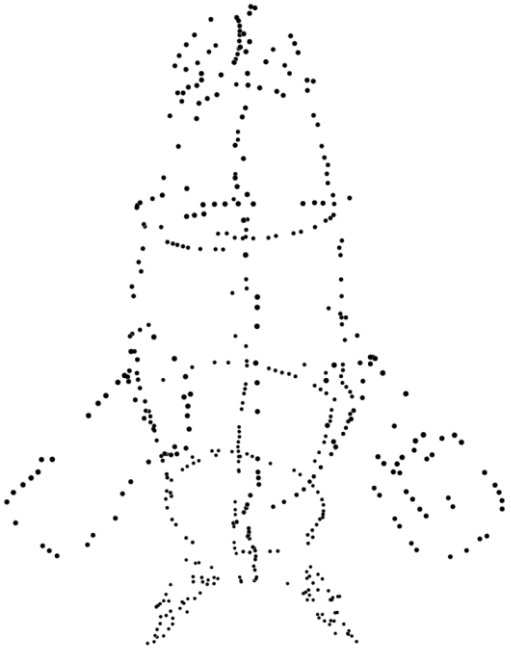
DAN ZENG [1], ERIN CHAMBERS [2], DAVID LETSCHER [2], TAO JU [1]

[1] Washington University in St. Louis
[2] Saint Louis University

Motivation

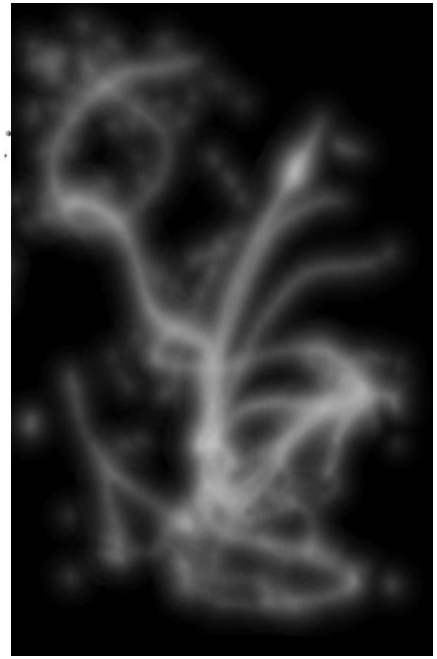


CT Scan

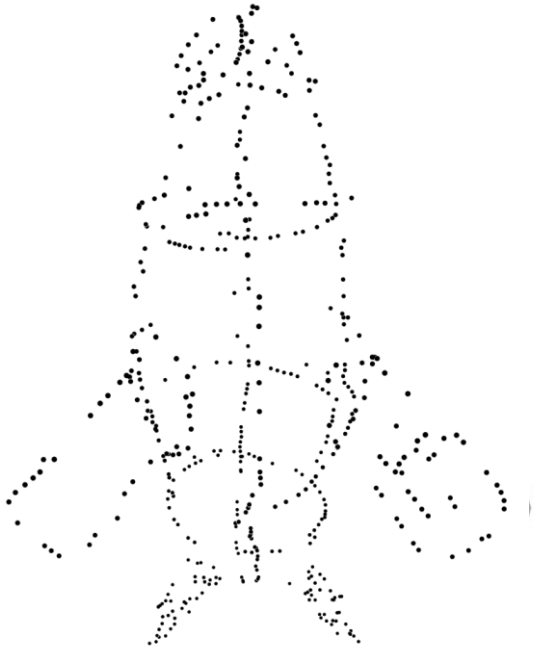


Point cloud

Motivation

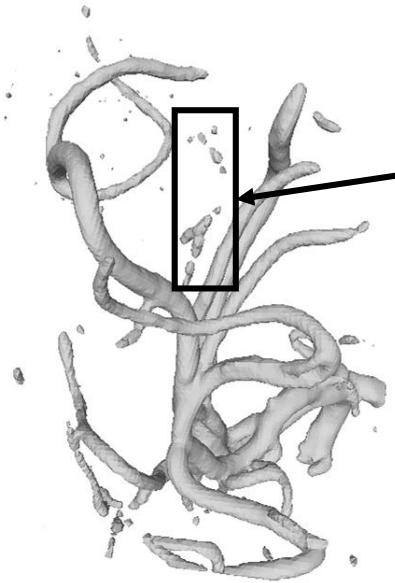


CT Scan
(Iso-surface)

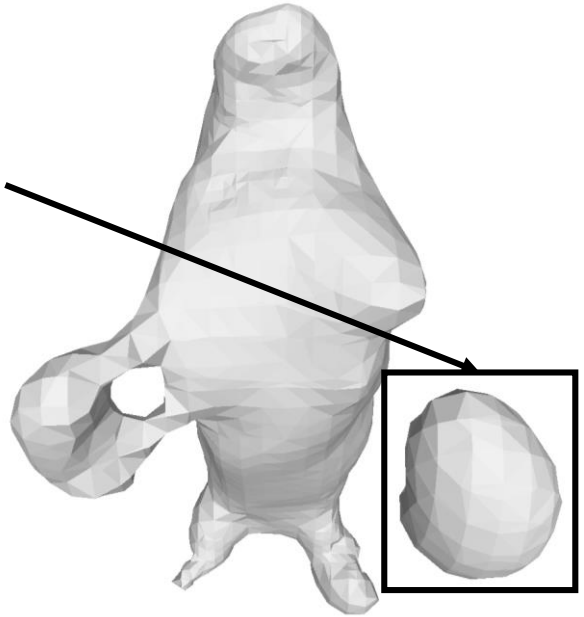


Point cloud
(Poisson Reconstruction)

Motivation



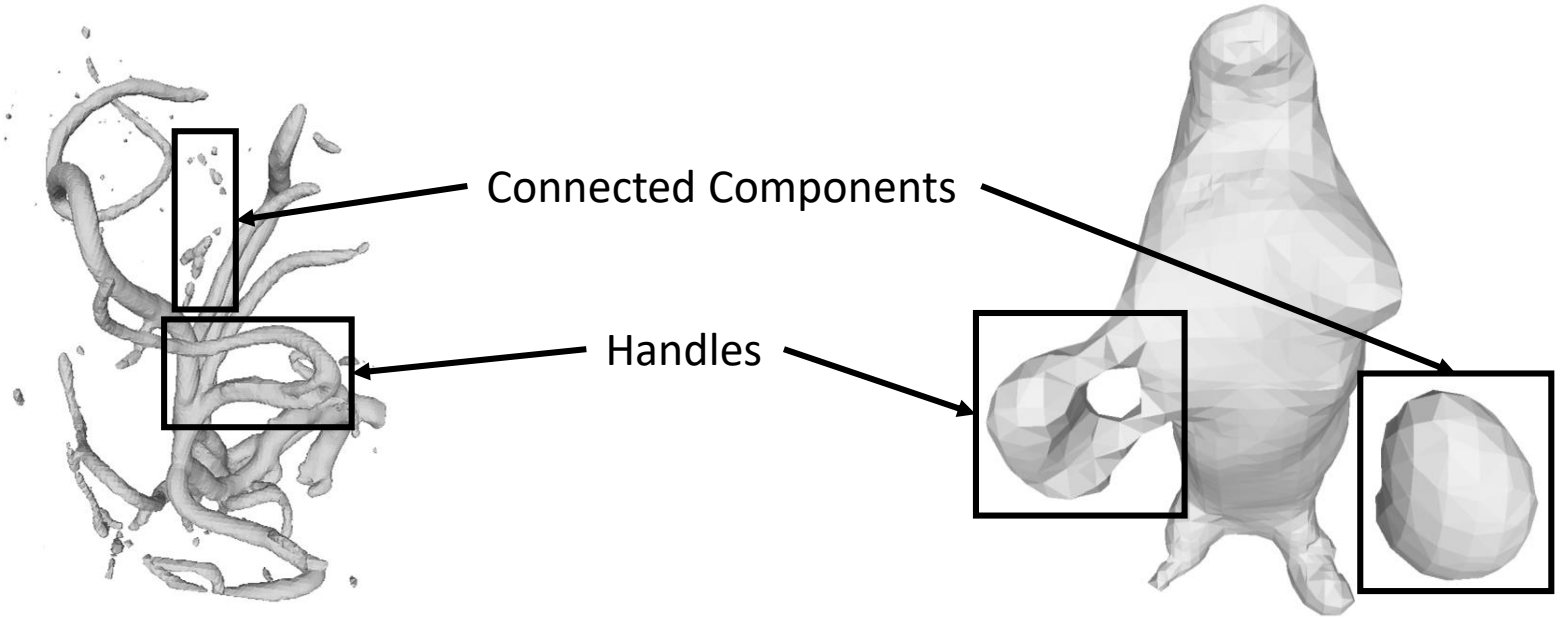
Connected Components



CT Scan
(Iso-surface)

Point cloud
(Poisson Reconstruction)

