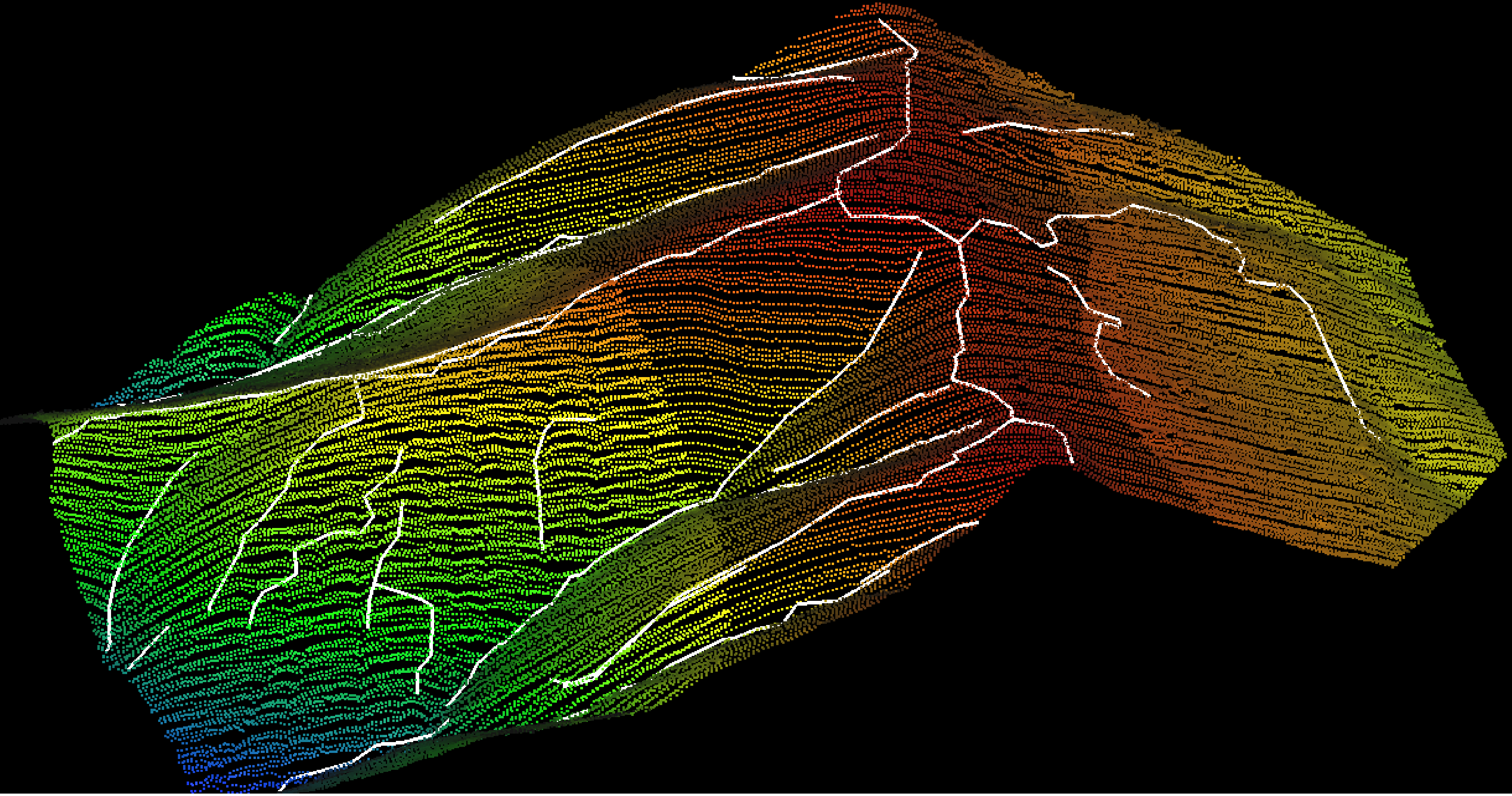


3D breakline extraction from point clouds with the Medial Axis Transform

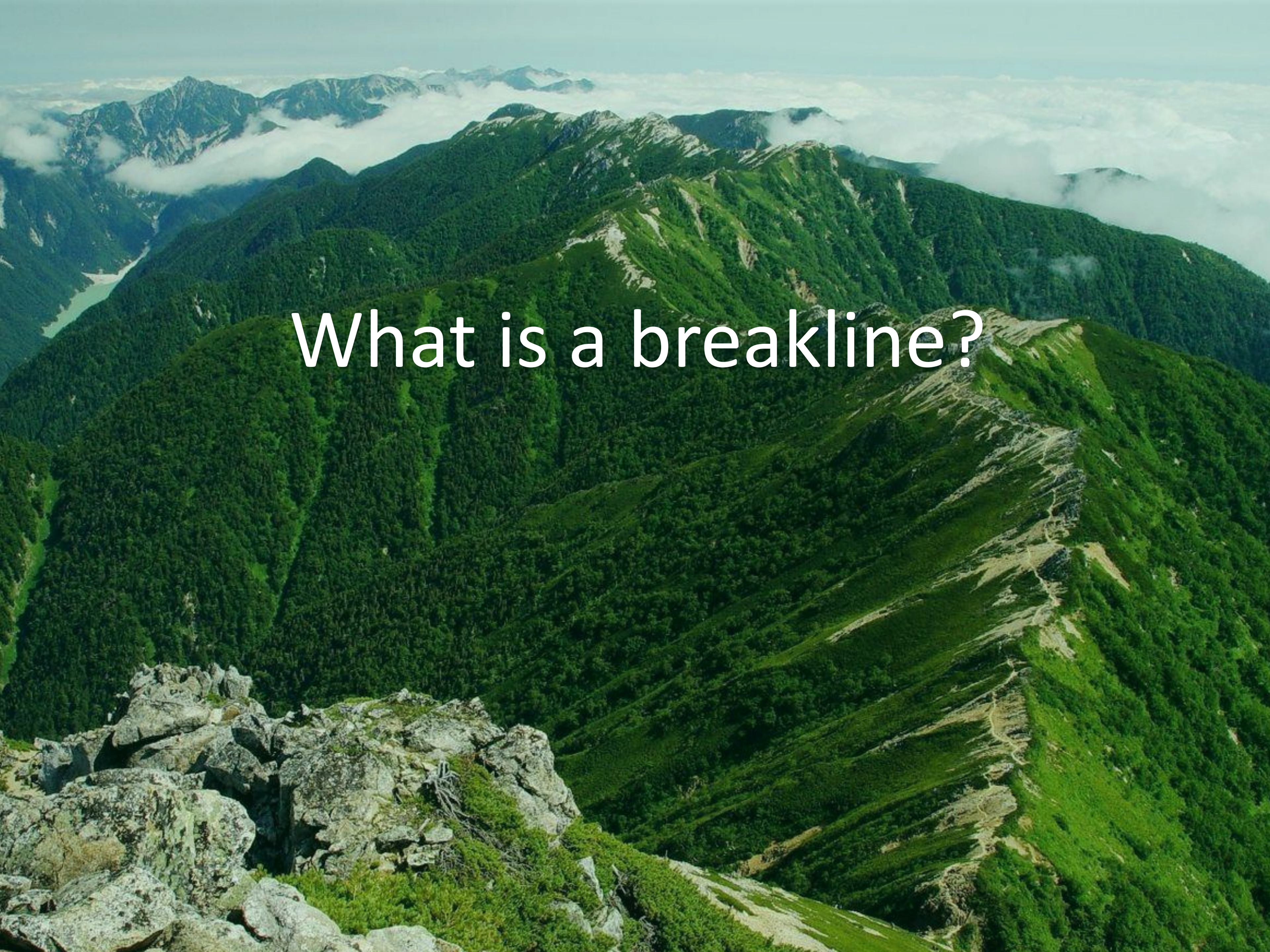
Qu Wang

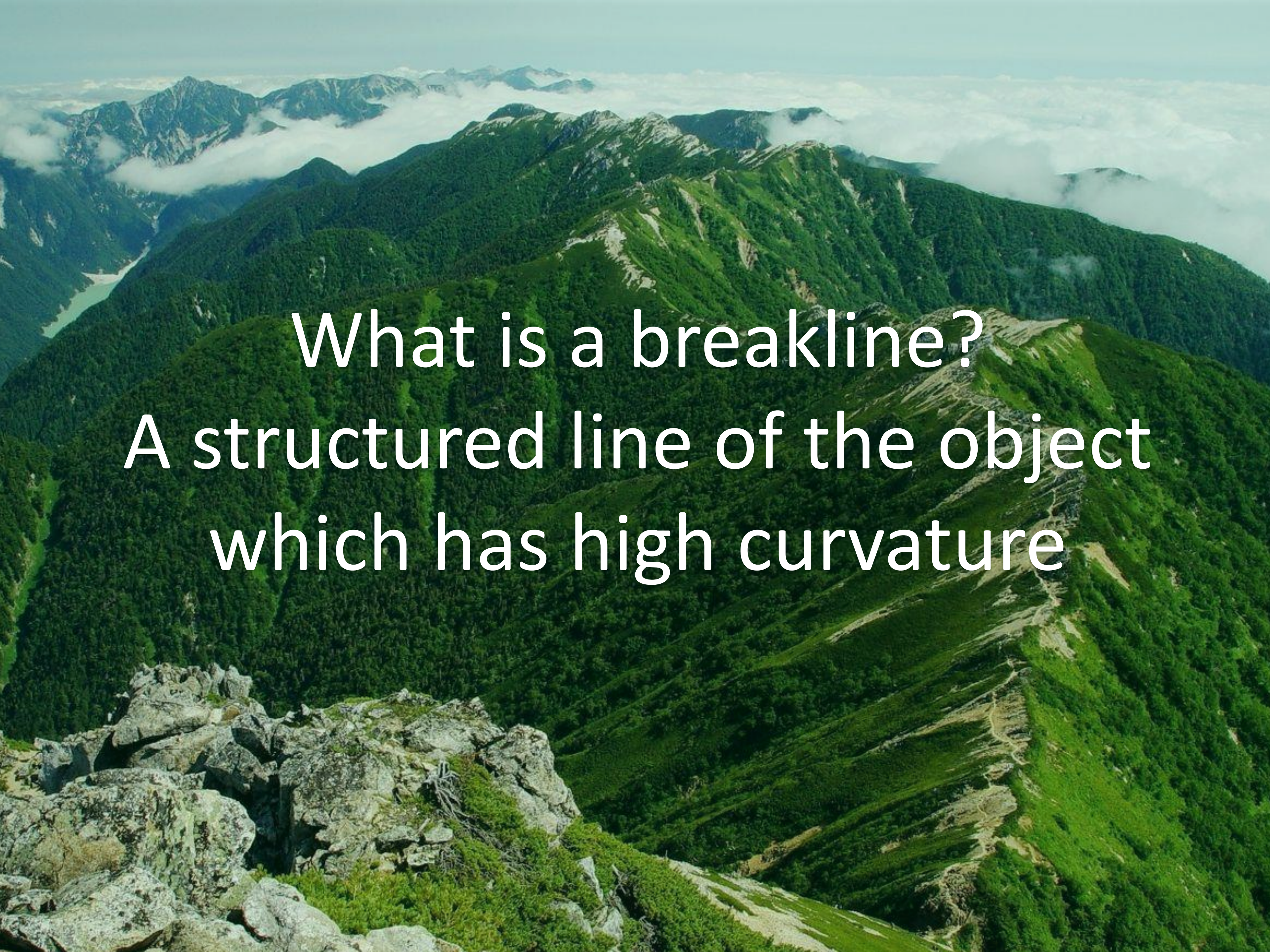
Mentor #1: Ravi Peters
Mentor #2: Hugo Ledoux

Chairperson: Arie Bergsma
Co-reader: Martijn Meijers

TU Delft

What is a breakline?

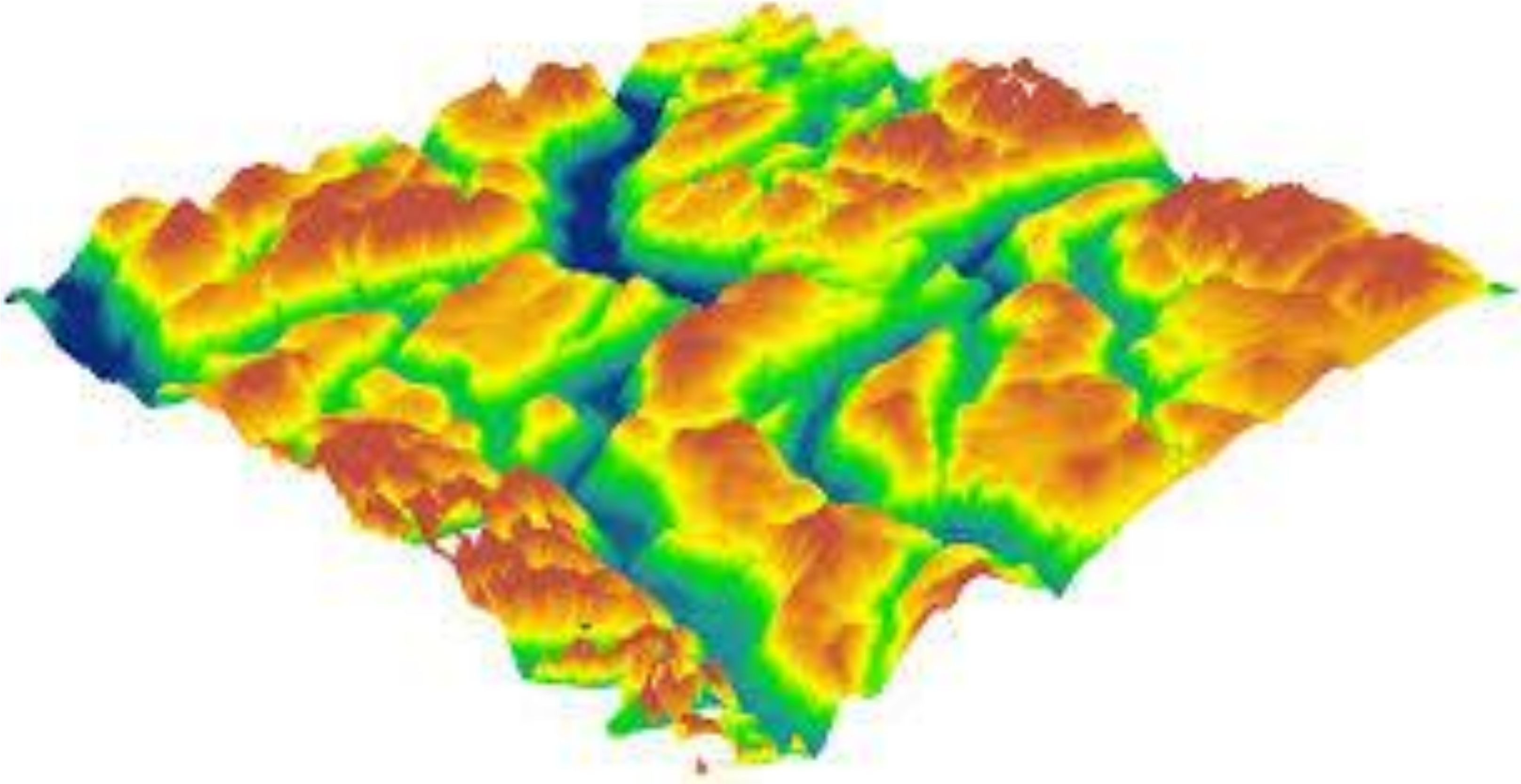




What is a breakline?
A structured line of the object
which has high curvature

Motivation

- Application
  - Topographic (e.g. generate DEM)



Motivation

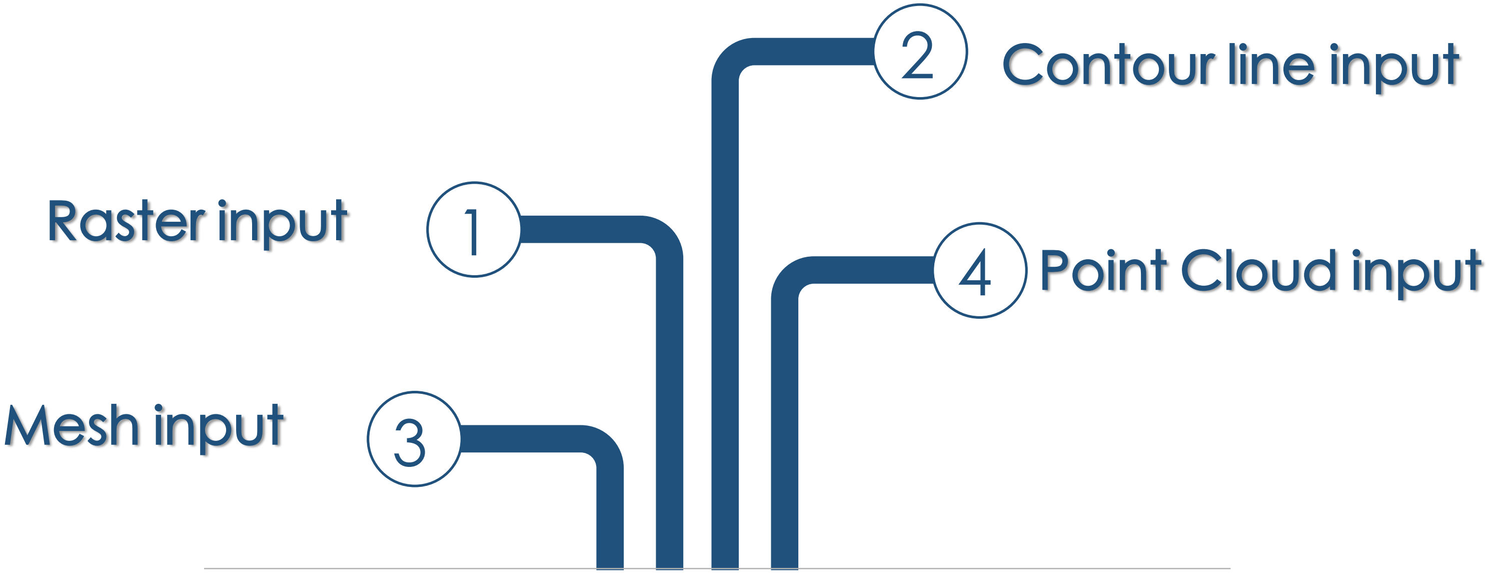
- Application
  - Hydrological (e.g. flood simulation)

Photo: Nabin Baral, ICIMOD
Dhading, Nawalparasi, Nepal (2009)