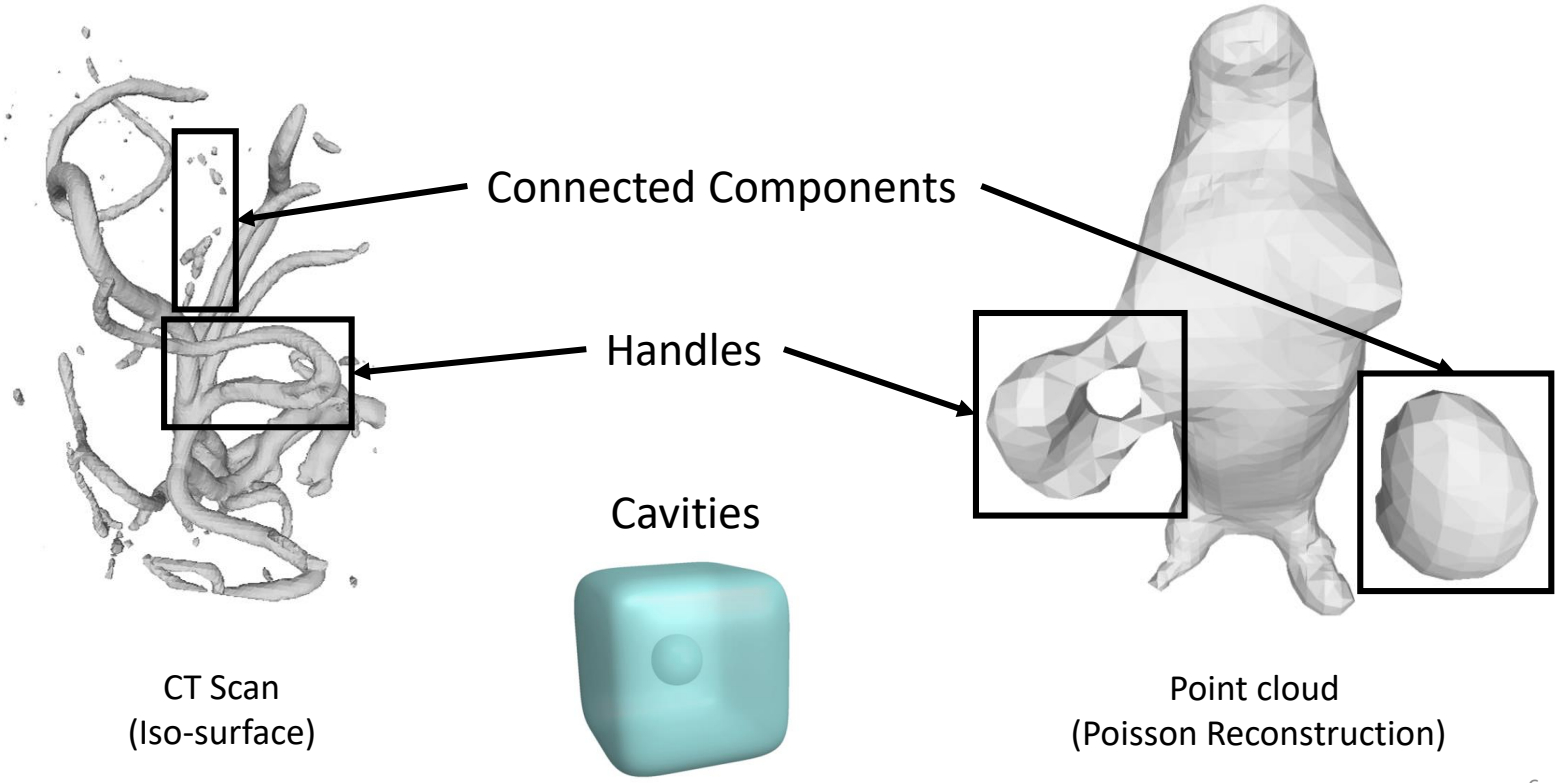
Motivation



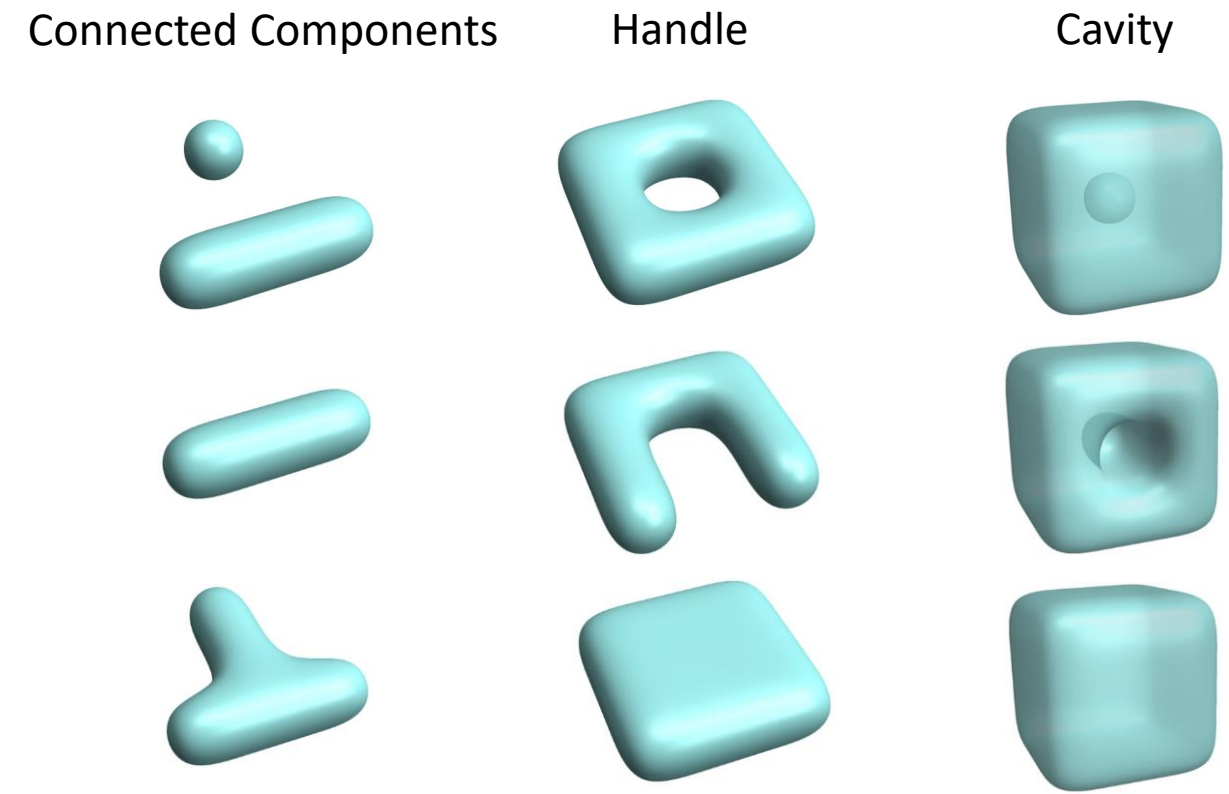
CT Scan
(Iso-surface)

Point cloud
(Poisson Reconstruction)

Motivation

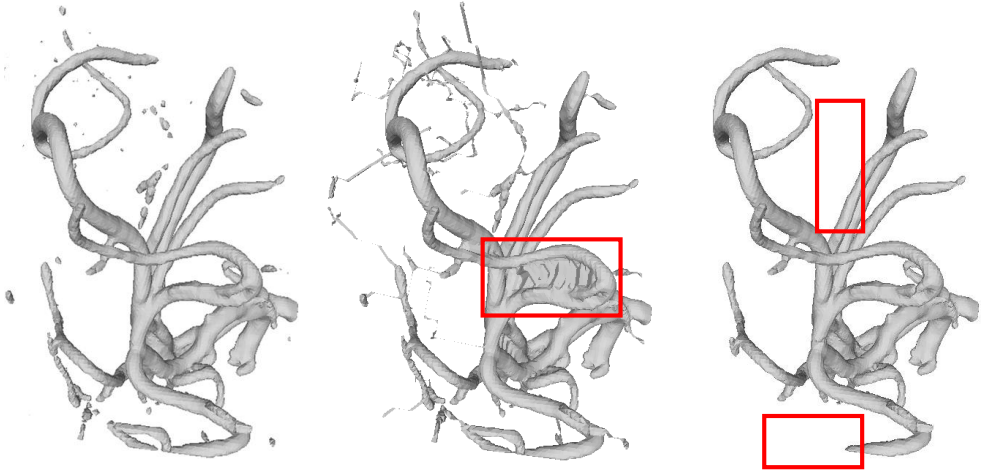

Connected Components
Handles
Cavities
CT Scan
(Iso-surface)
Point cloud
(Poisson Reconstruction)

Cutting and Filling



Related works

- Monotonic methods: only cutting, or only filling
  - [Chen et al. 2006], [Shattuck et al. 2001], [Han et al. 2002], [Zhou et al. 2007], [Nooruddin et al. 2003] [Bischoff et al. 2002], [Kriegeskorte et al. 2001], [Sczymzak et al. 2003], [Chambers et al. 2018]
  - Result in very large geometric changes



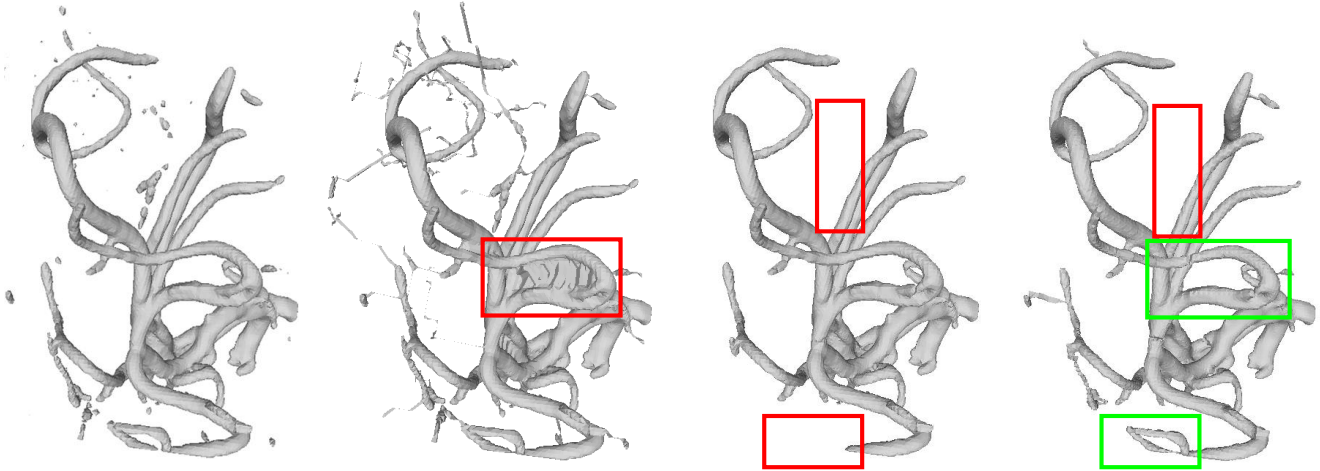
Input

Monotonic
(filling only)

Monotonic
(cutting only)

Related works

- Non-Monotonic methods: both cutting and filling
  - [Wood et al. 2004], [Kriegeskorte and Goebel 2001], [Ségonne et al. 2007], [Ju et al. 2007]
  - Existing methods apply greedy heuristics and can still result in excessive changes



Input

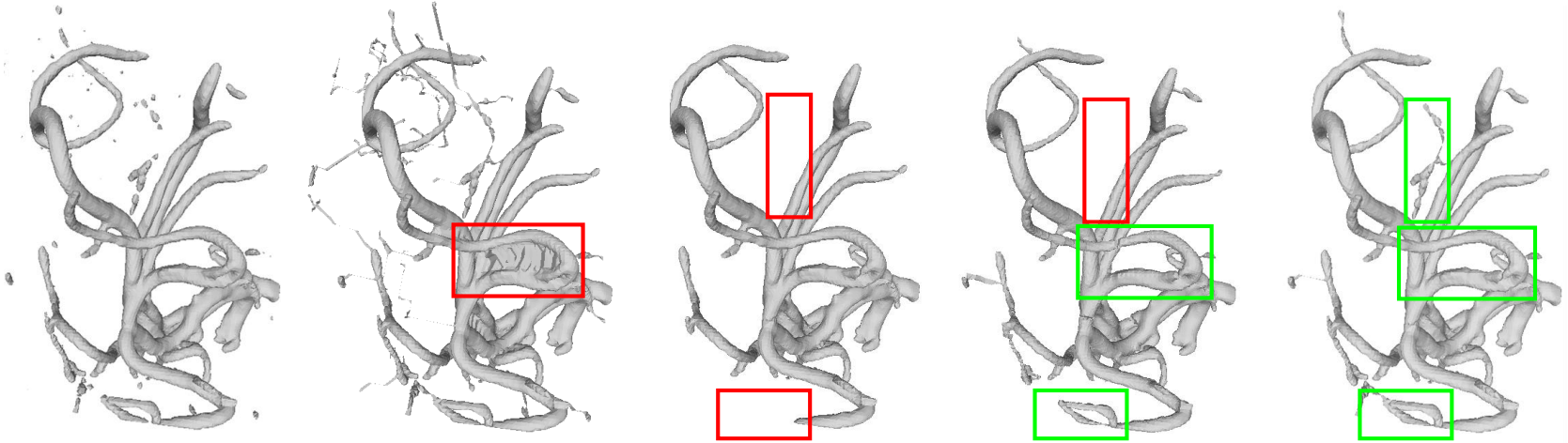
Monotonic
(filling only)

Monotonic
(cutting only)

Non-monotonic

Contributions

- A non-monotonic algorithm
  - Attempts to find the globally optimal set of cuts and fills that maximally simplifies topology while minimizing geometric changes



Input

Monotonic (filling only)

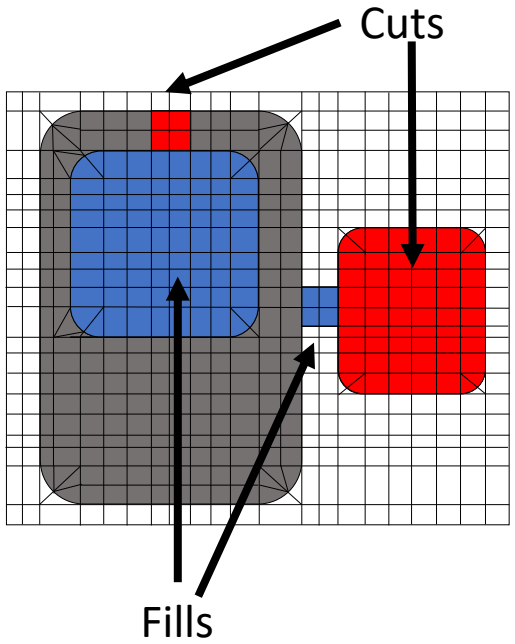
Monotonic (cutting only)

Non-monotonic

Ours

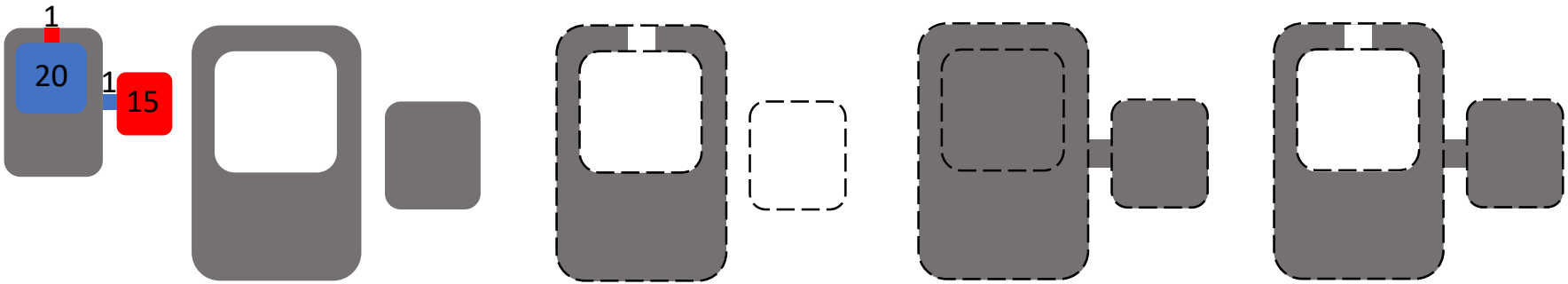
Input

- Shape represented as a cell complex
  - 2D: pixels, triangles, etc.
  - 3D: voxels, tetrahedra, etc.
- Cuts and fills
  - Cells to be removed or added to the shape
  - Each cut or fill cell is associated with a geometric cost
  - Can be obtained using existing monotonic simplification methods



Goal

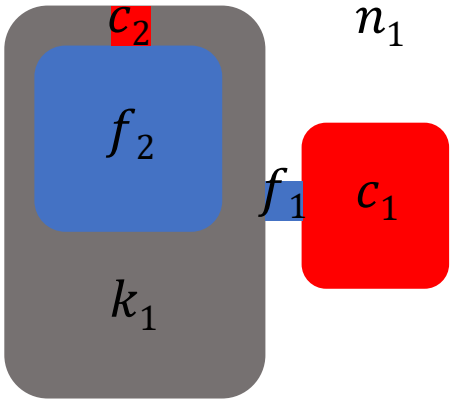
- Find a subset of the cuts and fills that
  - Minimizes the total number of topological features (1st priority)
  - Minimizes the total geometric cost (2nd priority)

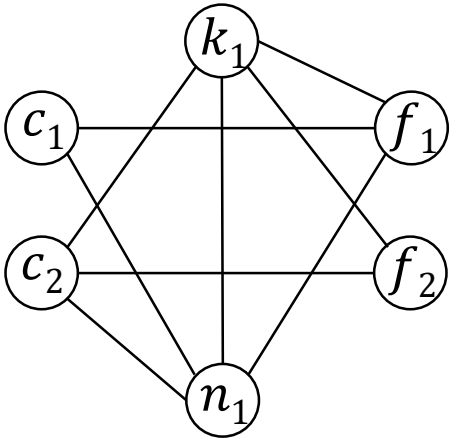


	Input	Only cutting	Only filling	Optimal
# components:	2	1	1	1
# cavities:	1	0	0	0
# geometric cost:	0	16	21	2