Motivation

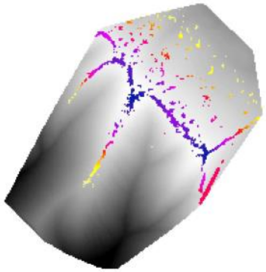
- Application
  - Monitoring

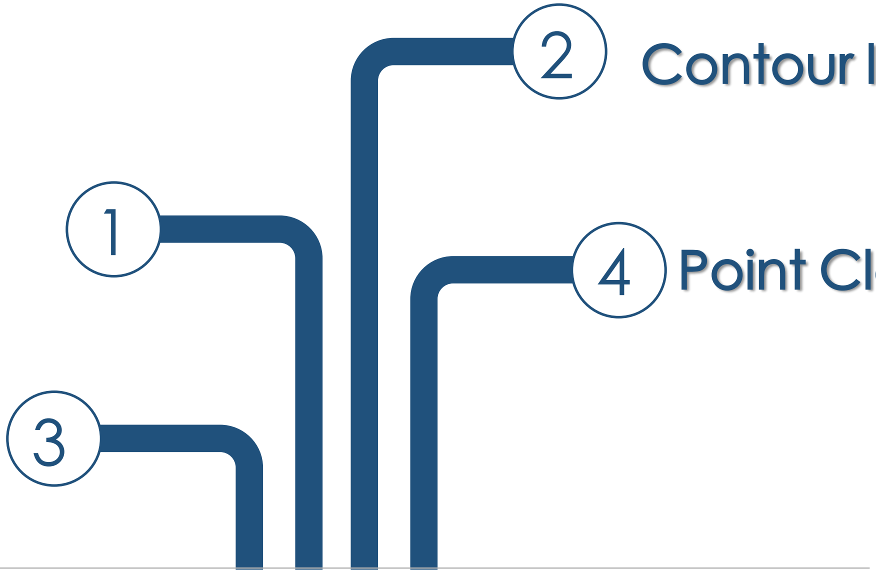
Existing method



Idea: detect points in breaklines;
make connection

Existing method





Raster input

1

2

Contour line input

4

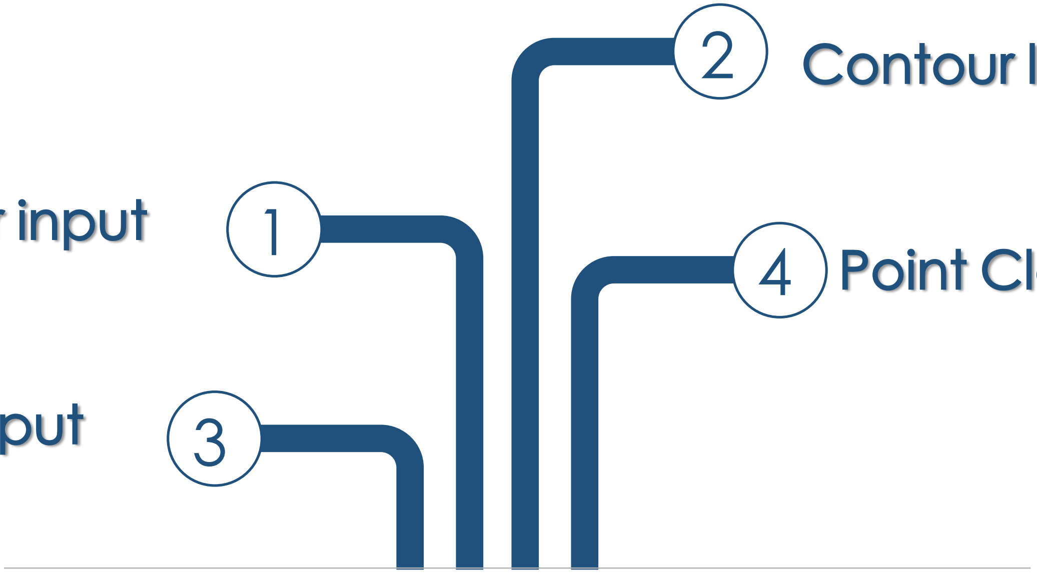
Point Cloud input

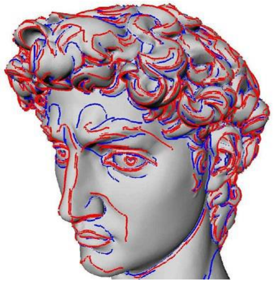
Mesh input

3

Idea: detect points in breaklines;
make connection

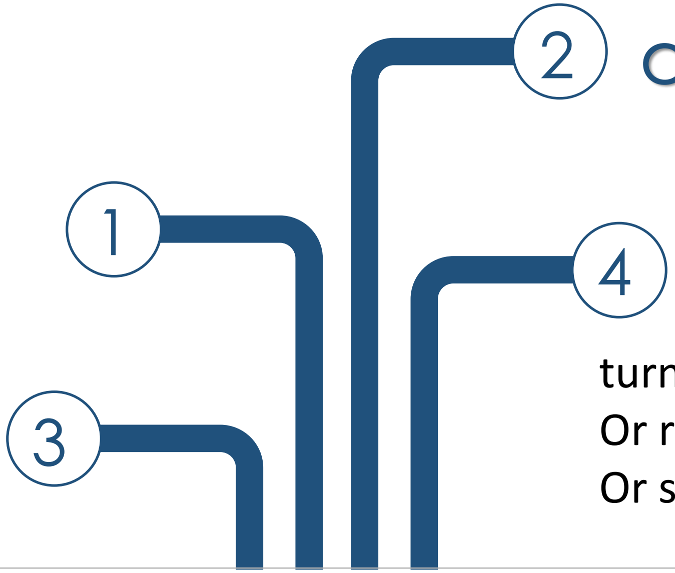
Existing method





Idea: detect points in breaklines;
make connection

Existing method



2 Contour line input

1 Raster input

4 Point Cloud input

turn into raster/contour line;
Or require 2D breakline input;
Or specific to one type of breakline

3 Mesh input

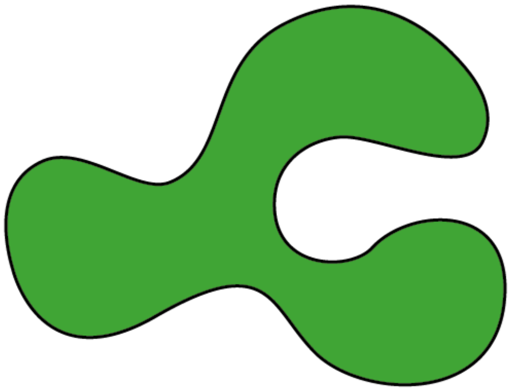
Idea: detect points in breaklines;
make connection

Motivation

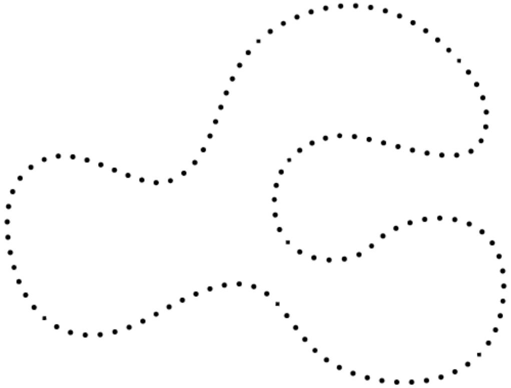
- No efficient method to generate breaklines from point clouds directly
- Converting point cloud into other formats will lose information

Related work: MAT

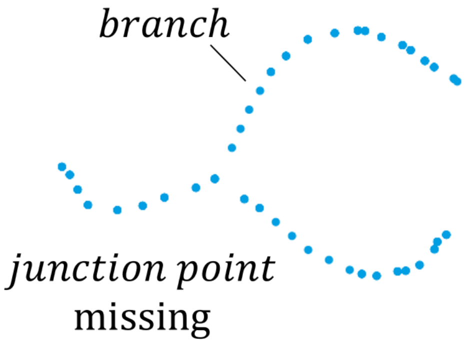
- Medial Axis Transform (MAT)
  - Represent the skeleton



2D object



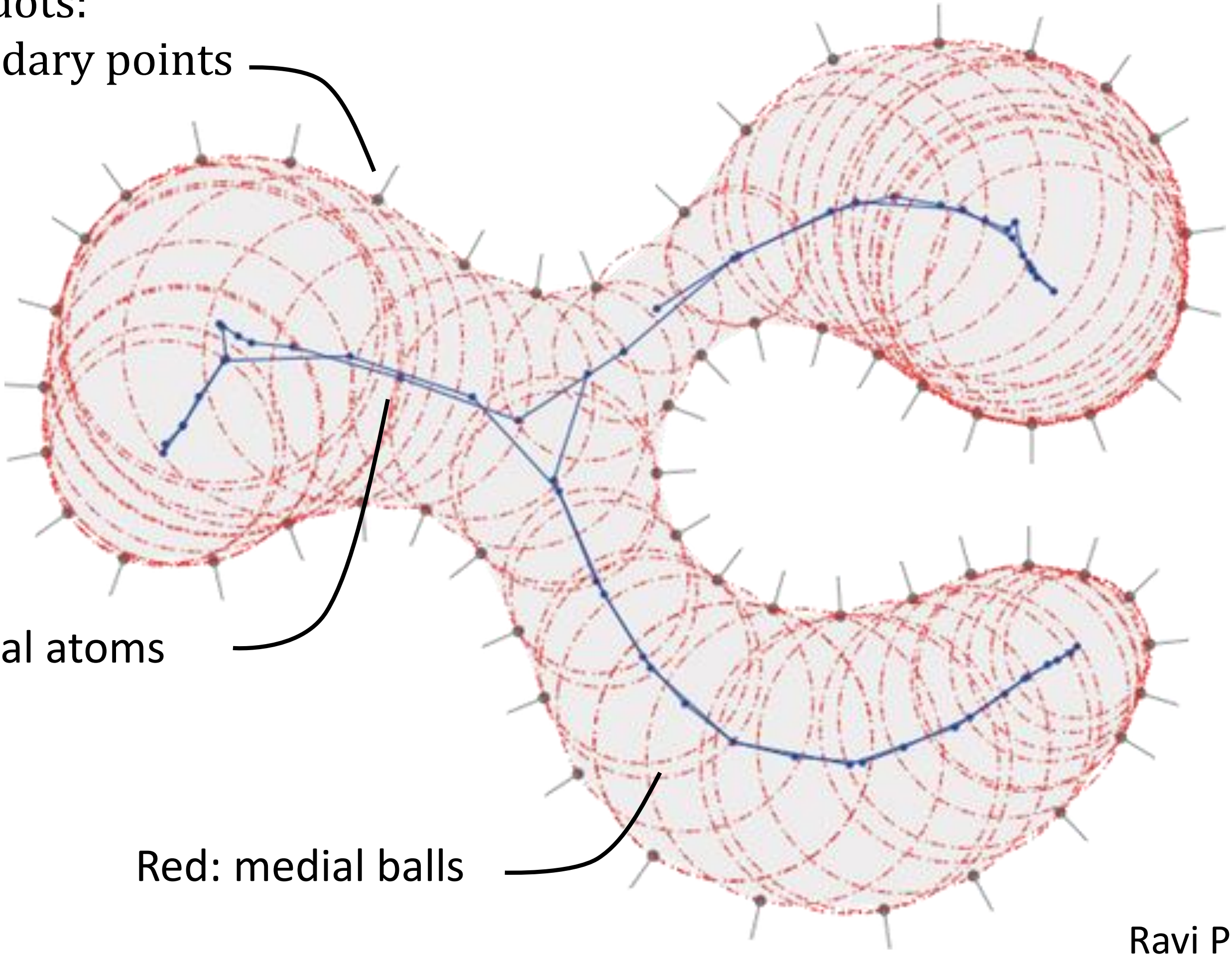
Boundary points



Interior MAT

Related work: MAT

- Definition: medial ball



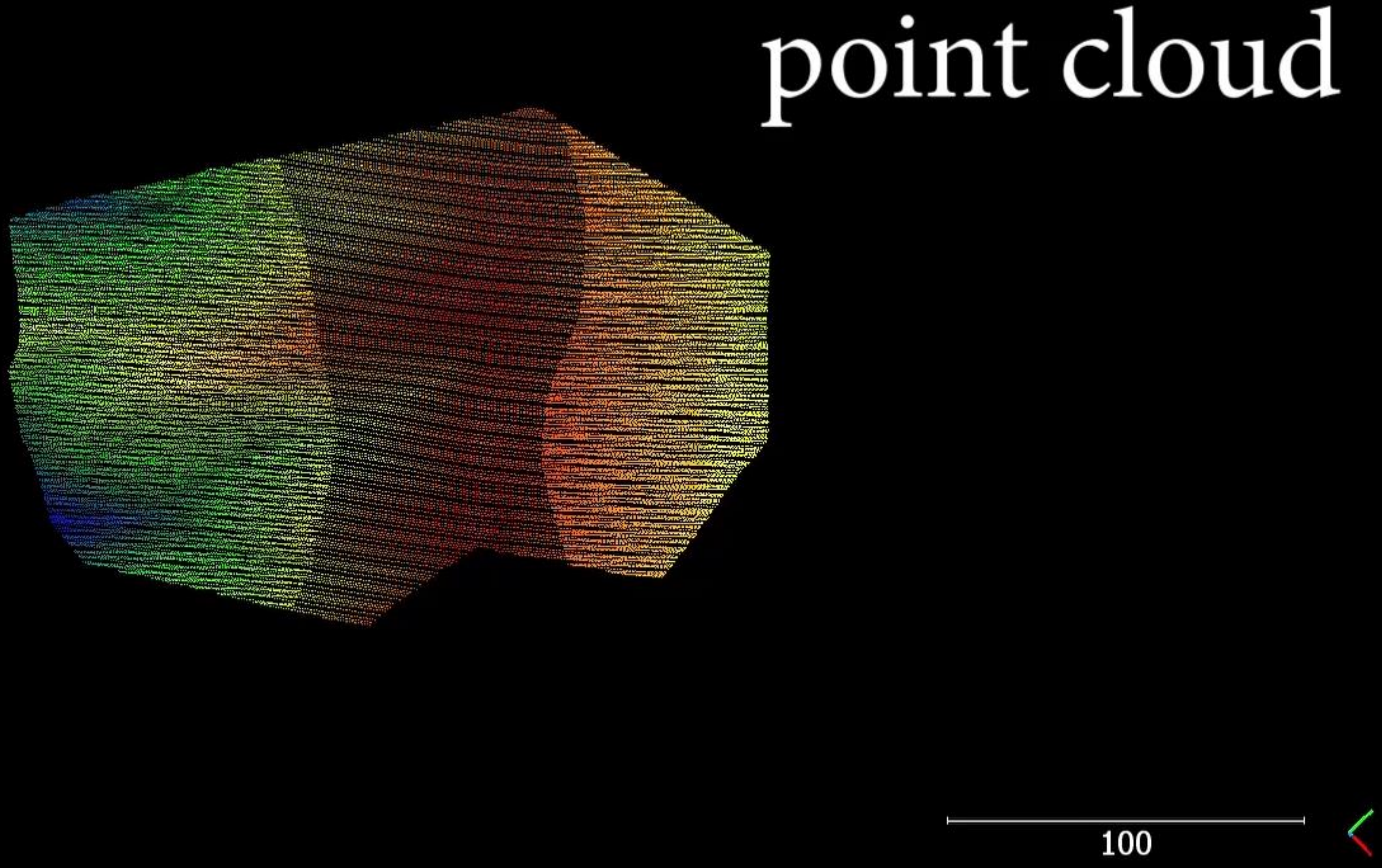
Black dots:
object's boundary points

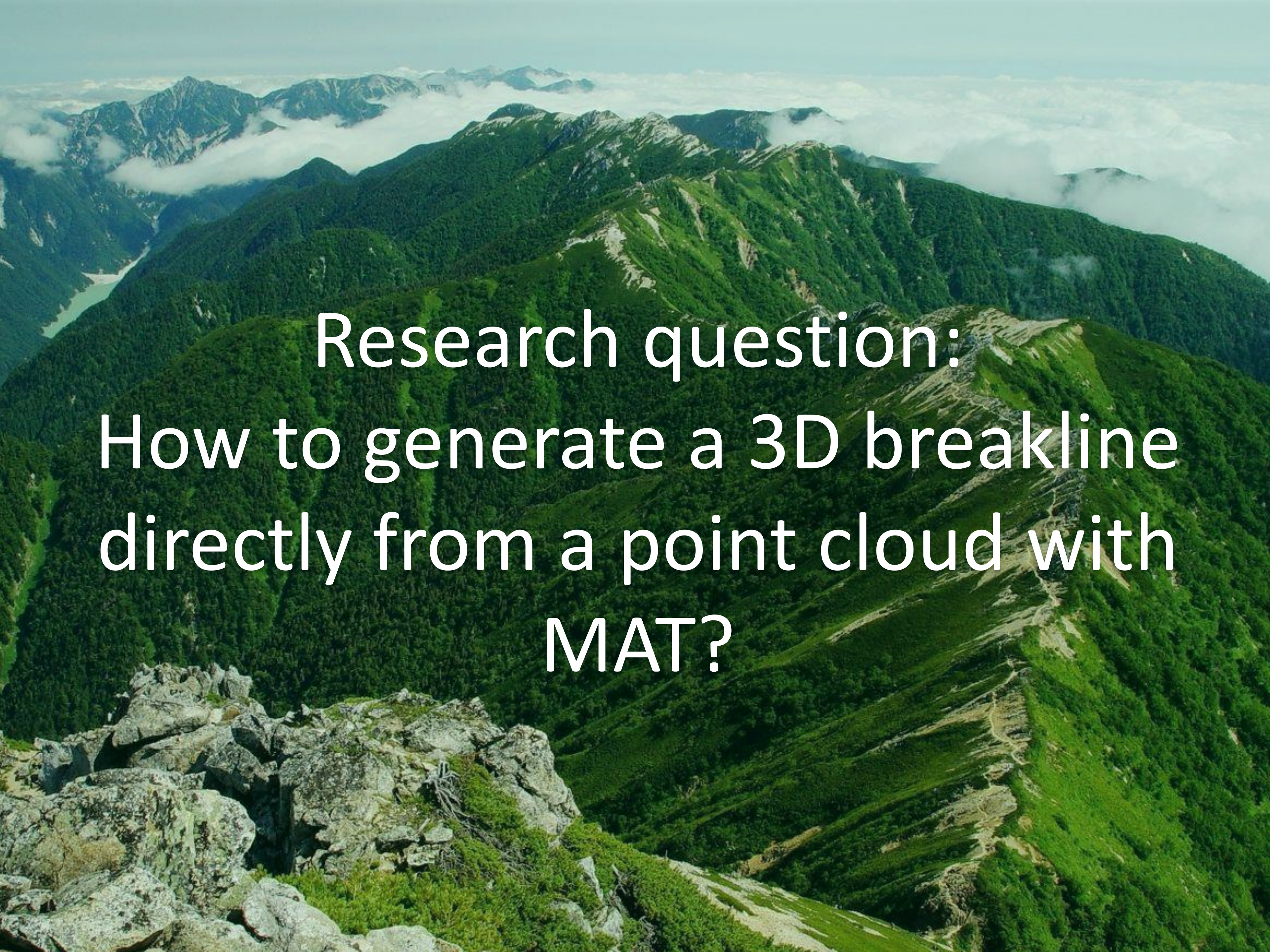
Blue dots: medial atoms

Red: medial balls

Ravi Peters, 2018

Related work: MAT





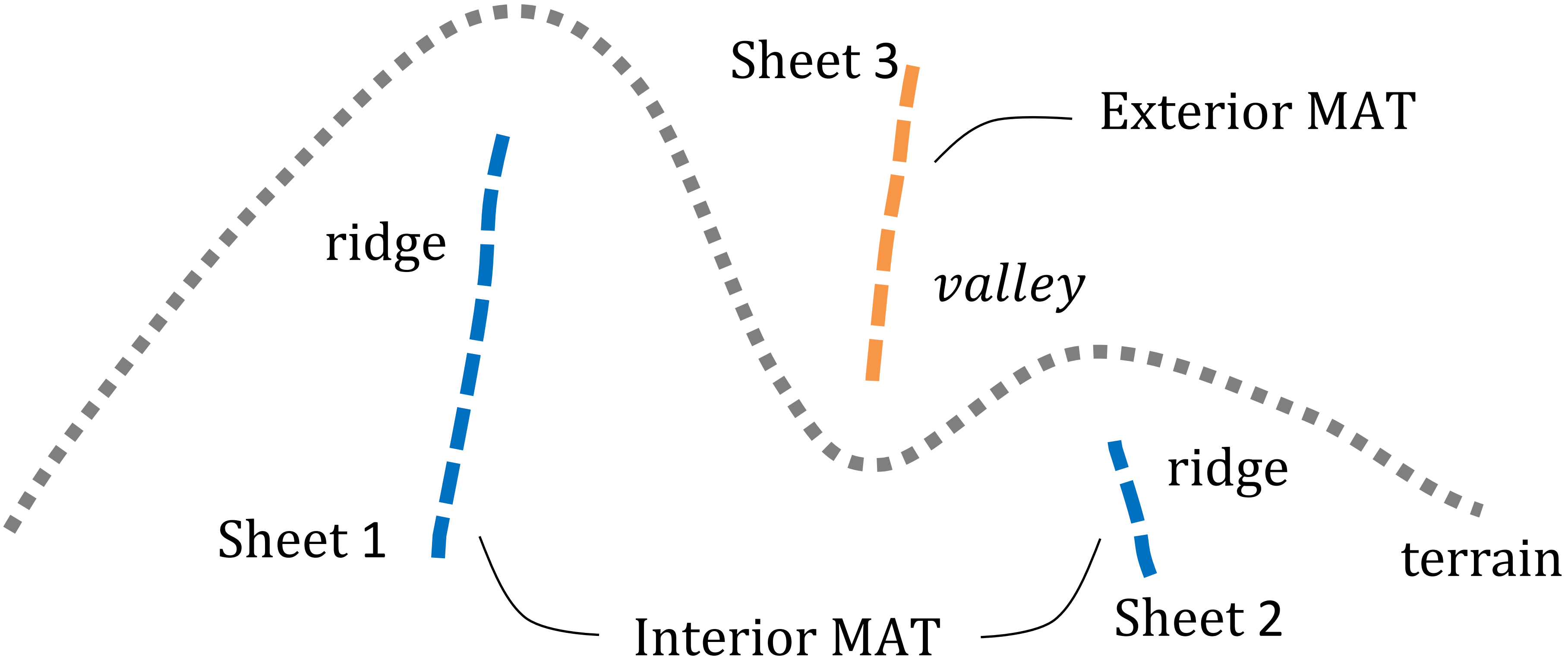
Research question:
How to generate a 3D breakline directly from a point cloud with MAT?