Graph formulation

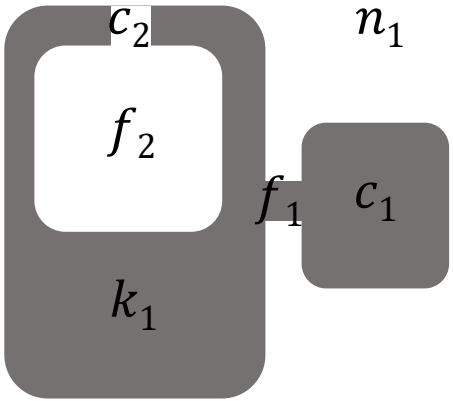
- Representing partitioning of space into regions of different types
  - Cut and fill nodes: connected components of cuts and fills
  - Kernel nodes: connected components of shape minus cuts
  - Neighborhood nodes: connected components of background minus fills

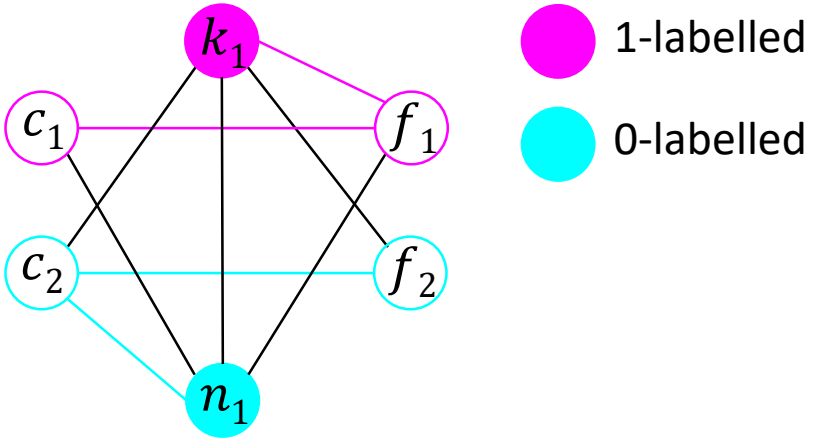




Graph formulation

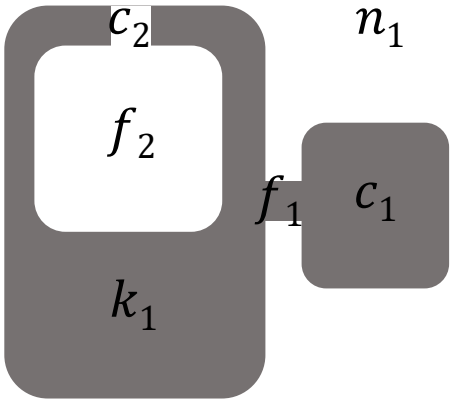
- 1/0 labelling of nodes represents selection of cuts and fills
  - 1-labelled fill nodes and 0-labelled cut nodes are selected
  - Kernel nodes are labelled 1, neighborhood nodes labelled 0

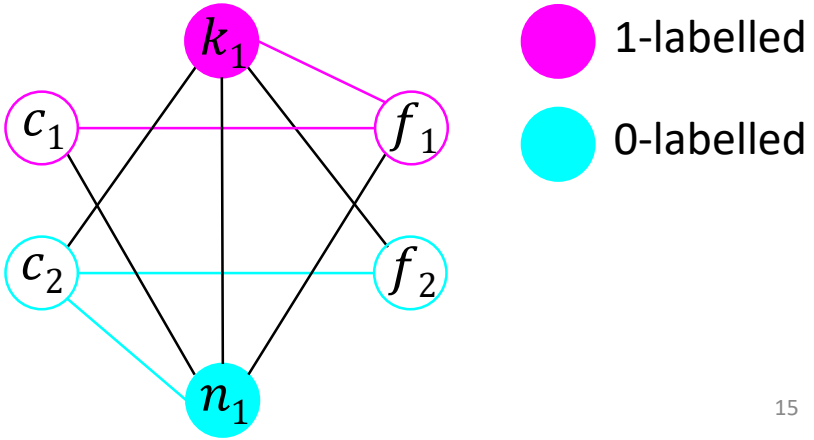




Graph formulation

- A node-wise cost is defined for each cut or fill node
  - Obtained from the Euler characteristic and geometric cost of the node



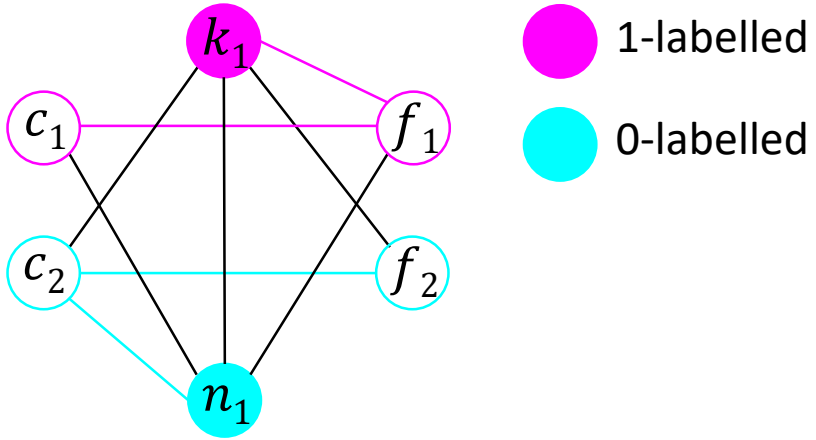


Graph formulation

- Problem: Find a 1/0 labelling that minimizes the sum of:
  - Number of connected components of the 1-labelled and 0-labelled subgraphs
  - Total labelling costs of all nodes

1-labelled components: 1

0-labelled components: 1



Graph Labelling

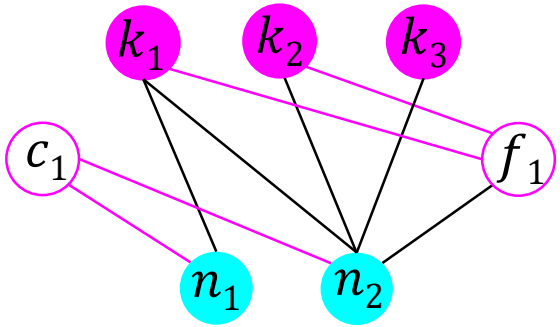
- Treat connectivity as constraints, instead of an energy term
  - Constrain both 0/1-labelled subgraphs to be *as connected as possible*
  - Solve the connectivity-constrained labelling problem using Steiner Tree

Graph Labelling

- Reachable sets
  - A set of kernel (resp. neighborhood) nodes that are connected when all cut and fill nodes are labelled 1 (resp. 0).

Graph Labelling

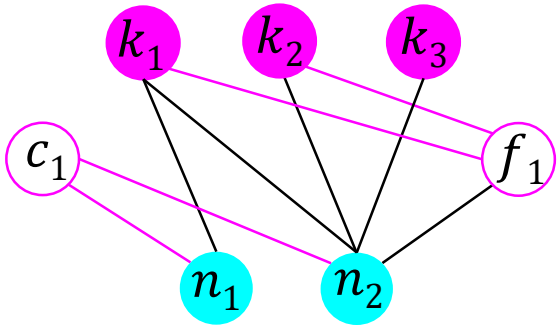
- Reachable sets
  - A set of kernel (resp. neighborhood) nodes that are connected when all cut and fill nodes are labelled 1 (resp. 0).



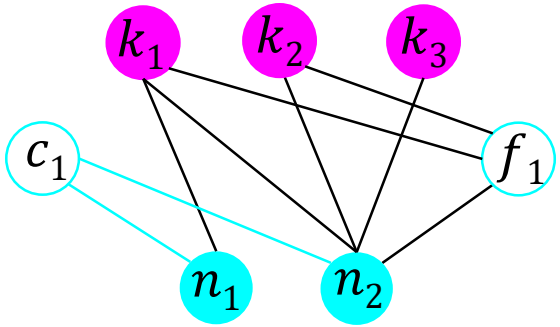
{ k1, k2 } and { k3 } are reachable sets

Graph Labelling

- Reachable sets
  - A set of kernel (resp. neighborhood) nodes that are connected when all cut and fill nodes are labelled 1 (resp. 0).