Scope

- Focusing on natural ground surfaces, such as ridges and valleys in mountains;
- Point cloud contains trees will not be considered;
- This project can not deal with holes caused by river or lake in the point cloud.

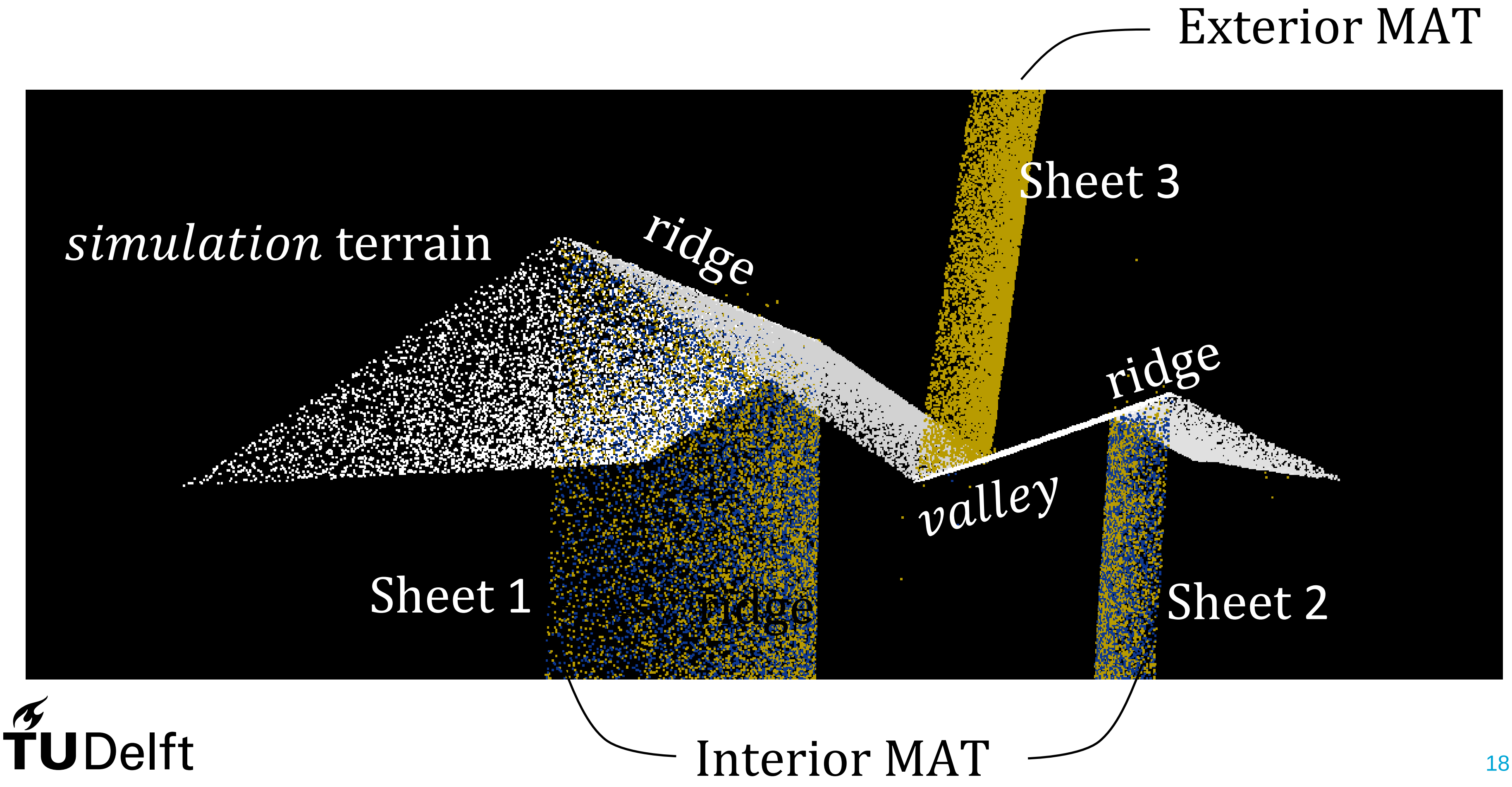
Methodology: MAT & breakline

- Link between breaklines and MAT



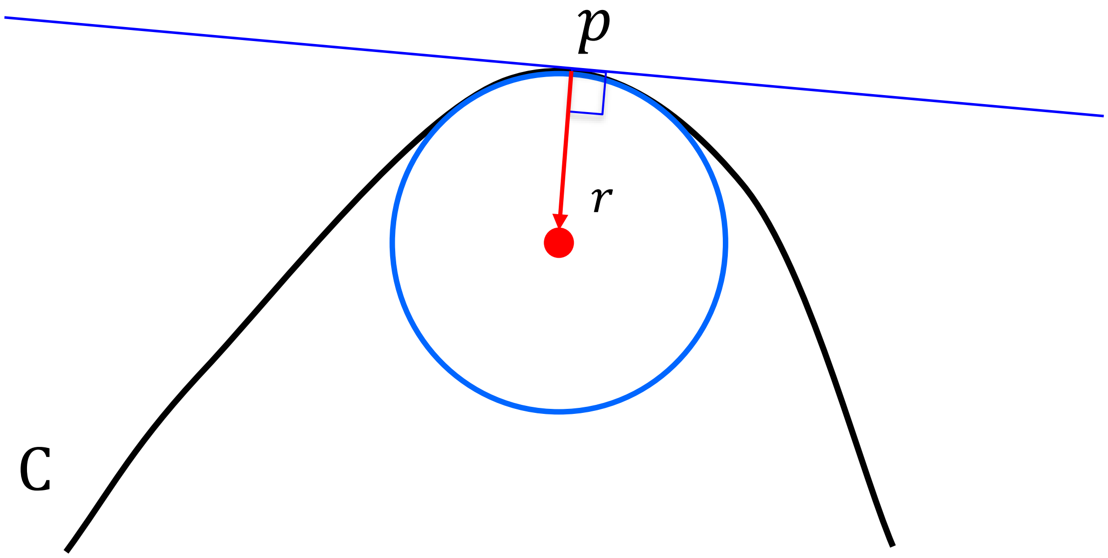
Methodology: MAT & breakline

- Link between breaklines and MAT



Methodology: MAT & breakline

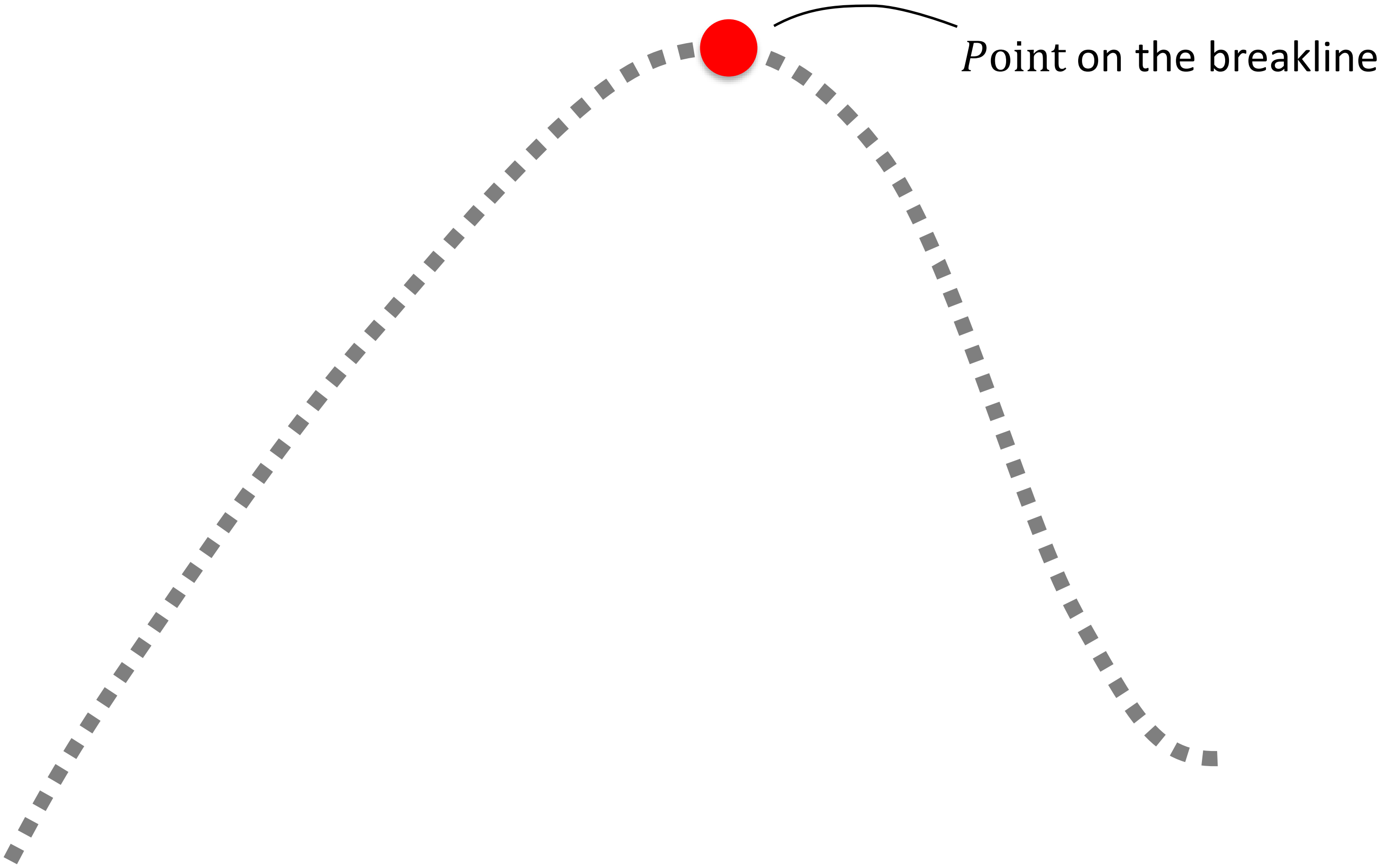
- Link between breaklines and MAT
  - Curvature and medial ball



$curvature = 1/r$

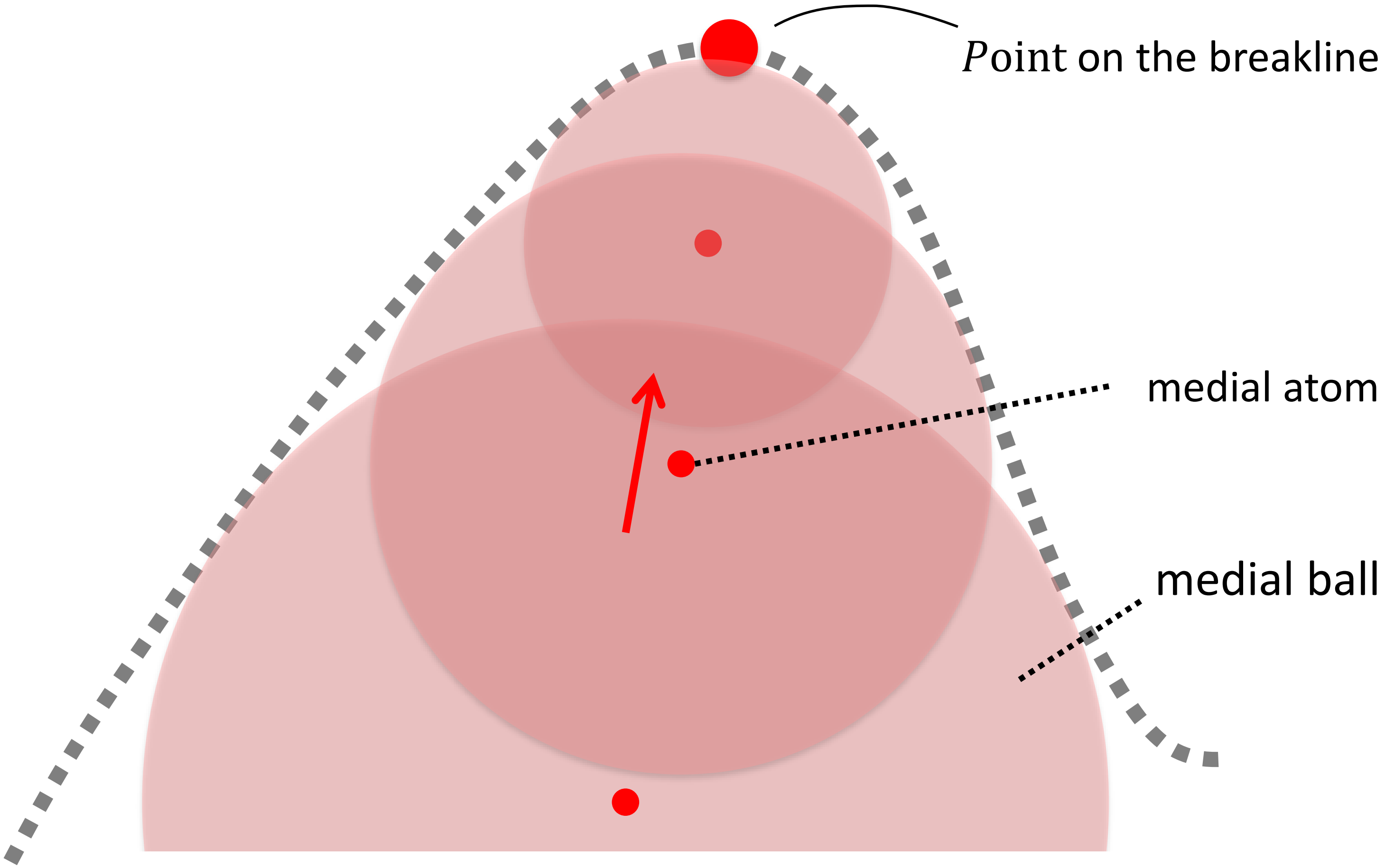
Methodology: MAT & breakline

- Link between breaklines and MAT

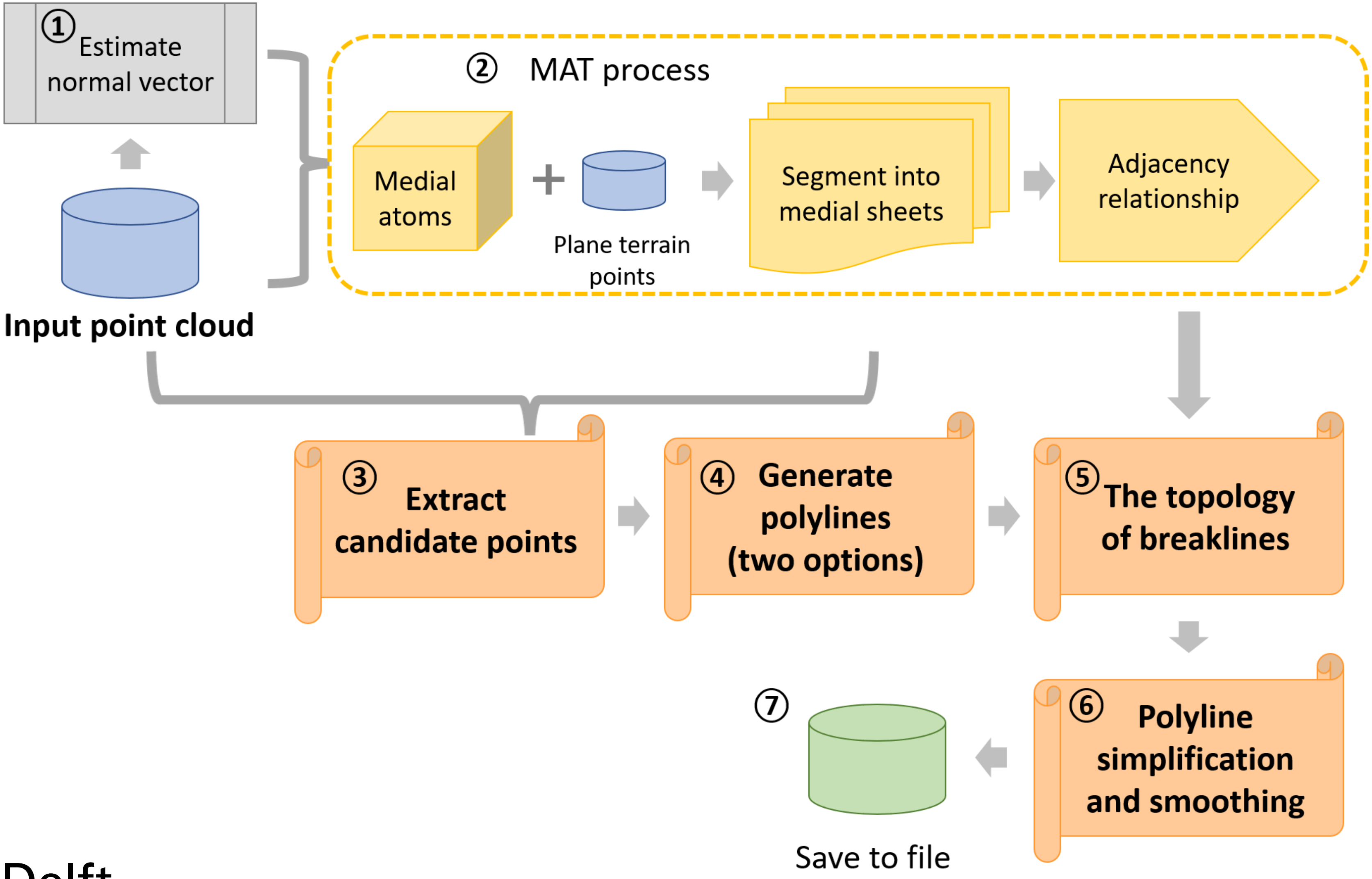


Methodology: MAT & breakline

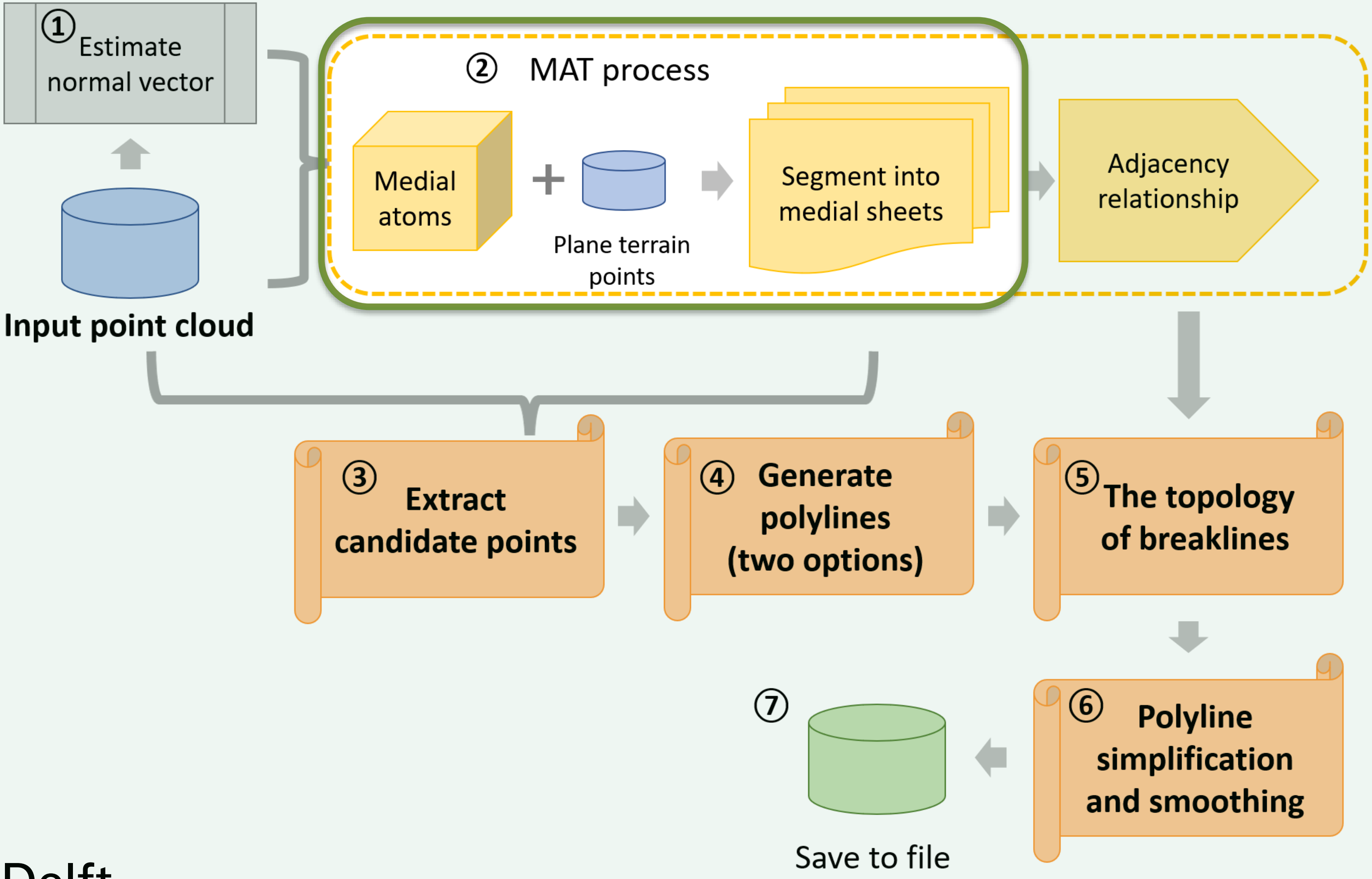
- Link between breaklines and MAT



Methodology overview


①Estimate normal vector
Input point cloud
② MAT process
Medial atoms
+
Plane terrain points
Segment into medial sheets
Adjacency relationship
③ Extract candidate points
④ Generate polylines (two options)
⑤ The topology of breaklines
⑥ Polyline simplification and smoothing
⑦
Save to file

Methodology overview



① Estimate normal vector

Input point cloud

② MAT process

Medial atoms

+

Plane terrain points

Segment into medial sheets

Adjacency relationship

③ Extract candidate points

④ Generate polylines (two options)

⑤ The topology of breaklines

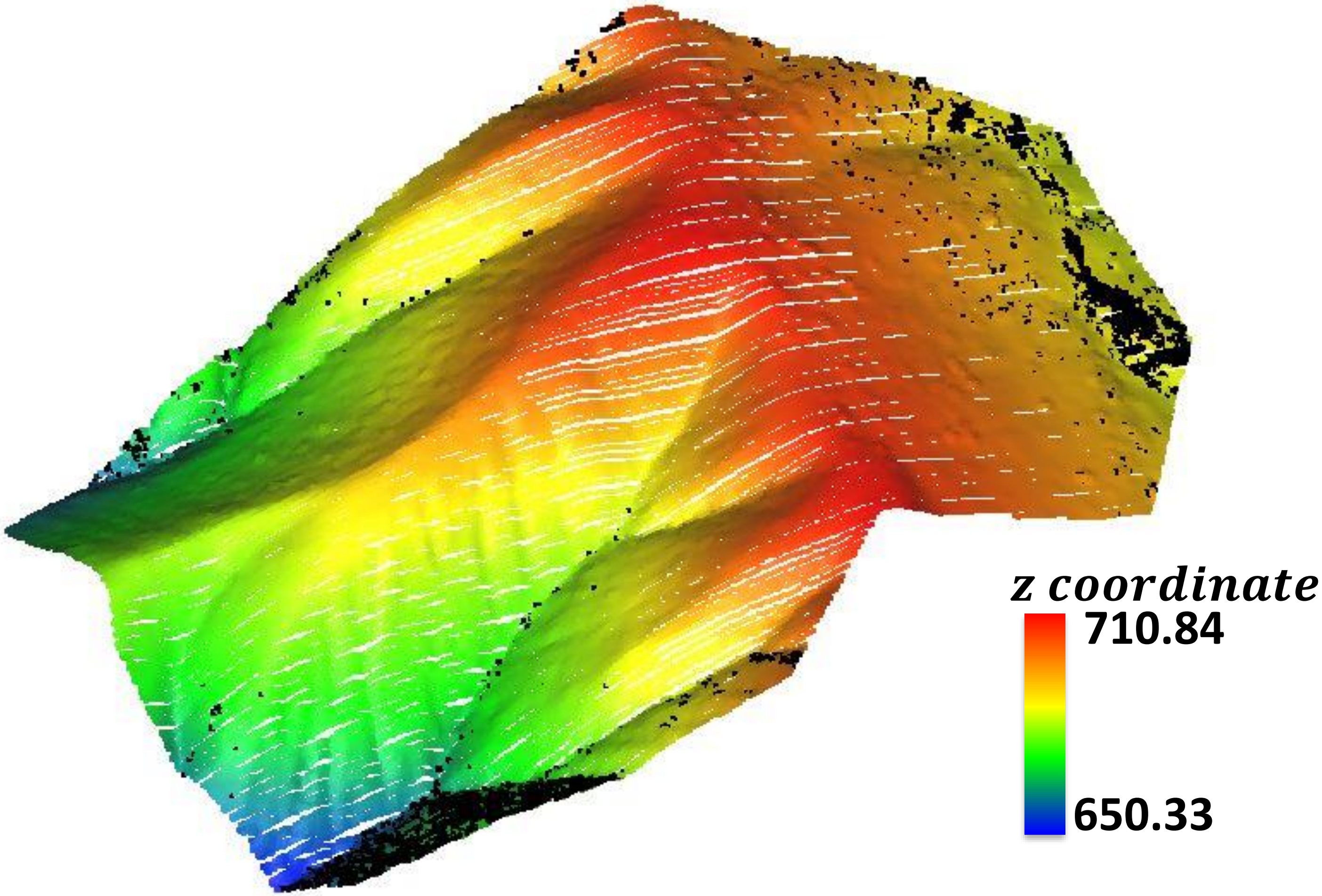
⑥ Polyline simplification and smoothing

⑦

Save to file

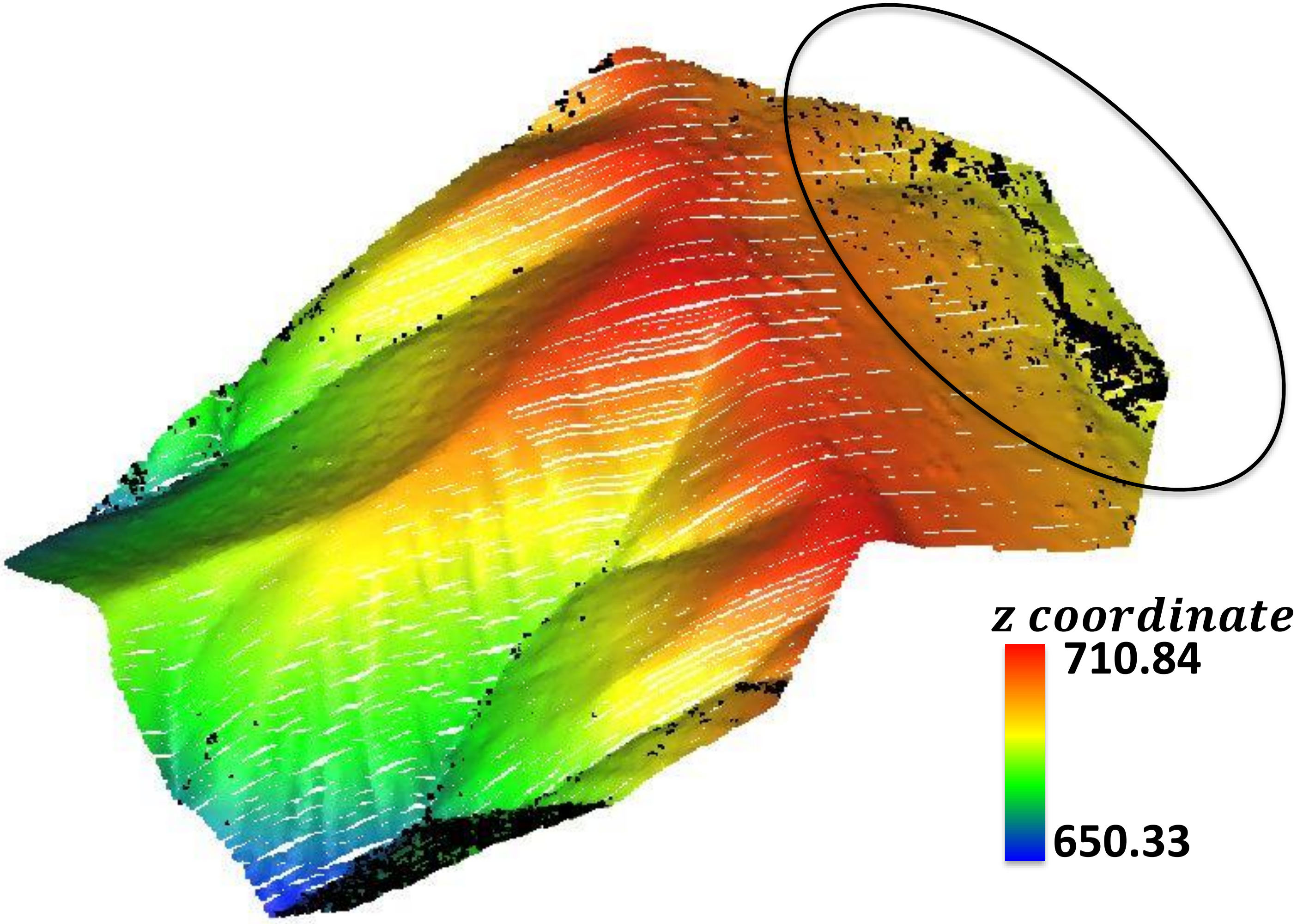
Methodology: MAT process

- Unshrinken points (black) → planar area



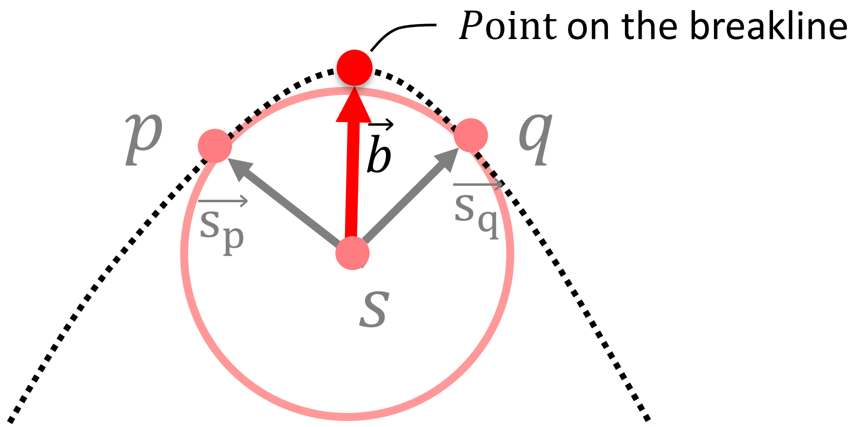
Methodology: MAT process

- Unshrinken points (black) → planar area



Methodology: MAT process

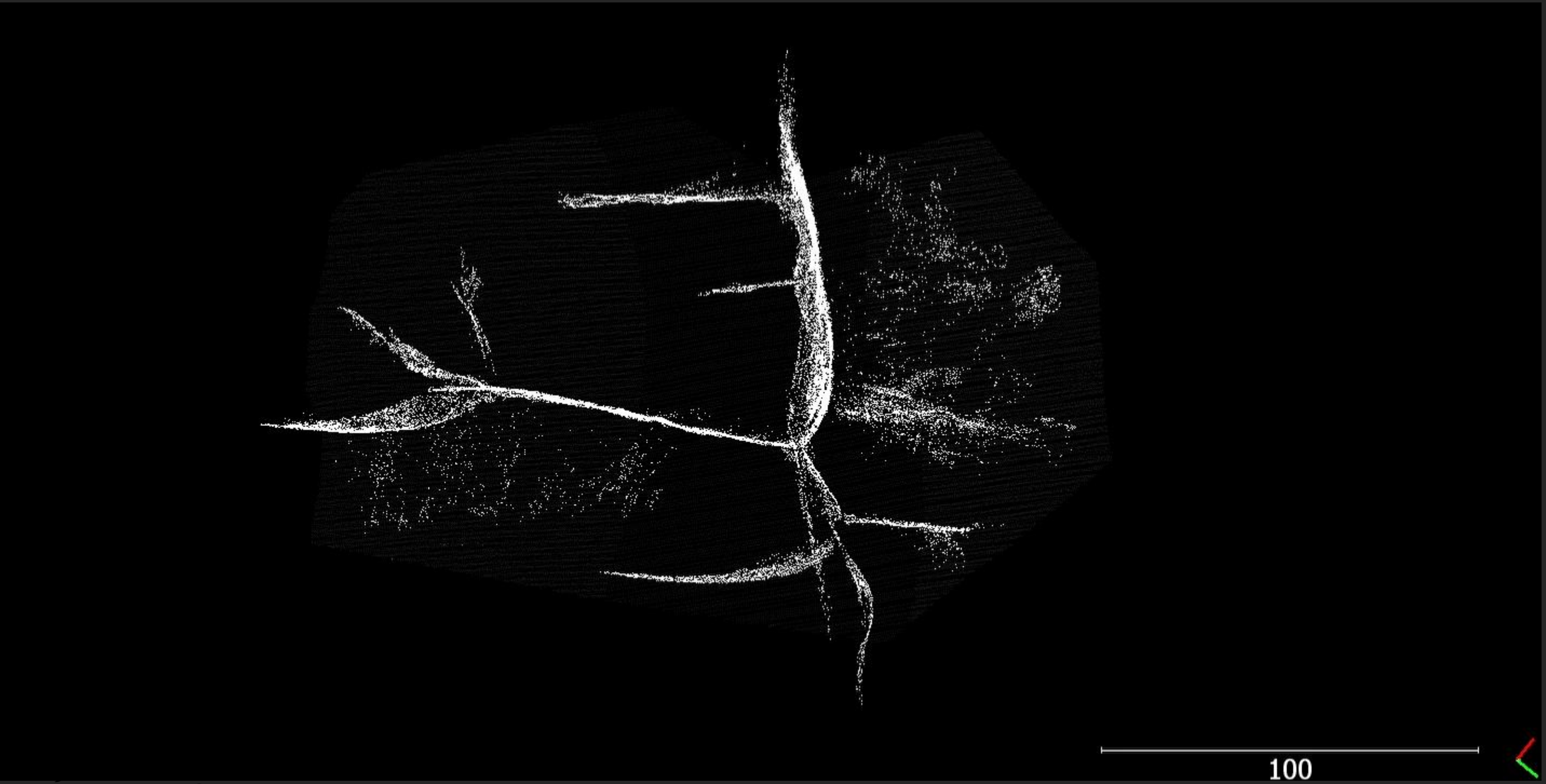
- Medial segmentation → medial sheet
- Medial geometry: bisector $\vec{b}$