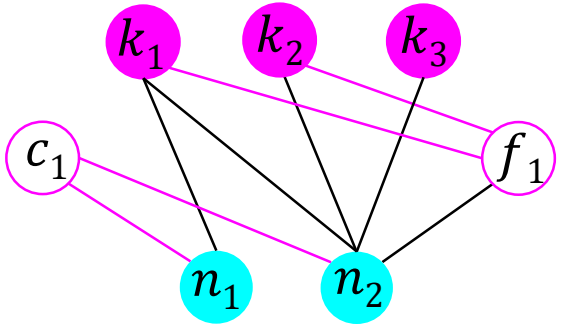
{ k1, k2 } and { k3 } are reachable sets



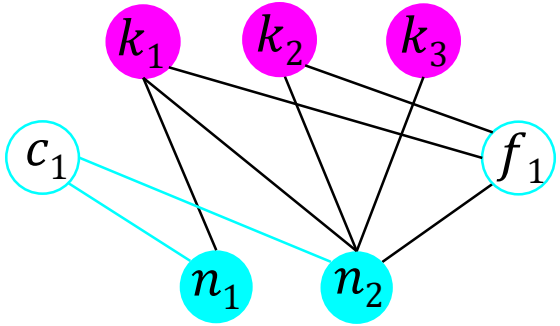
{n1, n2} are in reachable set

Graph Labelling

- Reachable sets
  - A set of kernel (resp. neighborhood) nodes that are connected when all cut and fill nodes are labelled 1 (resp. 0).
  - Such that any reachable set of kernel (resp. neighborhood) nodes are connected in the 1(resp. 0)-labelled subgraph



{ k1, k2 } and { k3 } are reachable sets



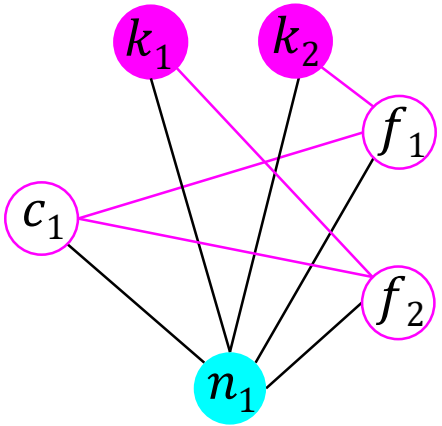
{n1, n2} are in reachable set

Graph Labelling

- As-connected-as-possible (ACAP) labelling
  - Minimizes the total labelling cost
  - Such that any reachable set of kernel (resp. neighborhood) nodes are connected in the 1(resp. 0)-labelled subgraph

Graph Labelling

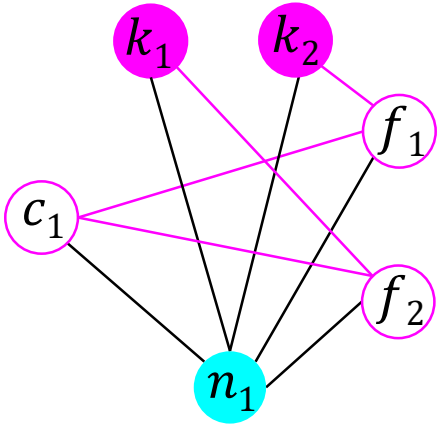
- As-connected-as-possible (ACAP) labelling
  - Minimizes the total labelling cost
  - Such that any reachable set of kernel (resp. neighborhood) nodes are connected in the 1(resp. 0)-labelled subgraph



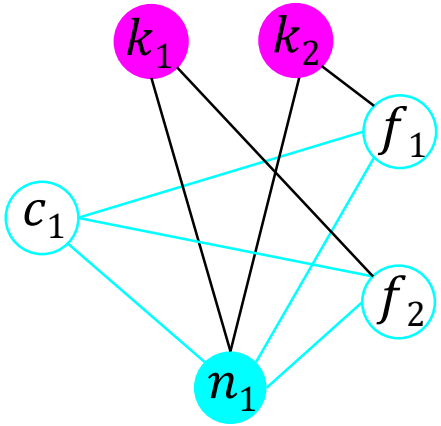
ACAP

Graph Labelling

- As-connected-as-possible (ACAP) labelling
  - Minimizes the total labelling cost
  - Such that any reachable set of kernel (resp. neighborhood) nodes are connected in the 1(resp. 0)-labelled subgraph



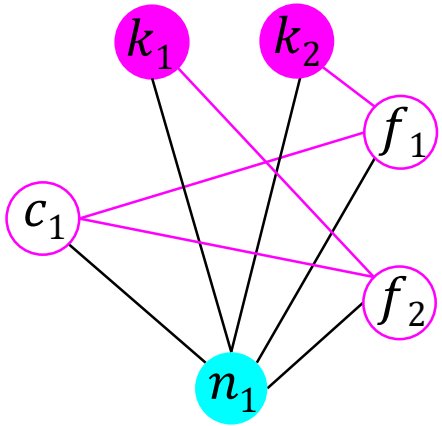
ACAP



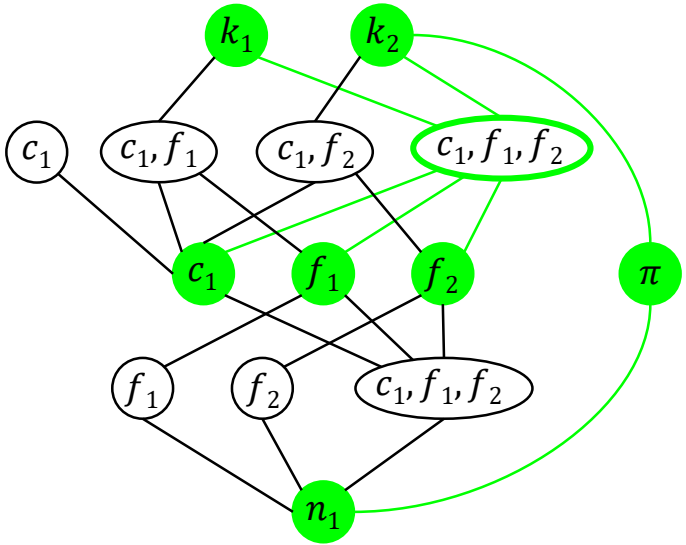
Not ACAP

Graph Labelling

- ACAP labelling as Node-Weighted Steiner Tree (NWST) problem
  - Solution of ACAP labelling, if exists, can be found by NWST on an augmented graph

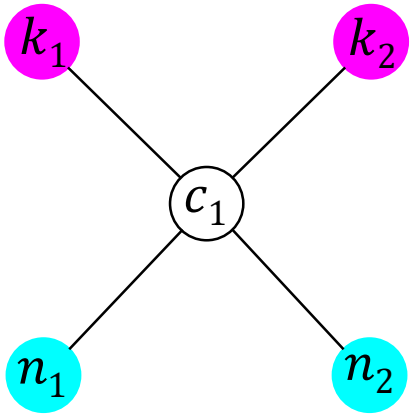


Augmented Graph (see paper for details)



Graph Labelling

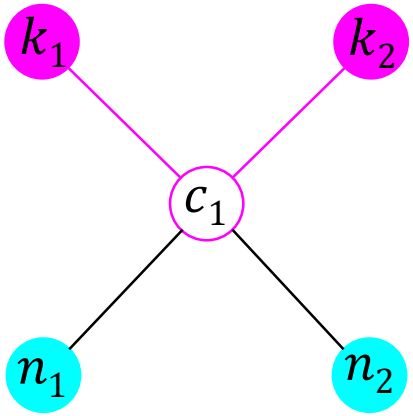
- ACAP labelling may not have a solution
  - An example is a double-articulation node



Graph Labelling

- ACAP labelling may not have a solution
  - An example is a double-articulation node

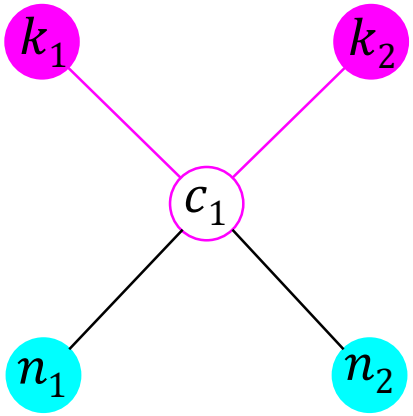
Cut c_1 is critical to connect the reachable kernel nodes



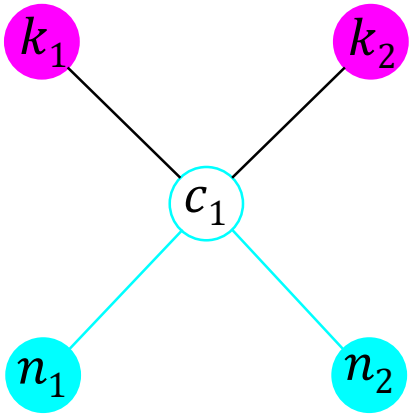
Graph Labelling

- ACAP labelling may not have a solution
  - An example is a double-articulation node

Cut c_1 is critical to connect the reachable kernel nodes



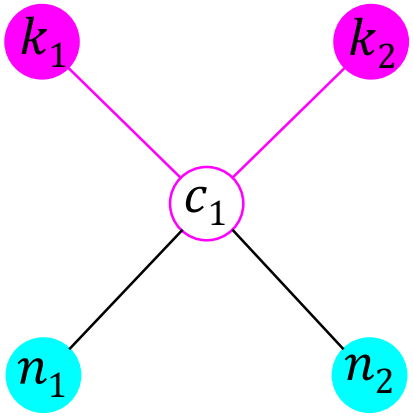
c_1 is also critical to connect the reachable neighborhood nodes



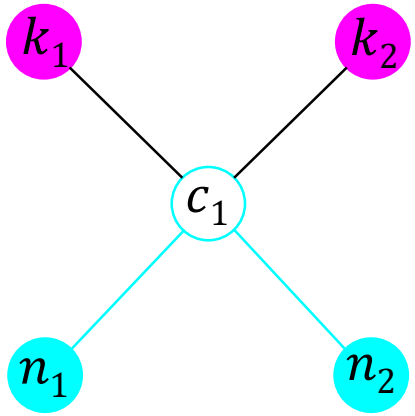
Graph Labelling

- ACAP labelling may not have a solution
  - An example is a double-articulation node
  - NWST would return a conflicting label of a cut/fill node