Ravi Peters, 2018

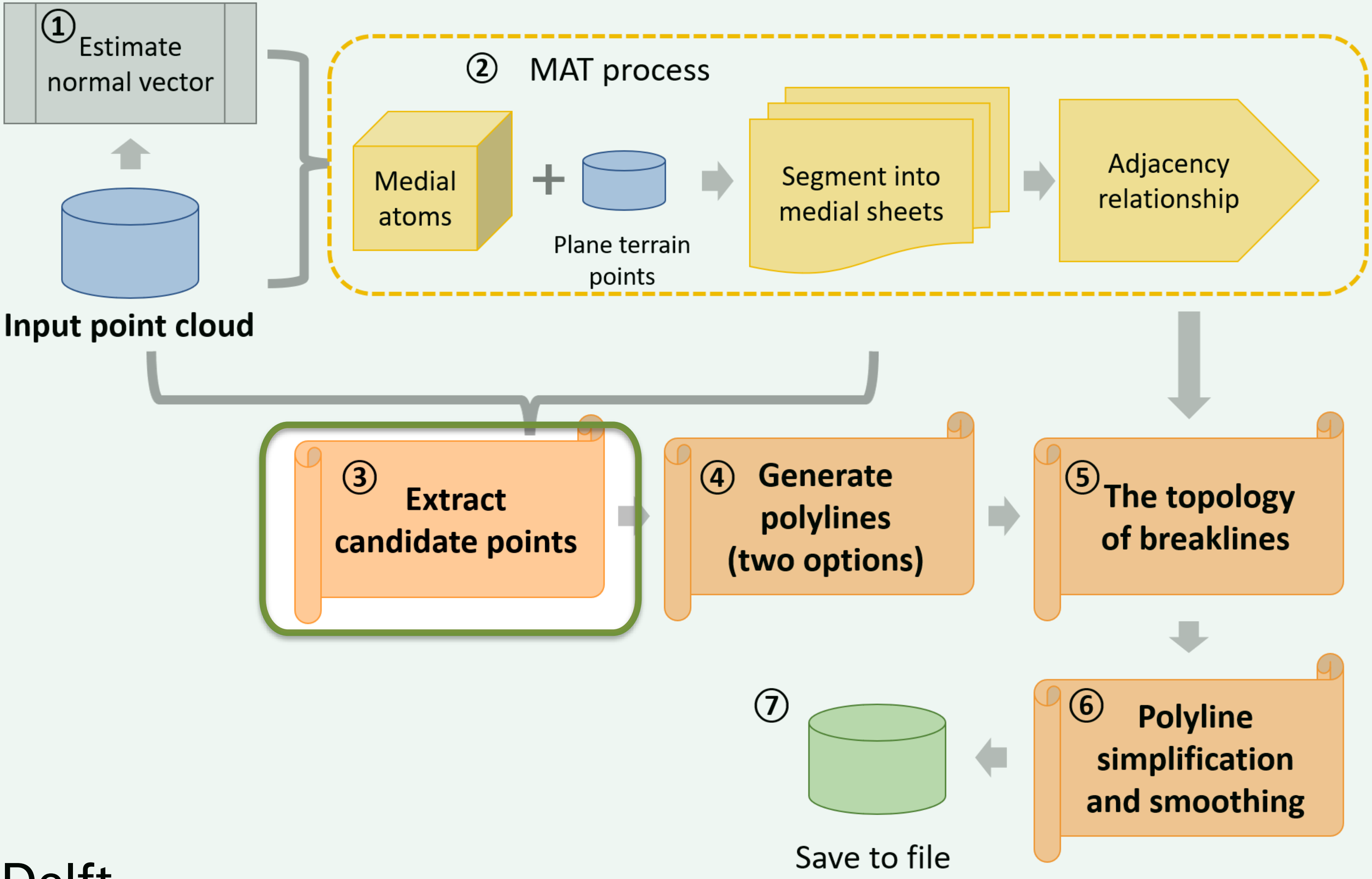
MAT segmentation

- Medial segmentation → medial sheets



TUDelft

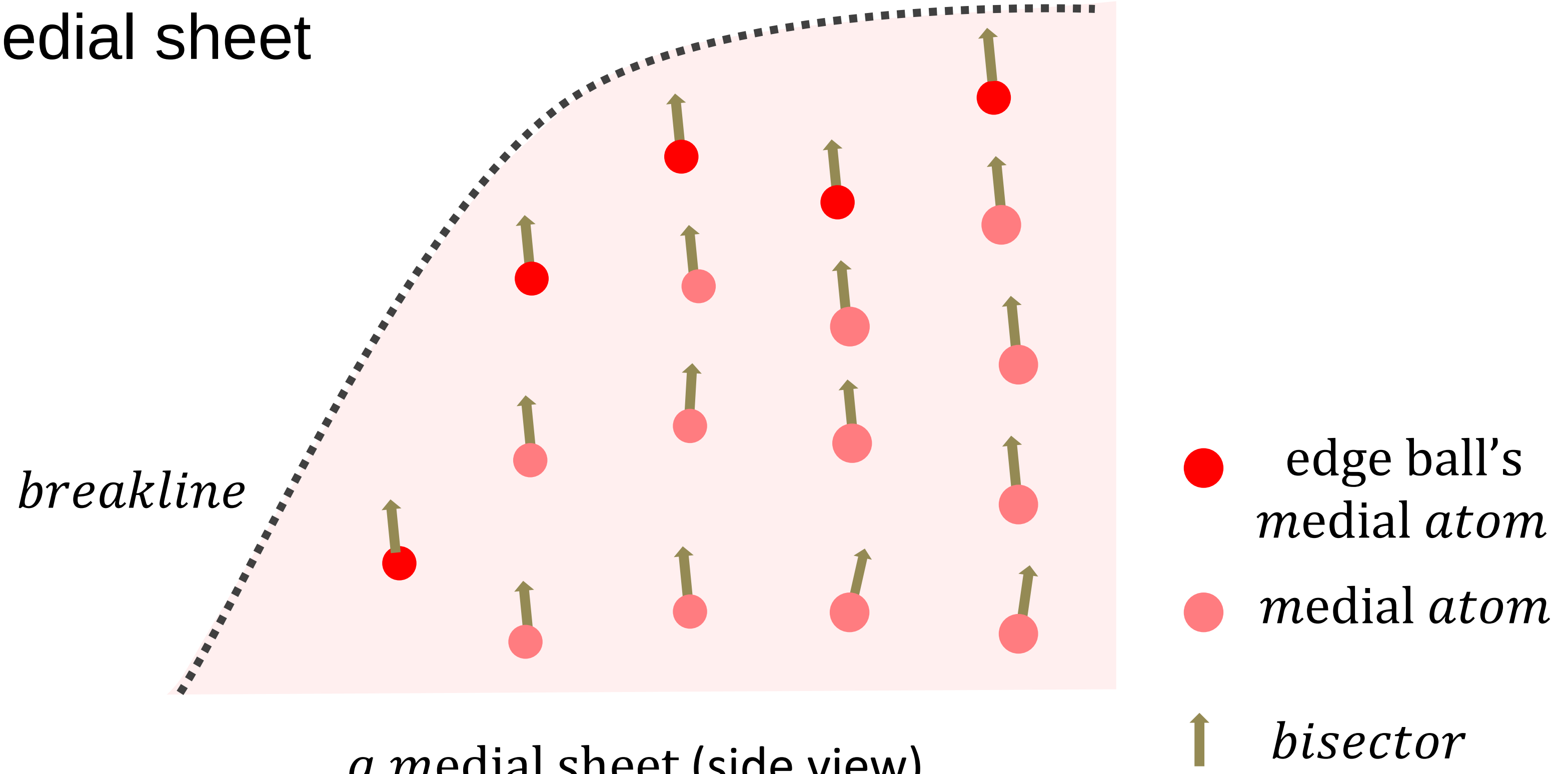
Methodology overview


① Estimate normal vector
Input point cloud
② MAT process
Medial atoms
+
Plane terrain points
Segment into medial sheets
Adjacency relationship
③ Extract candidate points
④ Generate polylines (two options)
⑤ The topology of breaklines
⑥ Polyline simplification and smoothing
⑦
Save to file

TU Delft

Methodology: extracting candidate points

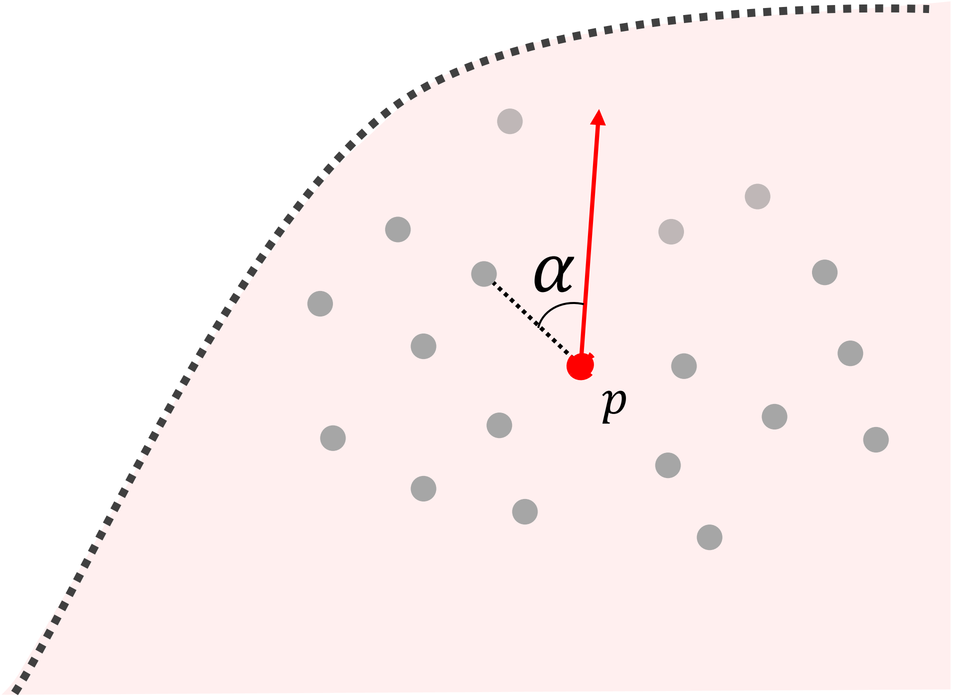
- Detecting edge balls
  - Edge balls: medial balls close to the edge of the medial sheet



a medial sheet (side view)

Methodology: extracting candidate points

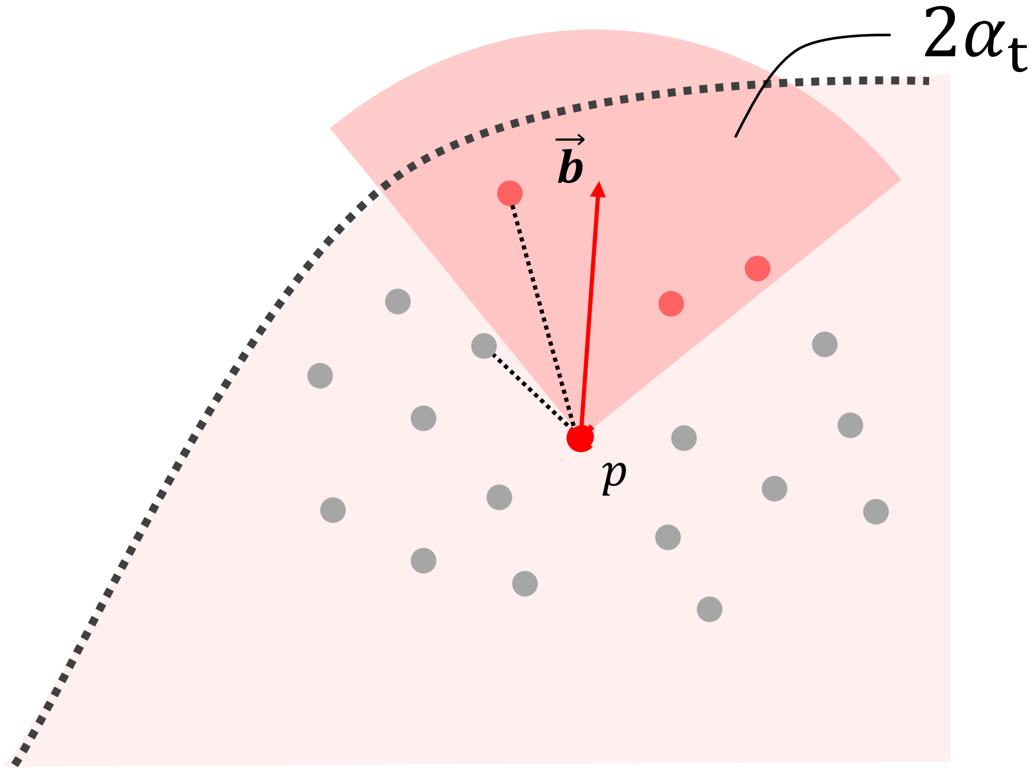
- Detecting edge balls



Calculate α for each neighbor

Methodology: extracting candidate points

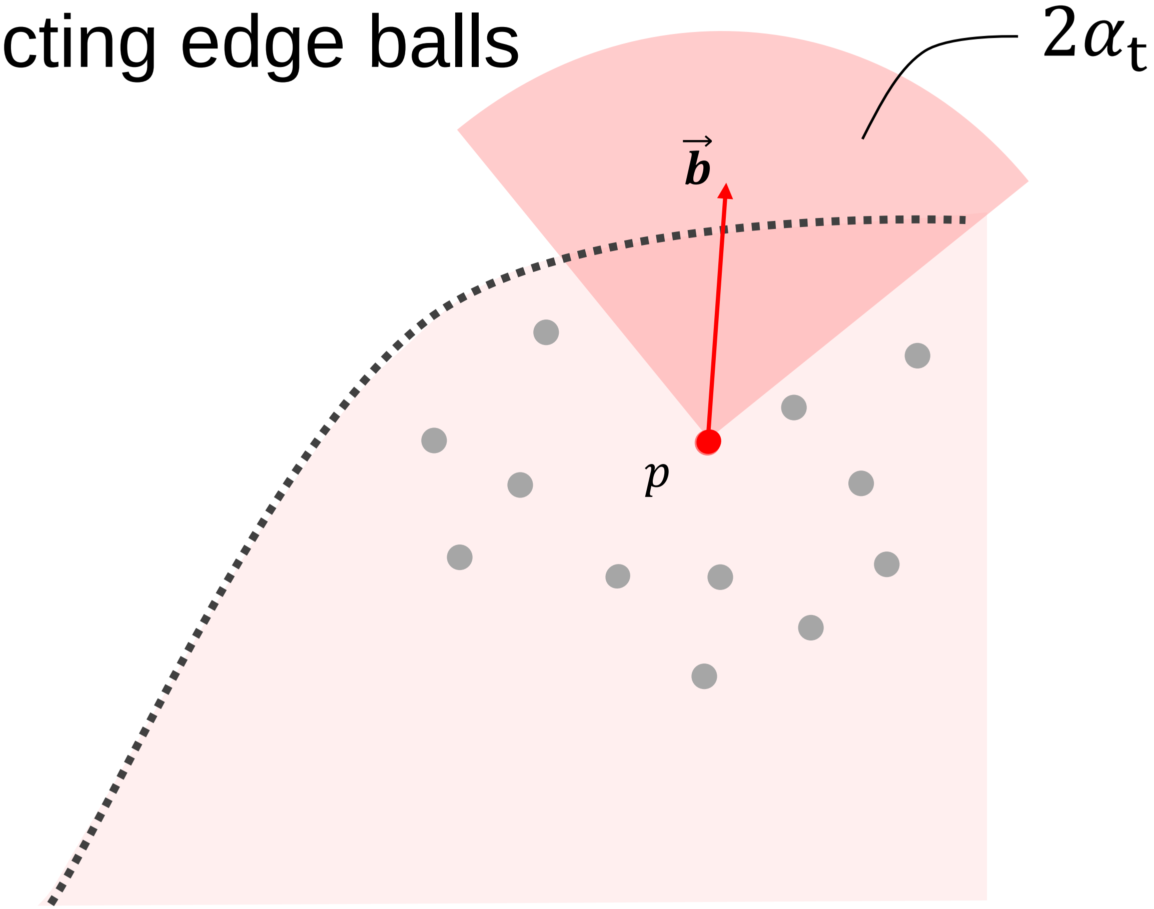
- Detecting edge balls



p is not an edge ball

Methodology: extracting candidate points

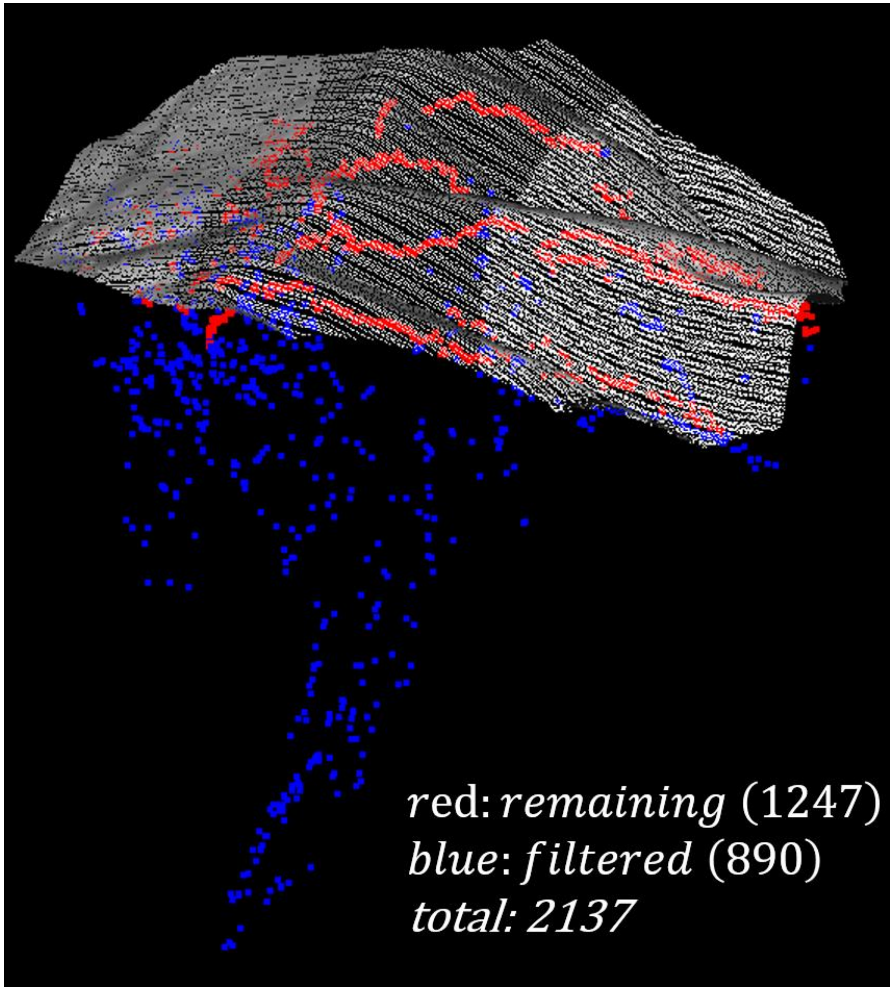
- Detecting edge balls



p is an edge ball

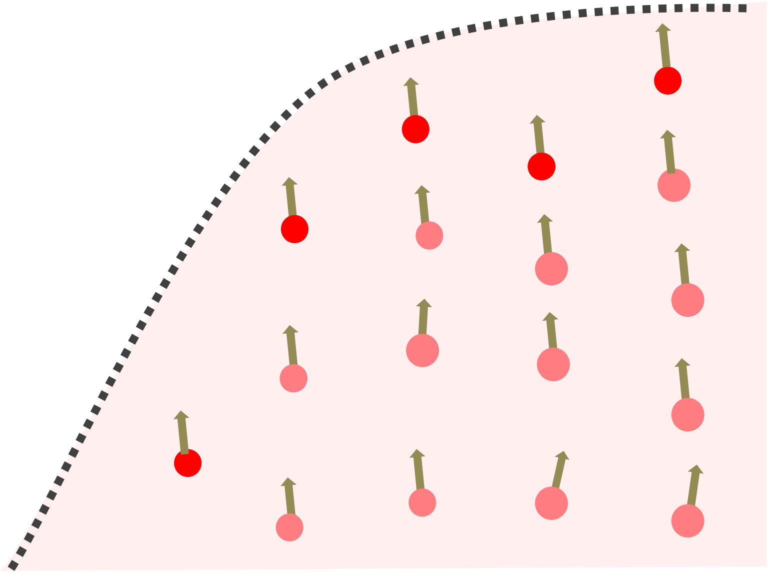
Methodology: extracting candidate points

- Filtering edge balls
  - Radius (r_{min}, r_{max}) → curvature $(1/r_{max}, 1/r_{min})$
  - Distance to the point cloud (3D)
  - Distance to the planar area (2D)



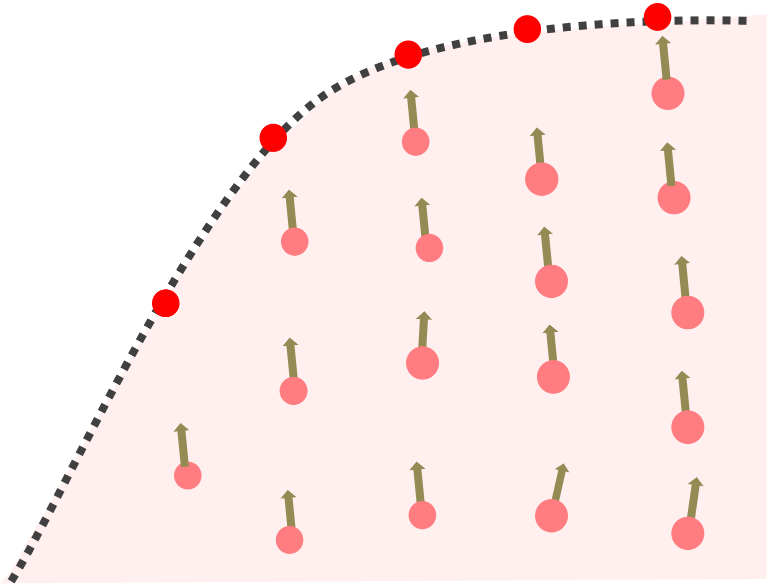
Methodology: extracting candidate points

- Remaining edge ball → candidate point



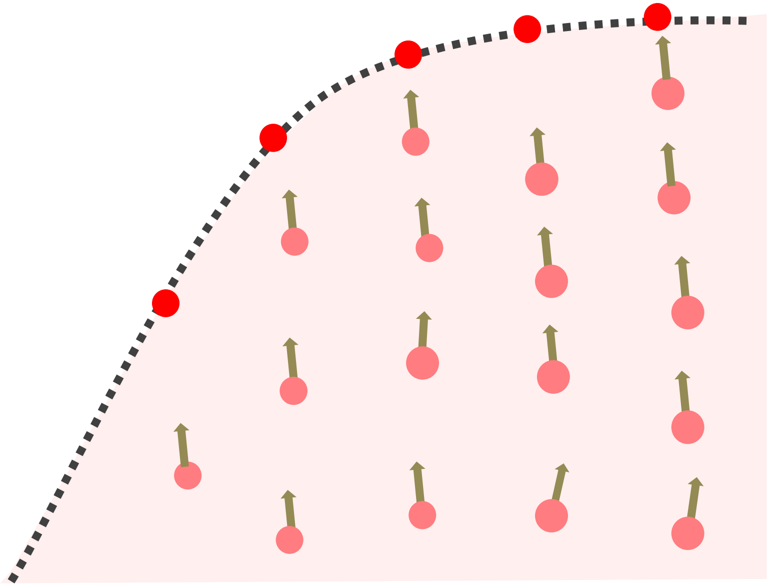
Methodology: extracting candidate points

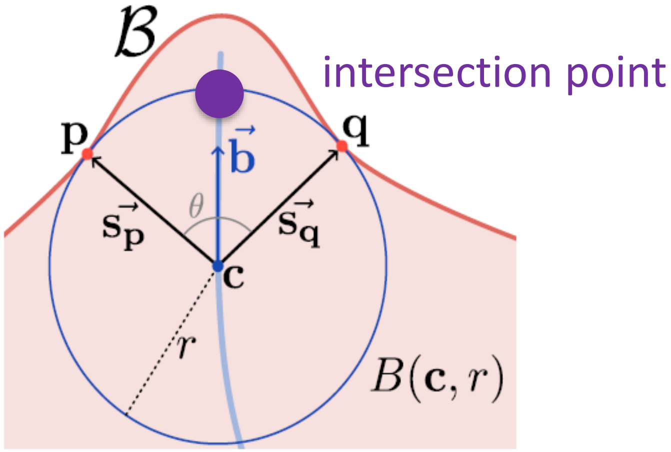
- Remaining edge ball → candidate point



Methodology: extracting candidate points

- Remaining edge ball → candidate point





Intersection of bisector and the medial ball

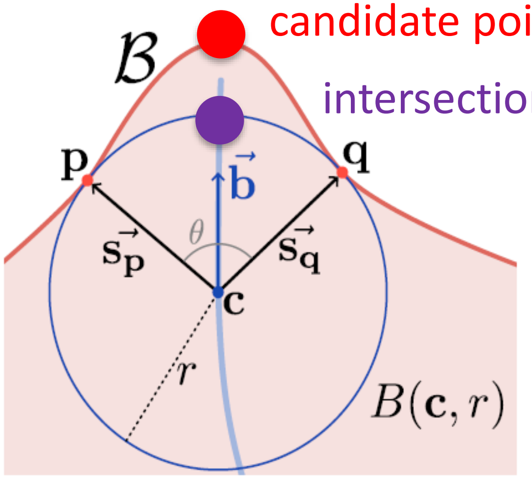
(x, y, z)

Methodology: extracting candidate points

- Remaining edge ball → candidate point

candidate point

intersection point

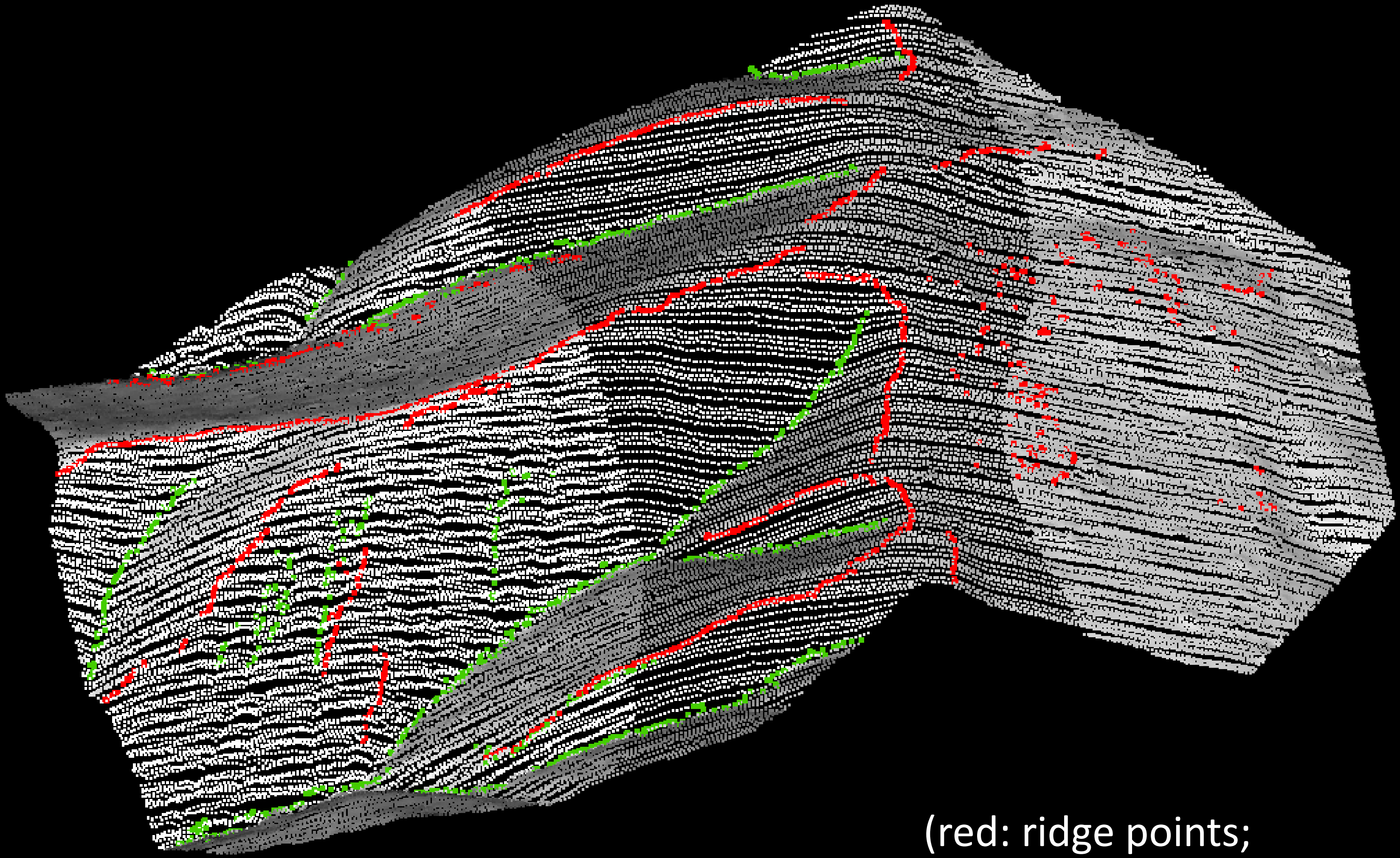


(x,y)

Intersection of bisector and the medial ball

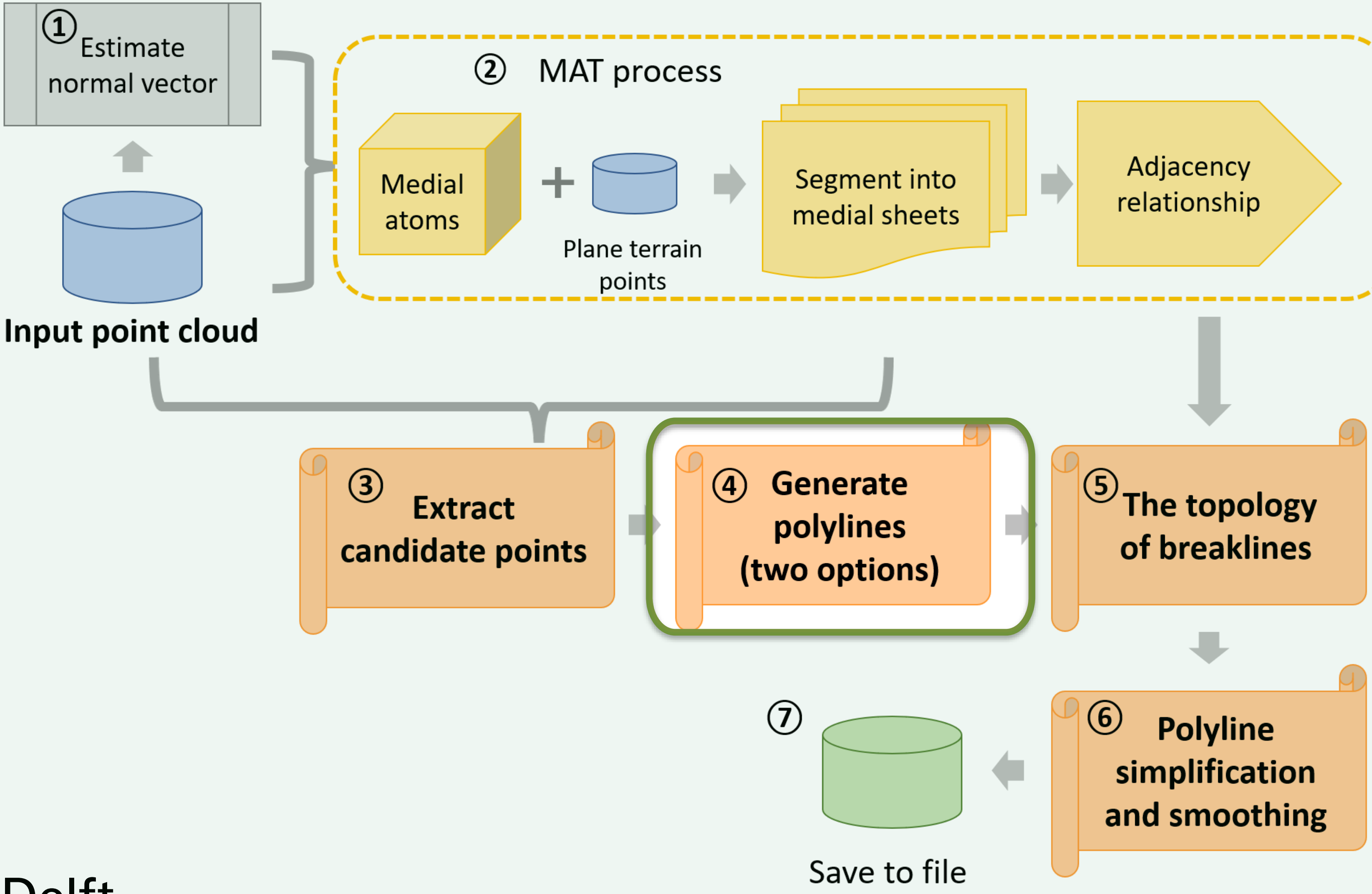
(x, y, z) (x, y, z)

Methodology: extracting candidate points



(red: ridge points;
Green: valley points)

Methodology overview



① Estimate normal vector

Input point cloud

② MAT process

Medial atoms

+

Plane terrain points

Segment into medial sheets

Adjacency relationship

③ Extract candidate points

④ Generate polylines (two options)

⑤ The topology of breaklines

⑥ Polyline simplification and smoothing

⑦

Save to file

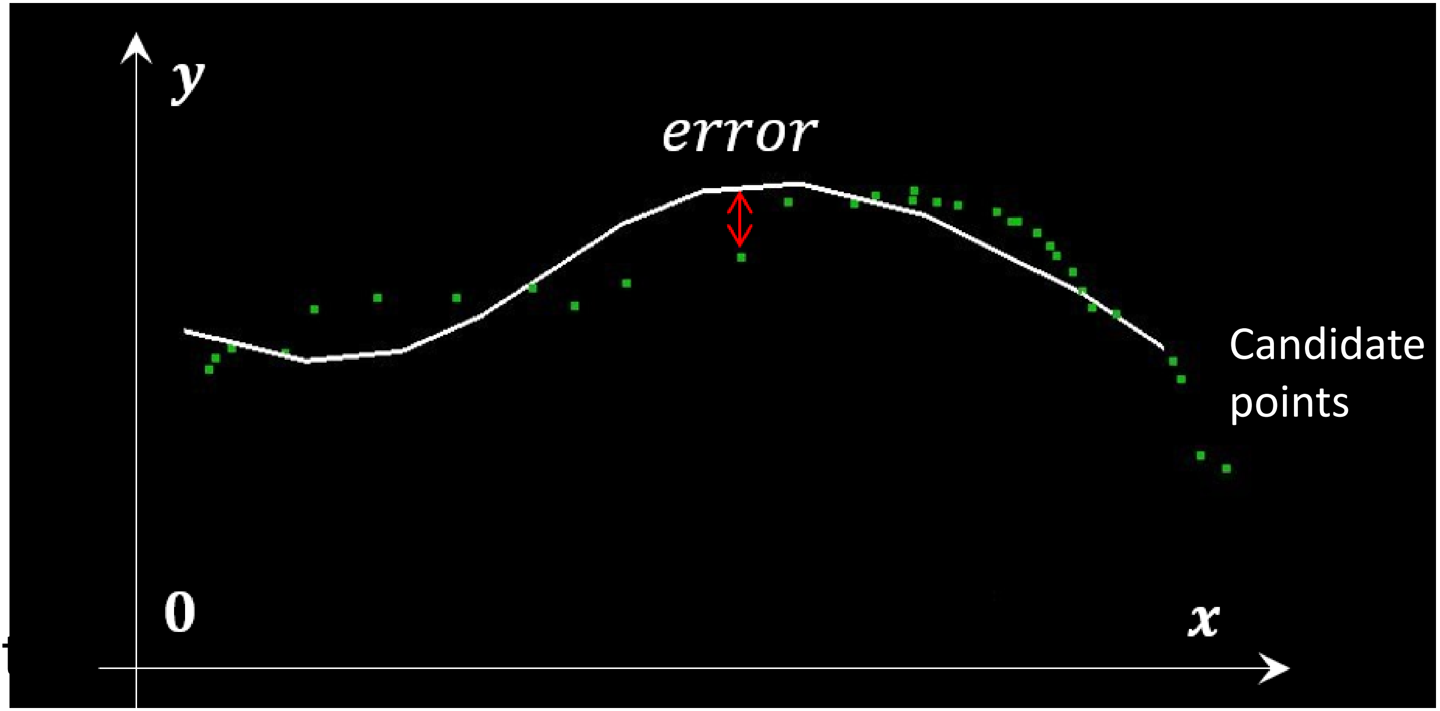
Option 1 : generating polylines using the polynomial fitting

Option 2 : generating polylines using the graph theory

Methodology: generating polylines using the polynomial fitting

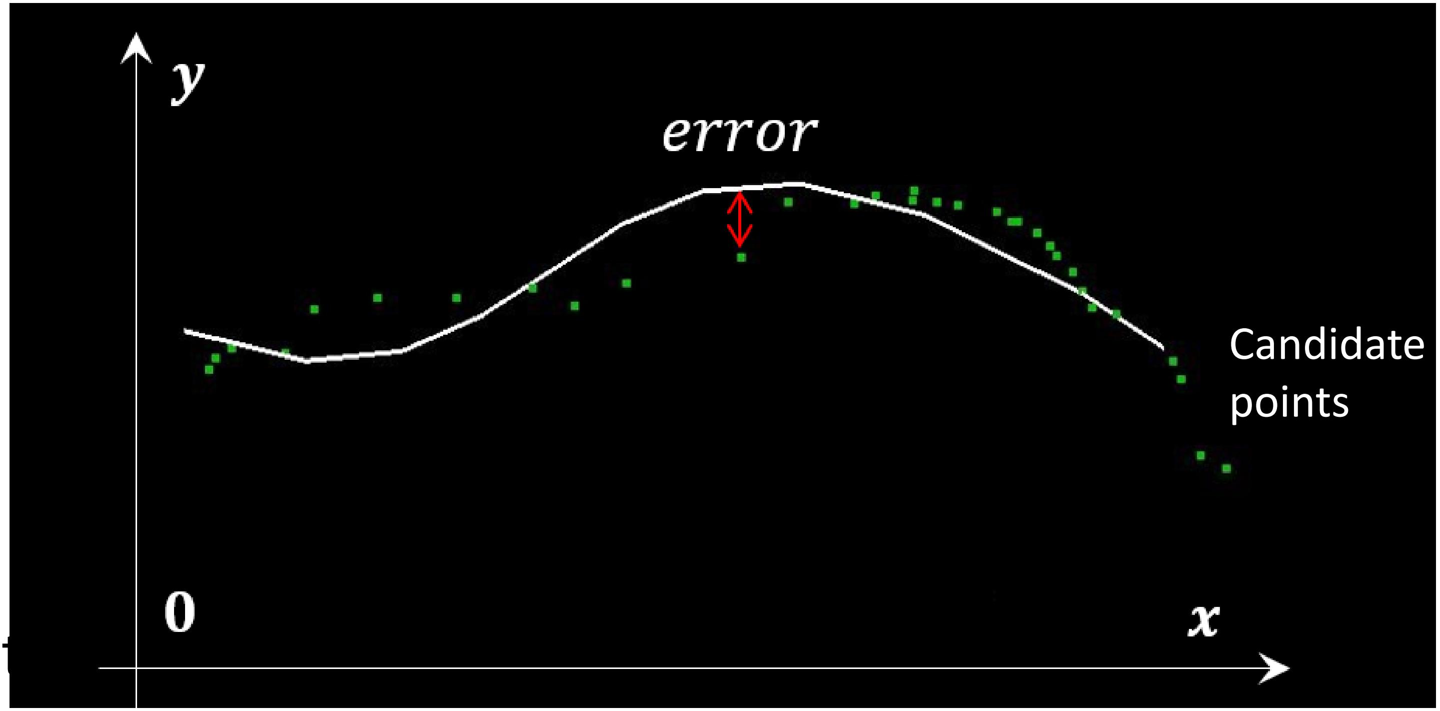
- For each medial sheet：
  - Fitting a cubic polynomial function with the candidate points

$$y = f(x) = t_3 \cdot x^3 + t_2 \cdot x^2 + t_1 \cdot x + t_0$$



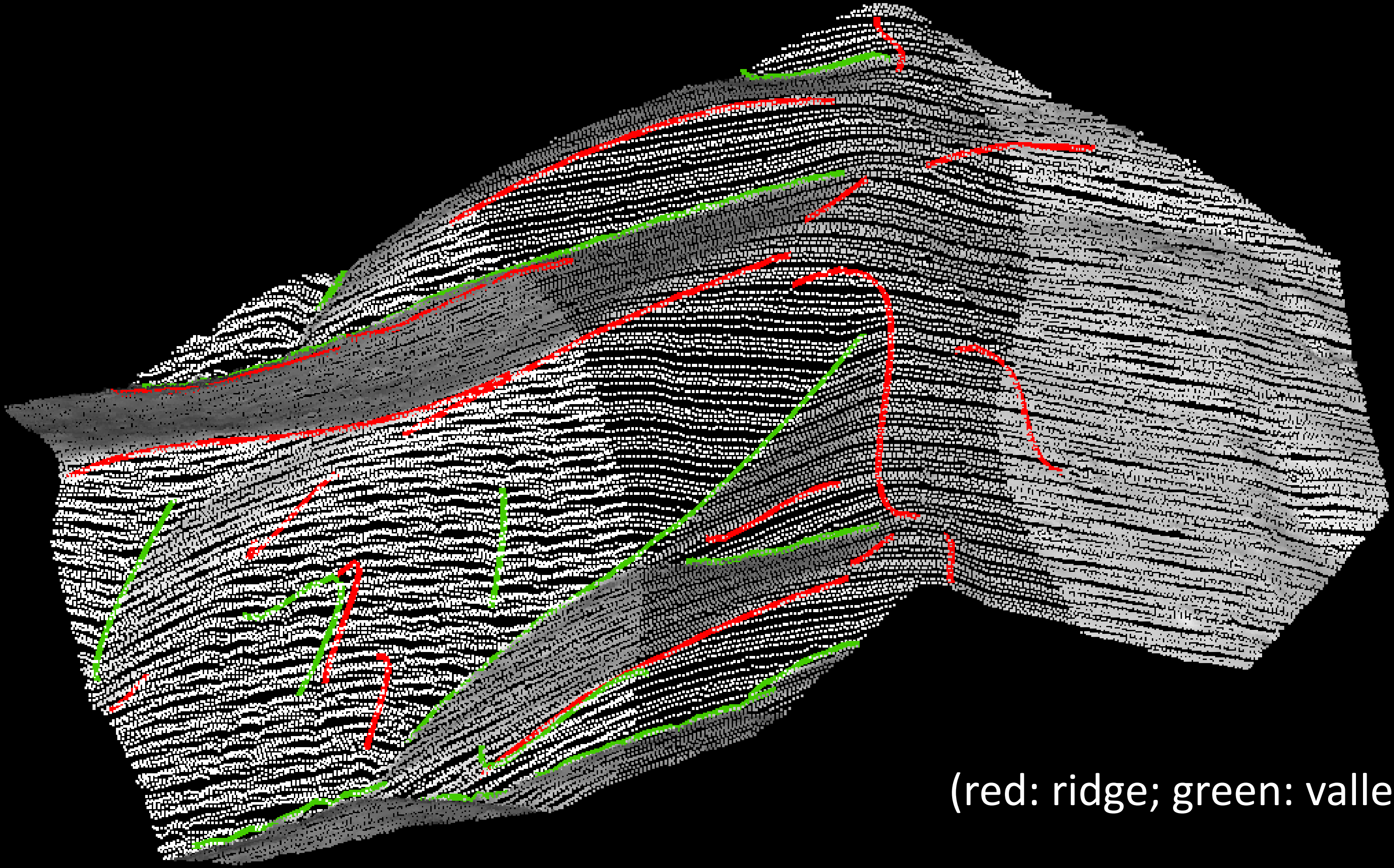
Methodology: generating polylines using the polynomial fitting