Cut c_1 is critical to connect the reachable kernel nodes

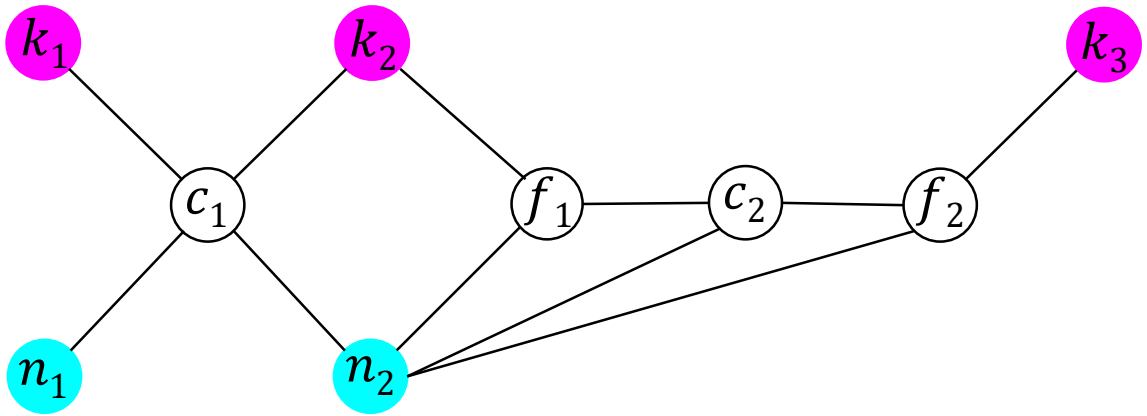


c_1 is also critical to connect the reachable neighborhood nodes



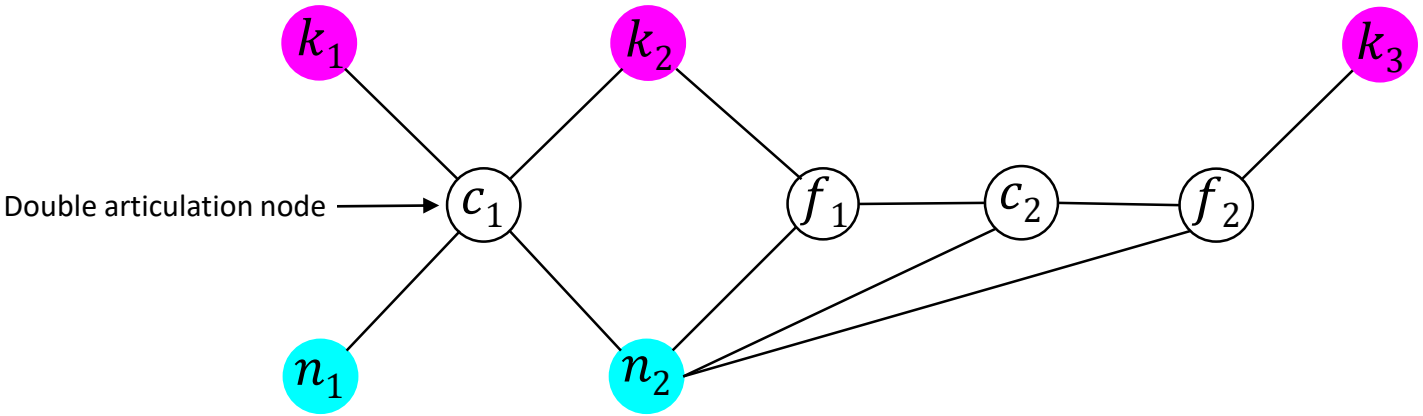
Graph Labelling

- Iterative labelling



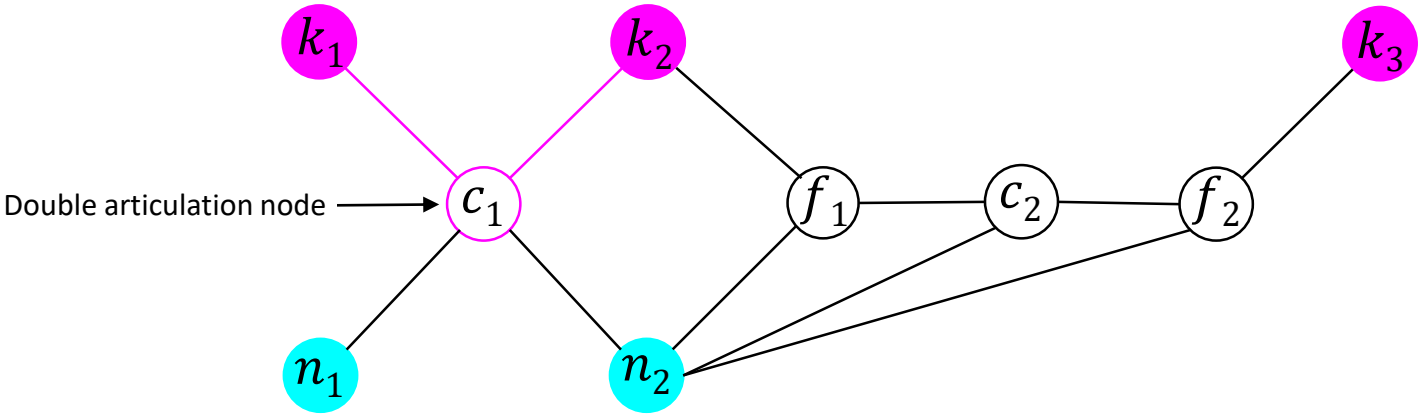
Graph Labelling

- Iterative labelling
  - Greedily label double-articulation nodes



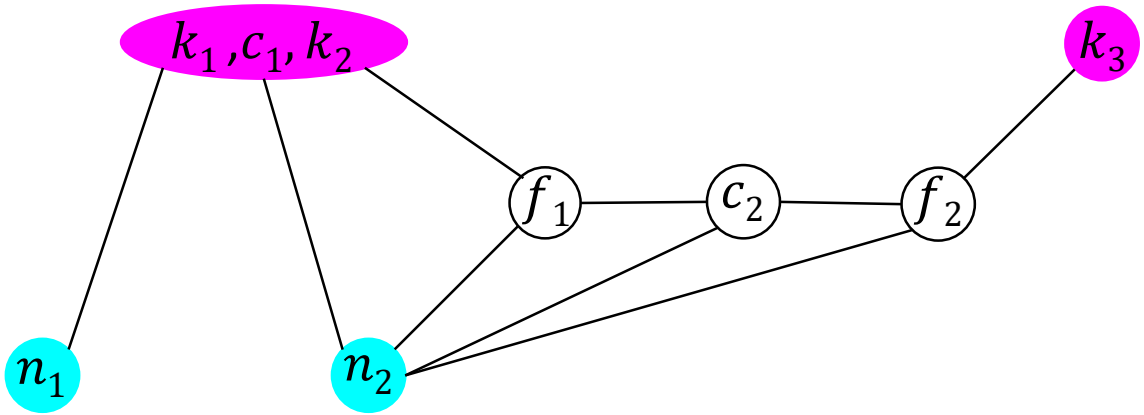
Graph Labelling

- Iterative labelling
  - Greedily label double-articulation nodes



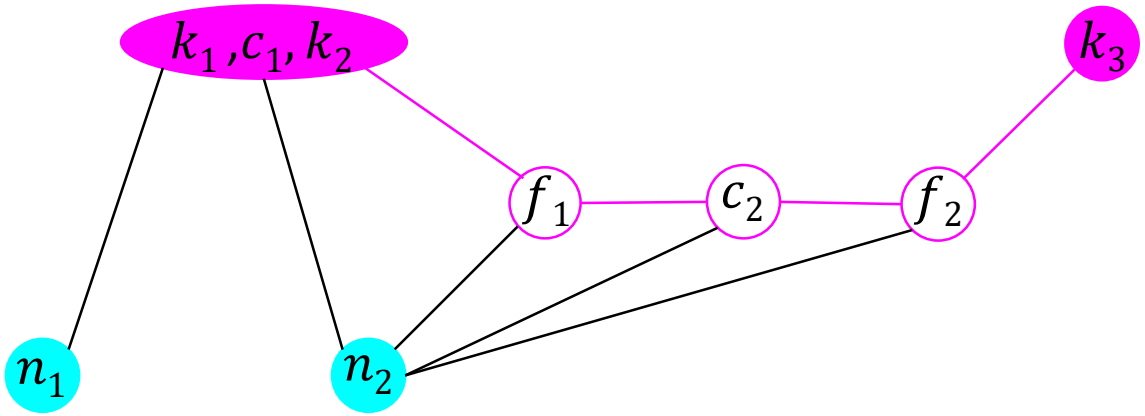
Graph Labelling

- Iterative labelling
  - Greedily label double-articulation nodes
  - Solve ACAP labelling (via NWST)