- For each medial sheet：
  - Fitting a cubic polynomial function with the candidate points
  - Eliminating unexpected breaklines by RMSR



TU Delft

Methodology: generating polylines using the polynomial fitting



(red: ridge; green: valley)

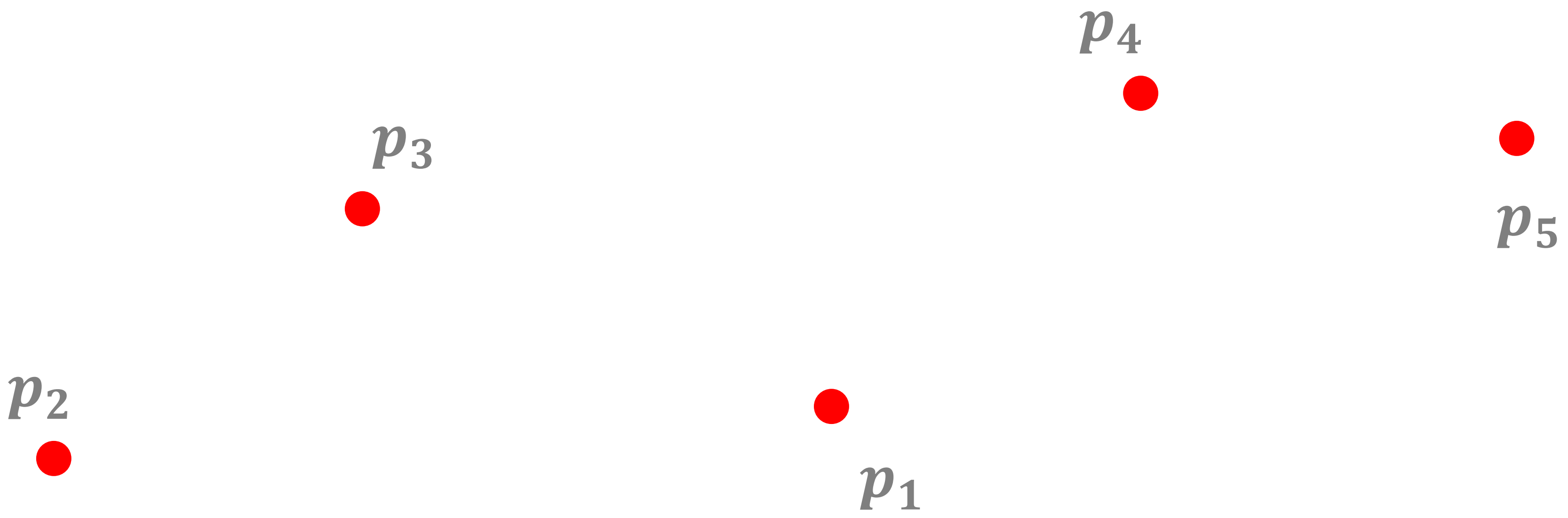
Methodology: generating polylines using the graph theory

Methodology: generating polylines using the graph theory

- For each medial sheet：
  - Connecting candidate point to its closest point by Minimum Spanning tree
  - Simplify to one polyline

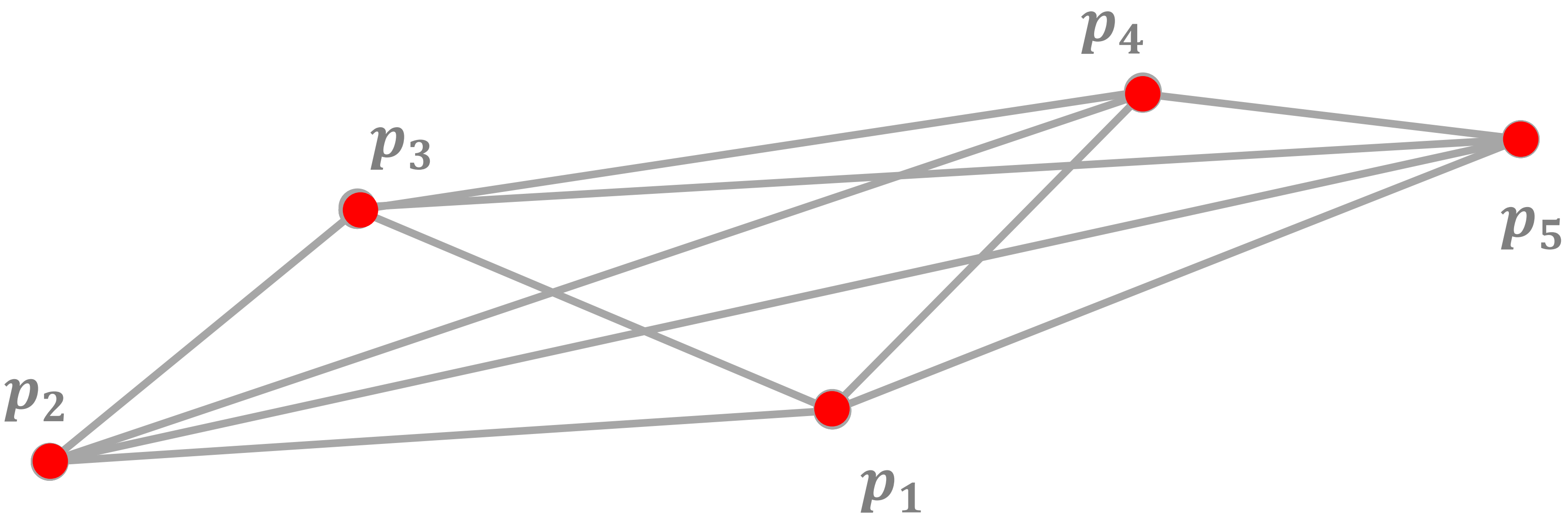
Methodology: generating polylines using the graph theory

- Connecting candidate point to its closest point by Minimum Spanning tree



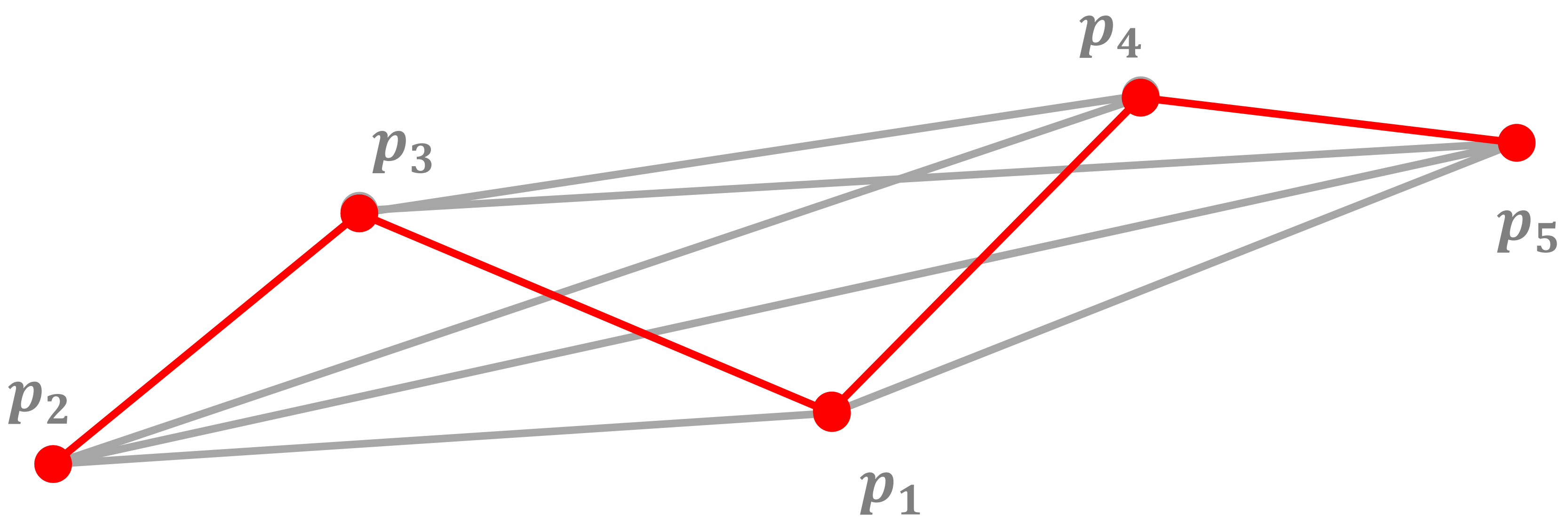
Methodology: generating polylines using the graph theory

- Connecting candidate point to its closest point by Minimum Spanning tree



Methodology: generating polylines using the graph theory

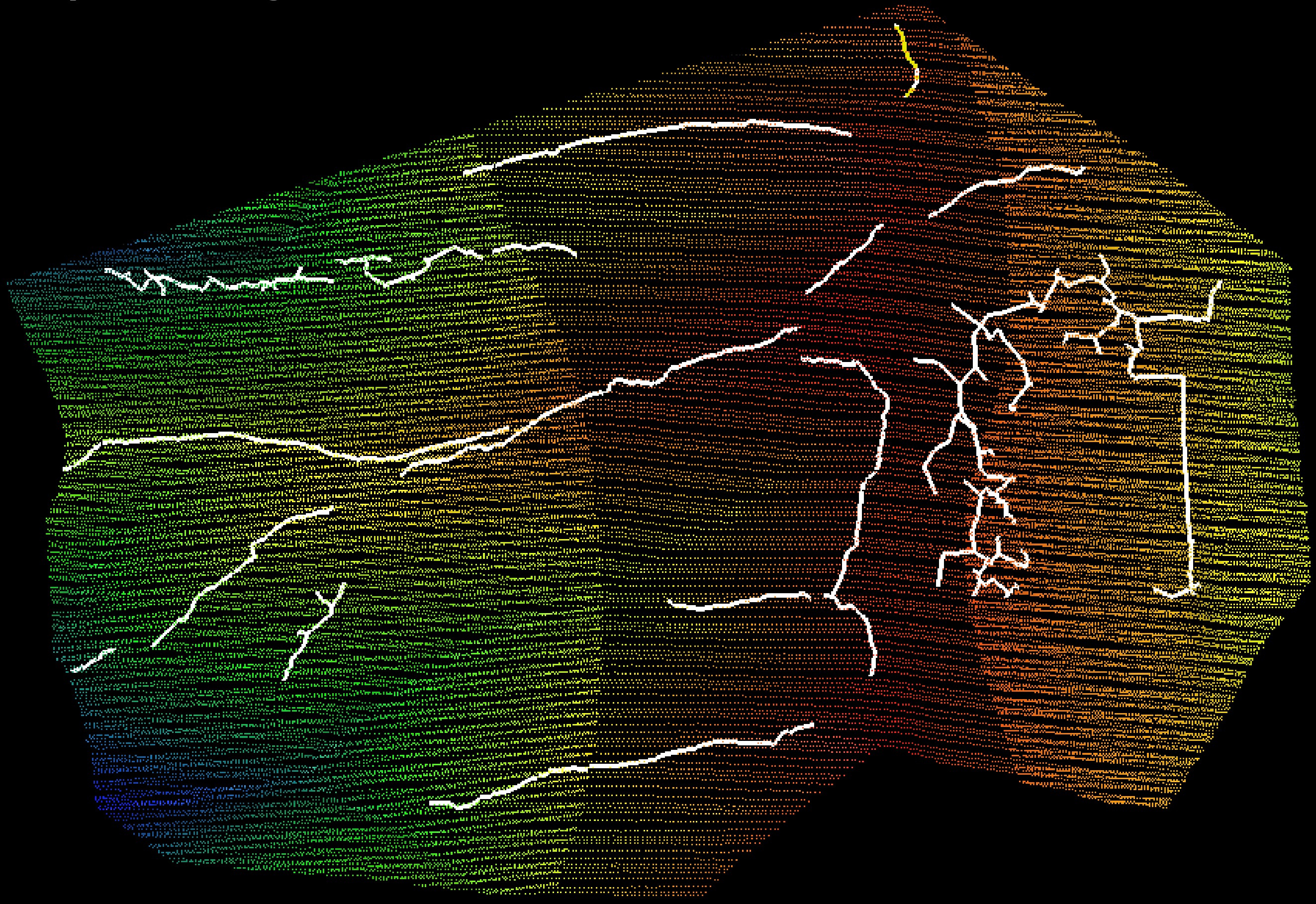
- Connecting candidate point to its closest point by Minimum Spanning tree



Methodology: generating polylines using the graph theory

- Connecting candidate point to its closest point by Minimum Spanning tree

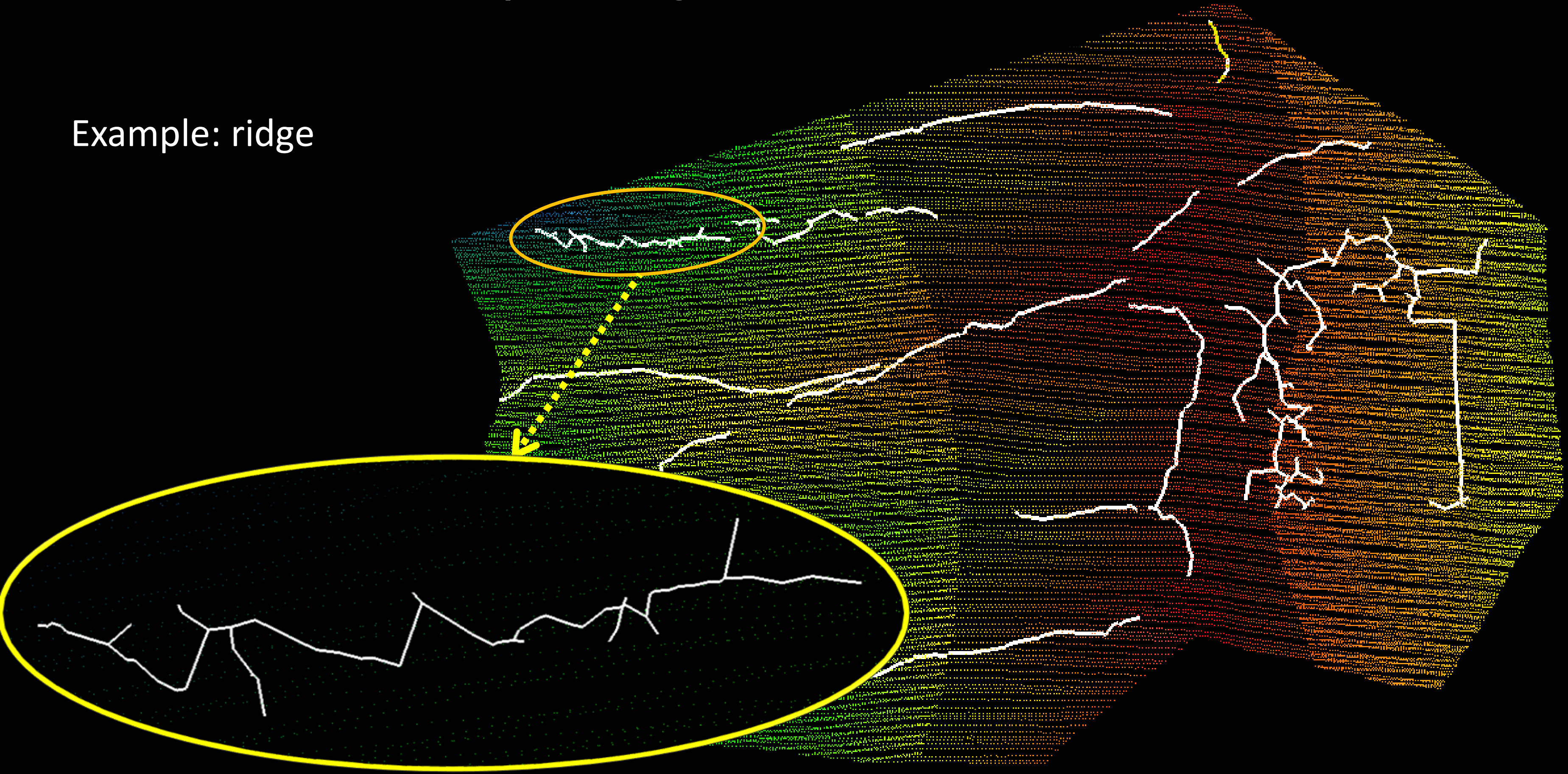
Example: ridge



Methodology: generating polylines using the graph theory

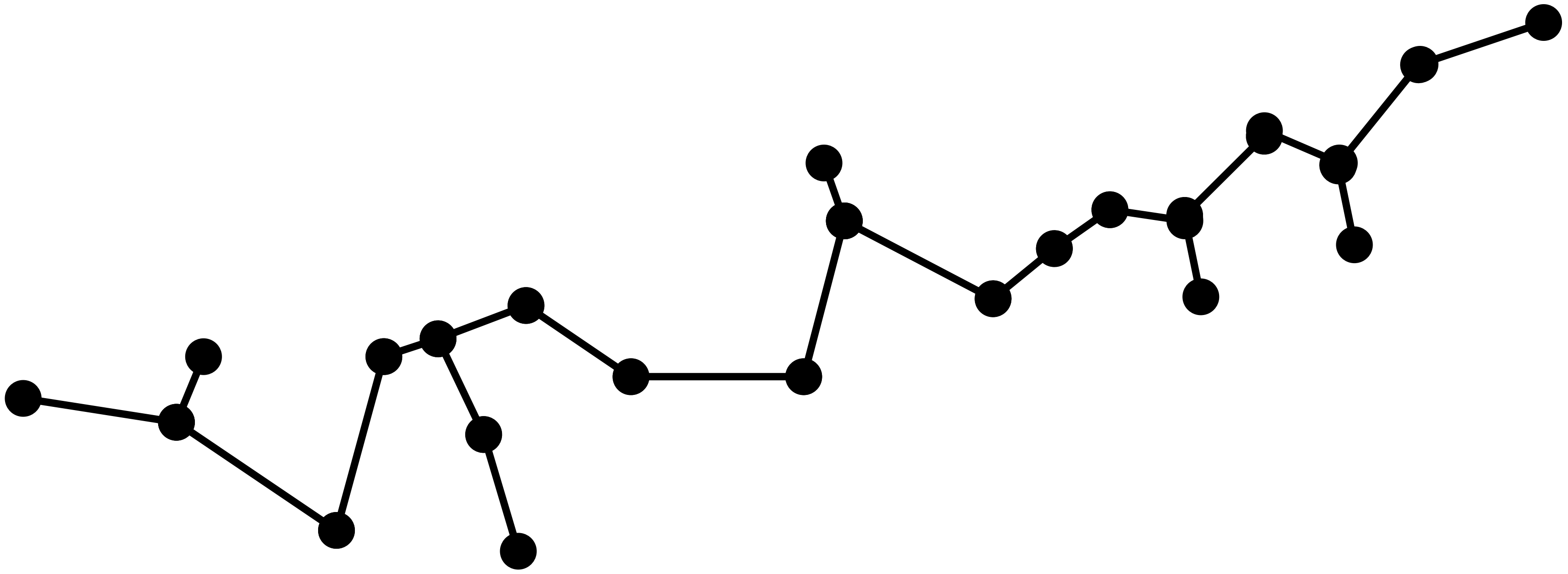
- Connecting candidate point to its closest point by Minimum Spanning tree