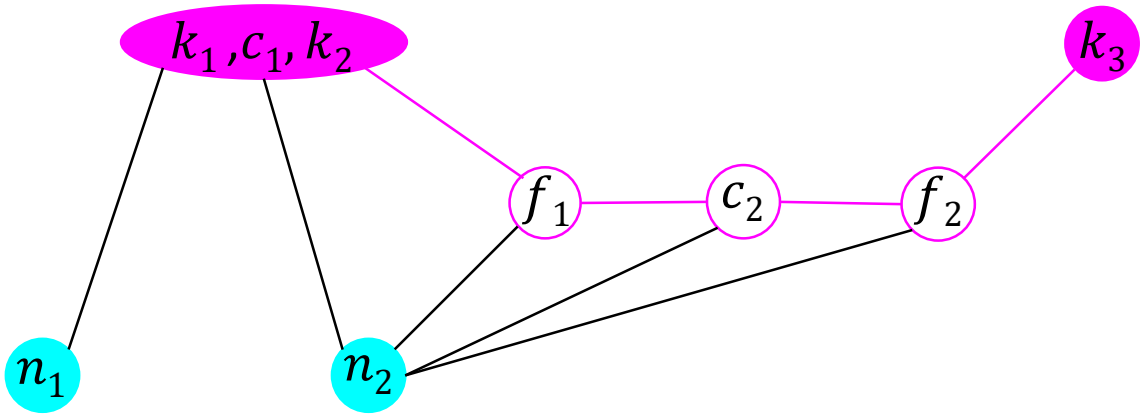
Graph Labelling

- Iterative labelling
  - Greedily label double-articulation nodes
  - Solve ACAP labelling (via NWST)



Graph Labelling

- Iterative labelling
  - Greedily label double-articulation nodes
  - Solve ACAP labelling (via NWST)
  - Return the solution if no node has conflicting labels; otherwise greedily label such node, and repeat

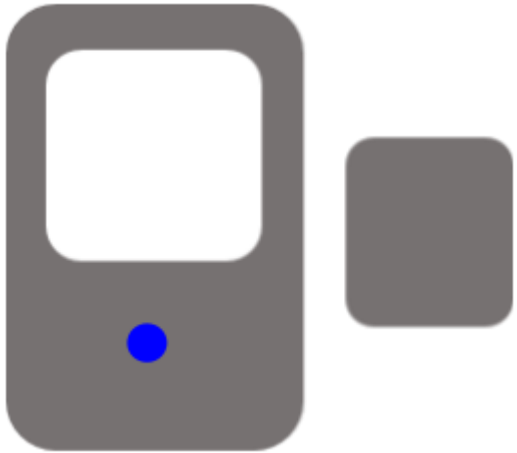


Results

- Input shapes represented as voxels
- Cuts and fills computed by two different (monotonic) methods:

Results

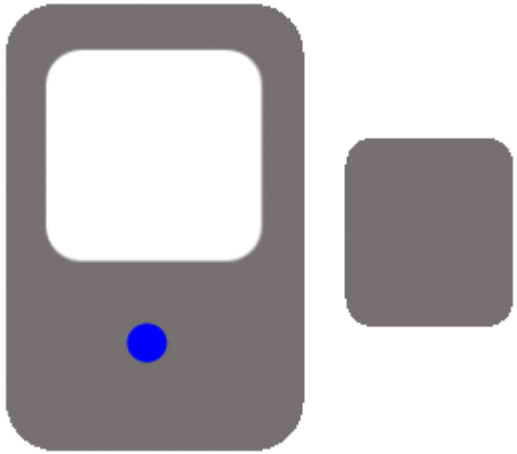
- Input shapes represented as voxels
- Cuts and fills computed by two different (monotonic) methods:



Inflation/deflation

Results

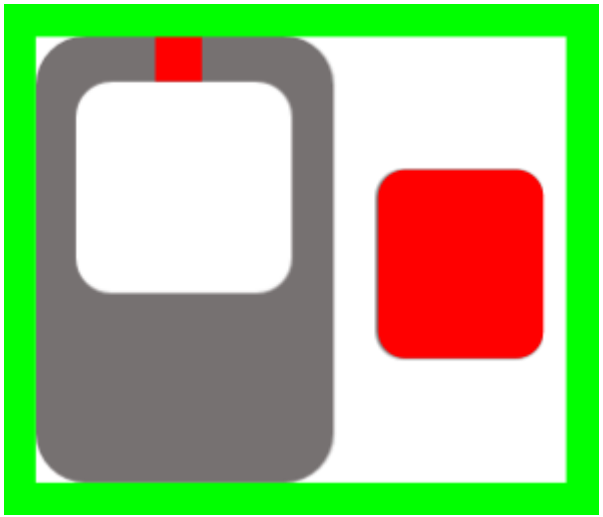
- Input shapes represented as voxels
- Cuts and fills computed by two different (monotonic) methods:



Inflation/deflation

Results

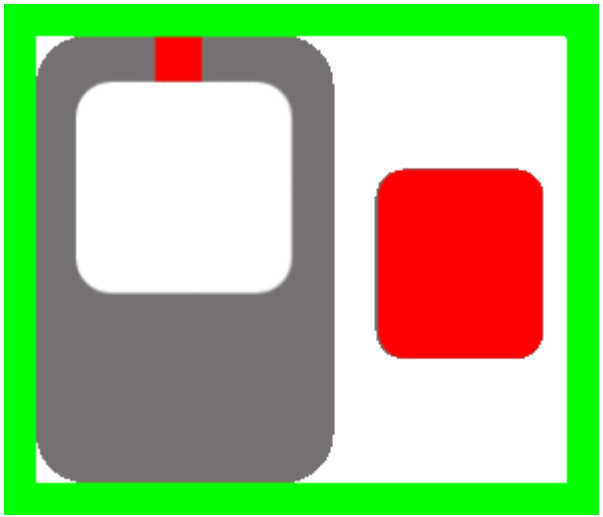
- Input shapes represented as voxels
- Cuts and fills computed by two different (monotonic) methods:



Inflation/deflation

Results

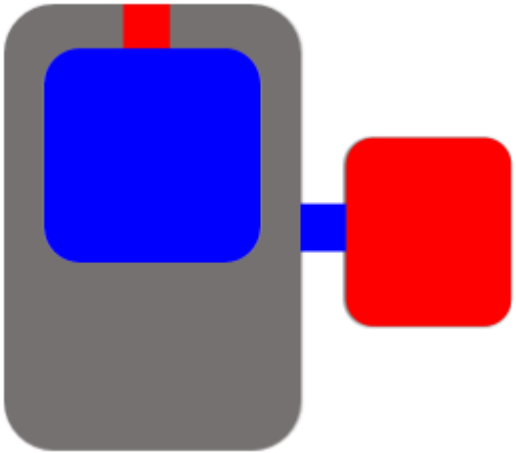
- Input shapes represented as voxels
- Cuts and fills computed by two different (monotonic) methods:



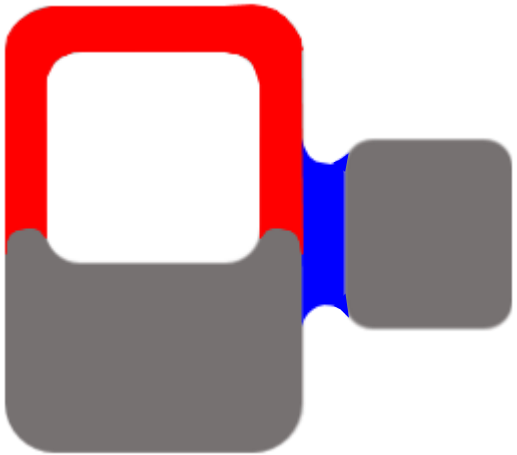
Inflation/deflation

Results

- Input shapes represented as voxels
- Cuts and fills computed by two different (monotonic) methods:



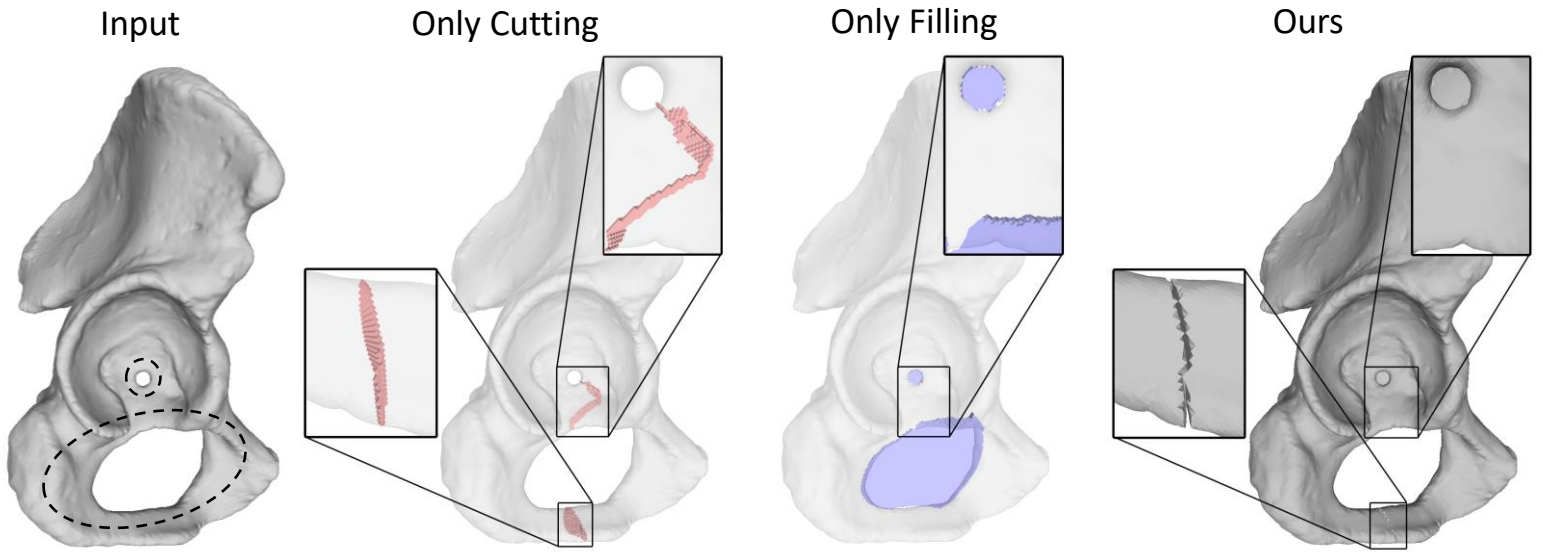
Inflation/deflation



Opening/closing

Results

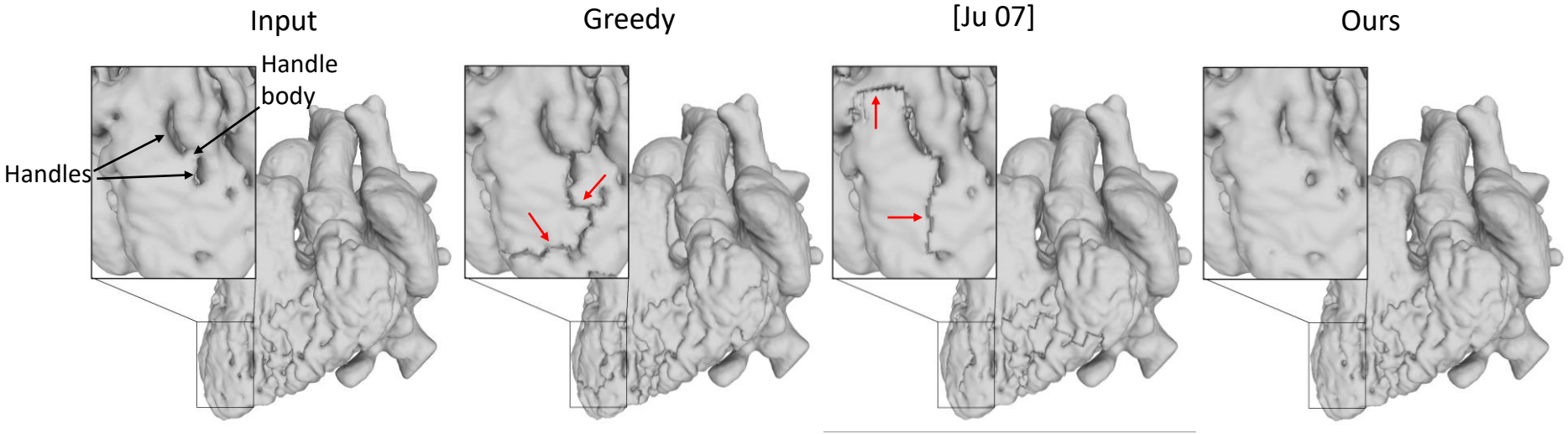
- Hip model (inflation/deflation cuts and fills)



	Input	Only Cutting	Only Filling	Ours
Components:	1	1	1	1
Handles:	2	0	0	0
Cavities:	0	0	0	0
Geometric cost:	0	756	6591	397

Results

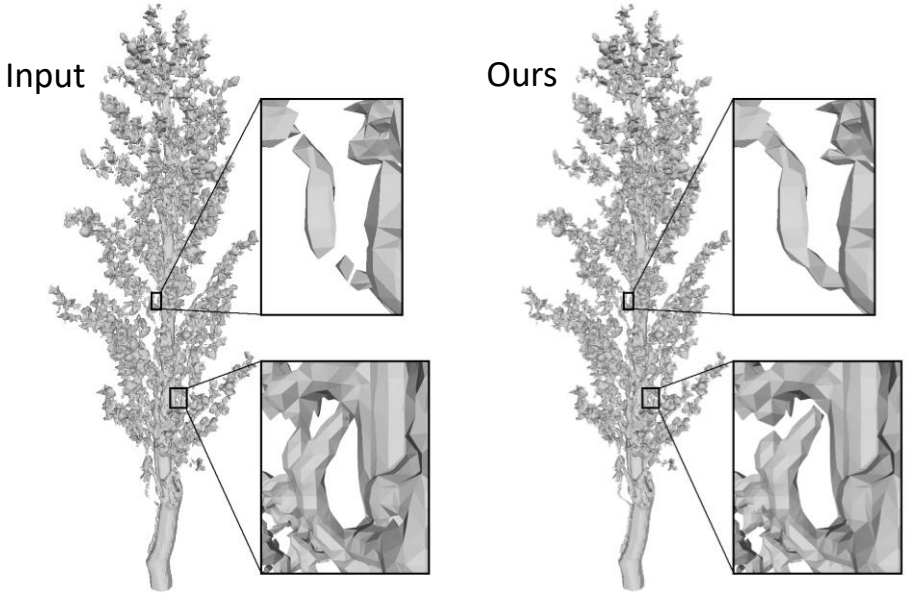
- Heart CT Scan (inflation/deflation cuts and fills)



	Input	Greedy	[Ju 07]	Ours
Components:	2	1	1	1
Handles:	215	4	0	0
Cavities:	13	0	0	0
Geometric cost:	0	2738	N/A	902

Results

- Sorghum Panicle CT Scan (inflation/deflation cuts and fills)



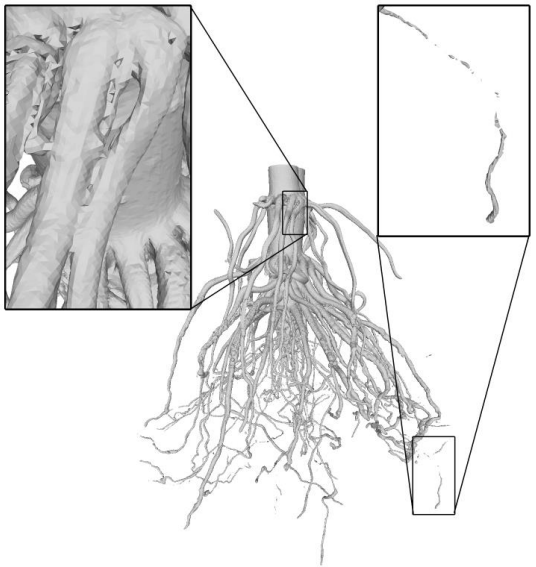
	Input	Ours	Lower bound
Components:	1749	22	20
Handles:	871	16	10
Cavities:	1	0	0

Timing
5.5 s

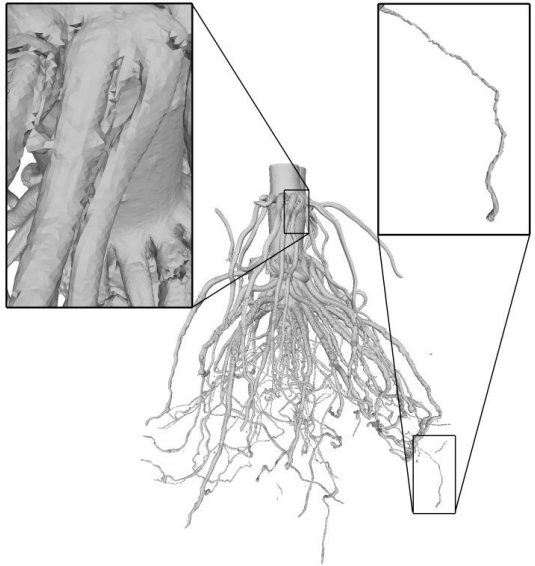
Results

- Corn root CT Scan (inflation/deflation cuts and fills)

Input



Ours



Timing
35.1 s

	Input	Ours	Lower bound
Components:	1958	32	32
Handles:	2117	3	0
Cavities:	158	0	0

Conclusion

- Summary: a global optimization algorithm which attempts to maximally simplify topology, while minimizing geometric change

Conclusion

- Summary: a global optimization algorithm which attempts to maximally simplify topology, while minimizing geometric change
- Future work:
  - Explore more effective and efficient optimization methods
  - Compute better cuts and fills
  - Allow user control over the target topology

Conclusion

- Summary: a global optimization algorithm which attempts to maximally simplify topology, while minimizing geometric change
- Future work:
  - Explore more effective and efficient optimization methods
  - Compute better cuts and fills
  - Allow user control over the target topology
- Website: https://danzeng8.github.io/topo-simplifier/
  - Paper, code, examples, discussion