Example: ridge



Methodology: generating polylines using the graph theory

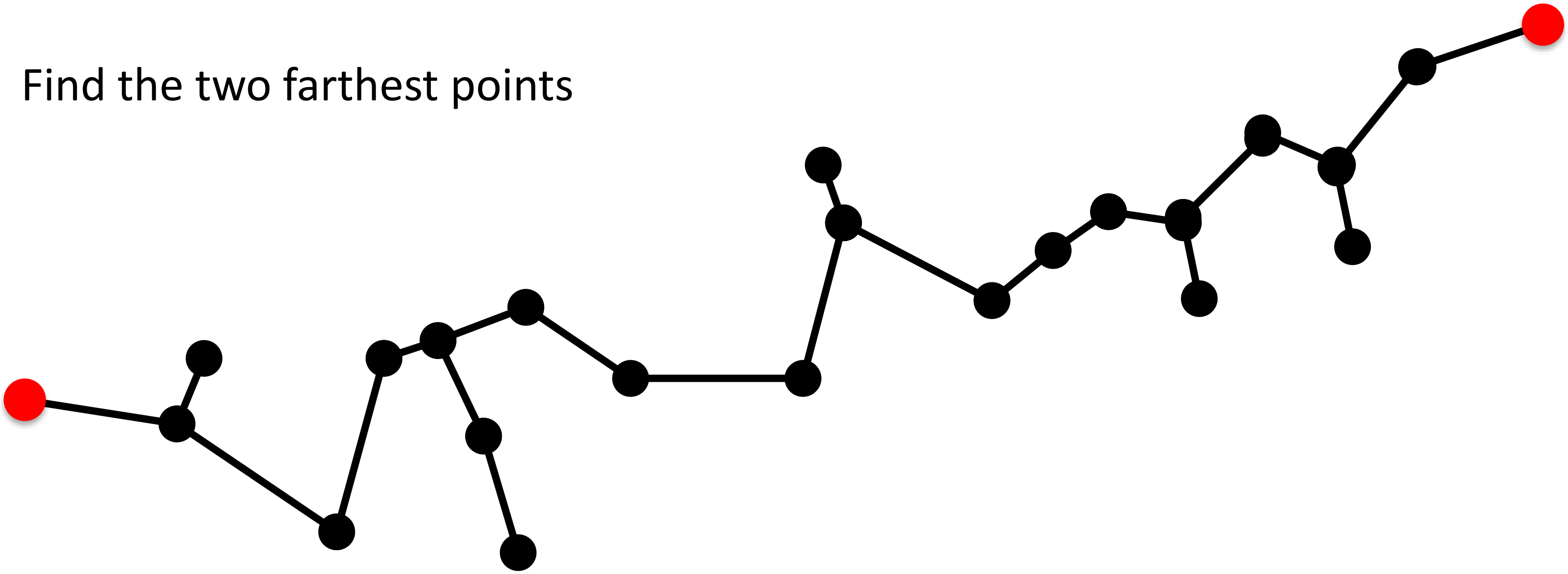
- Simplify to one polyline



Methodology: generating polylines using the graph theory

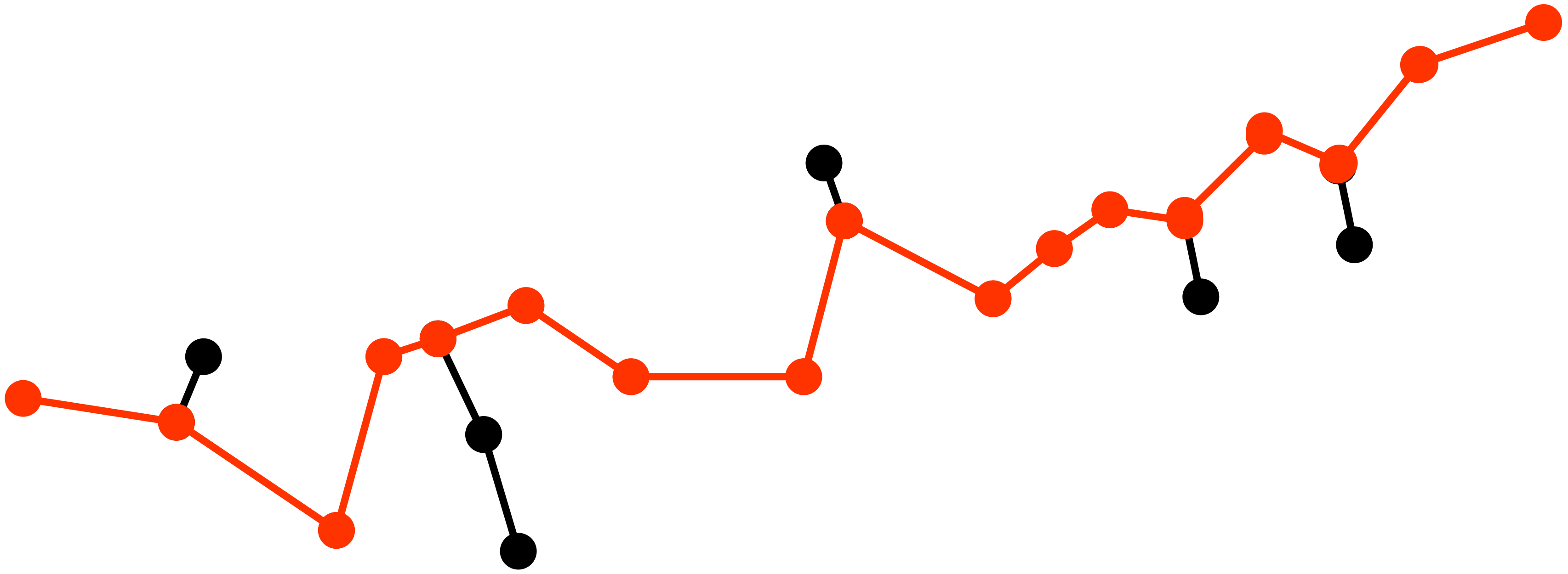
- Simplify to one polyline

Find the two farthest points



Methodology: generating polylines using the graph theory

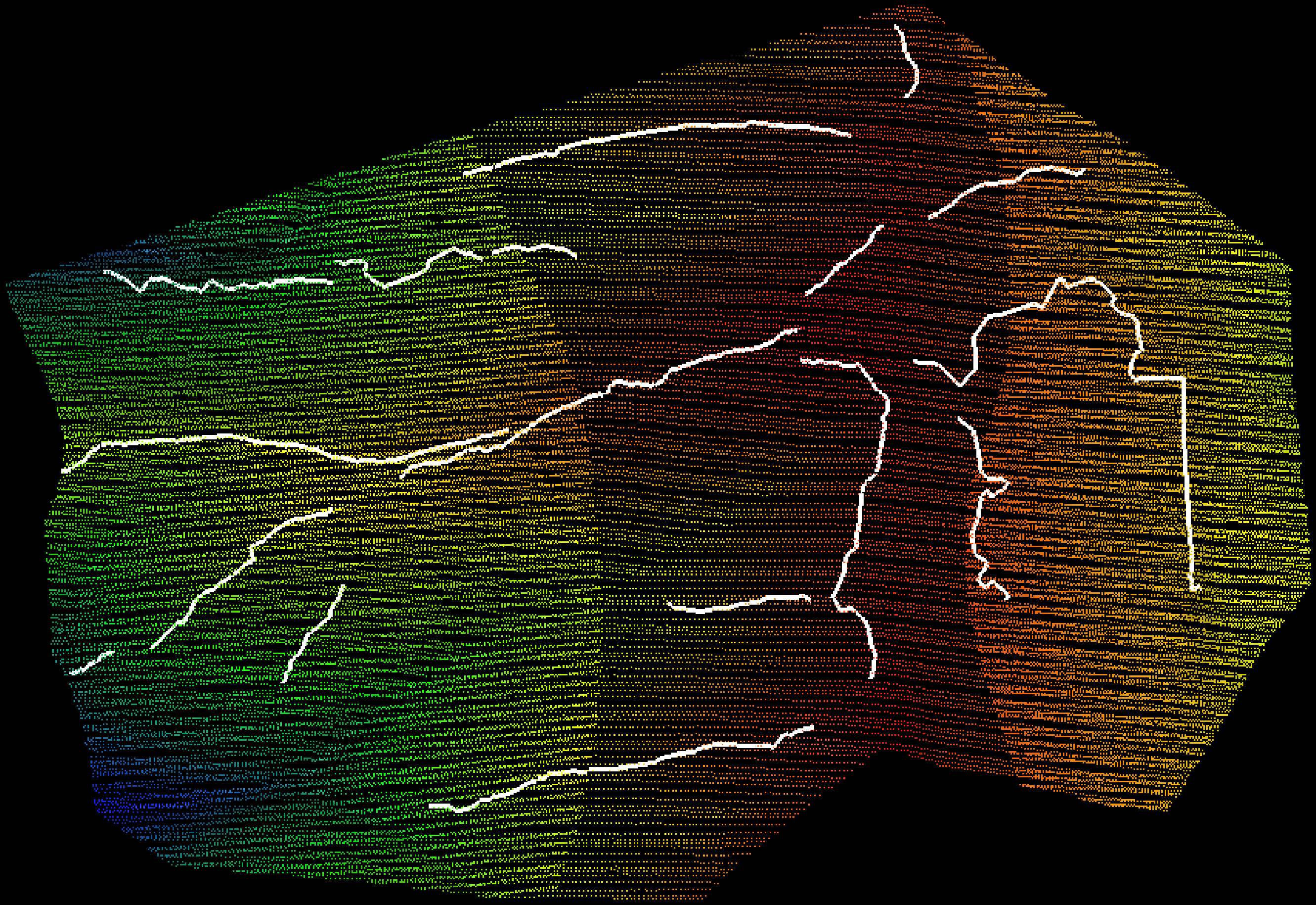
- Simplify to one polyline



Methodology: generating polylines using the graph theory

- – Simplify to one polyline by the shortest path algorithm

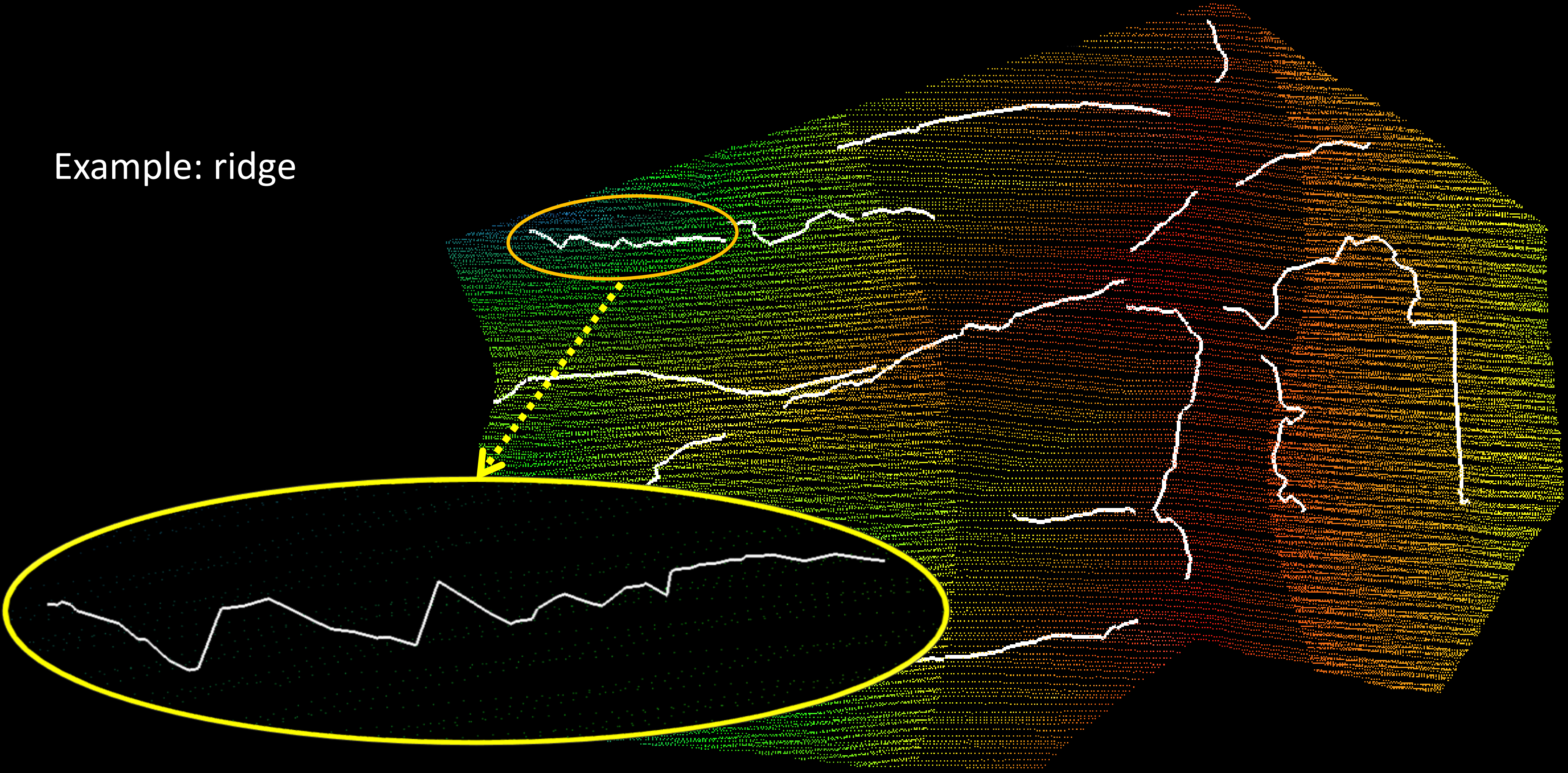
Example: ridge



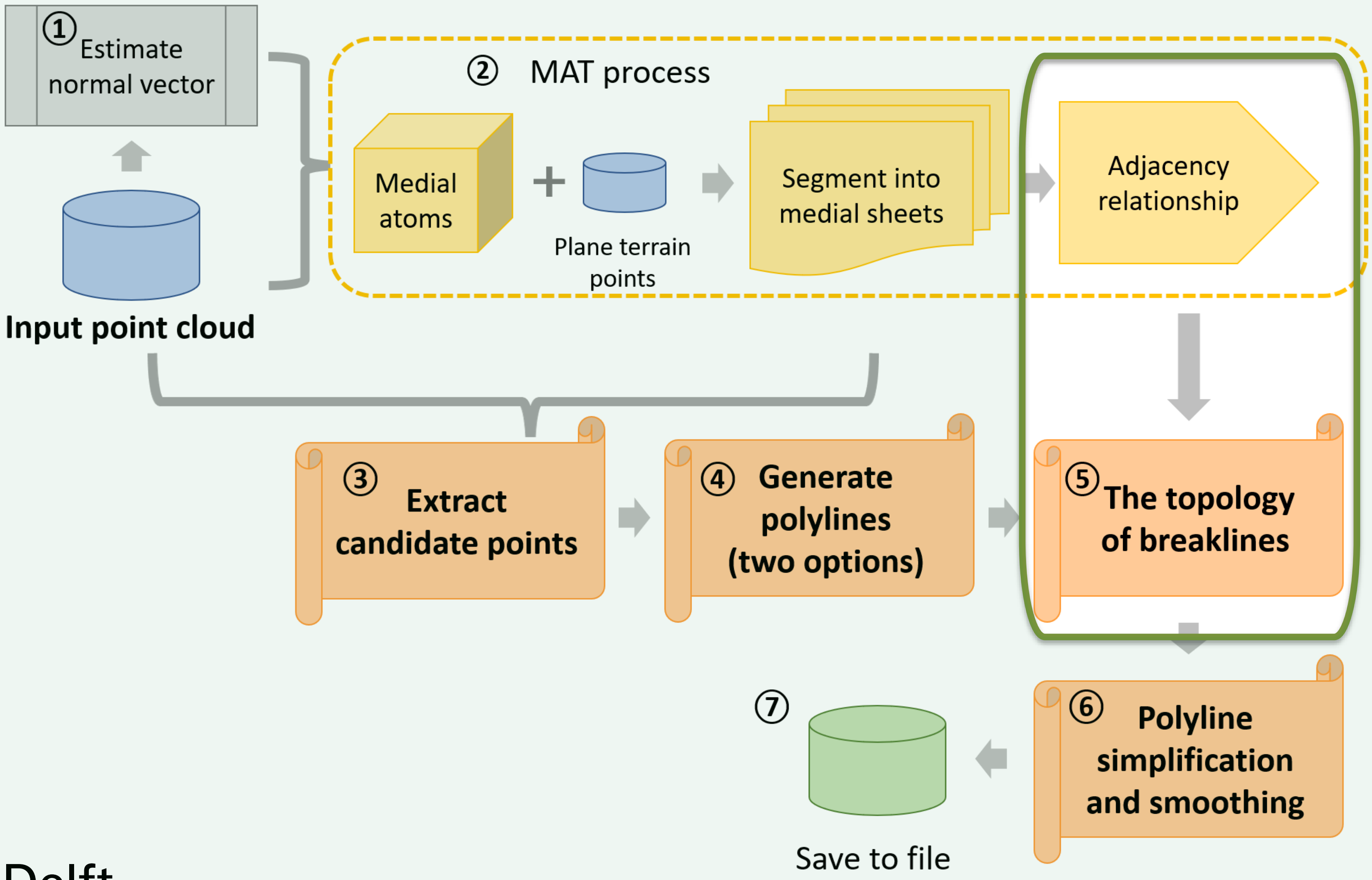
Methodology: generating polylines using the graph theory

- Simplify to one polyline by the shortest path algorithm

Example: ridge



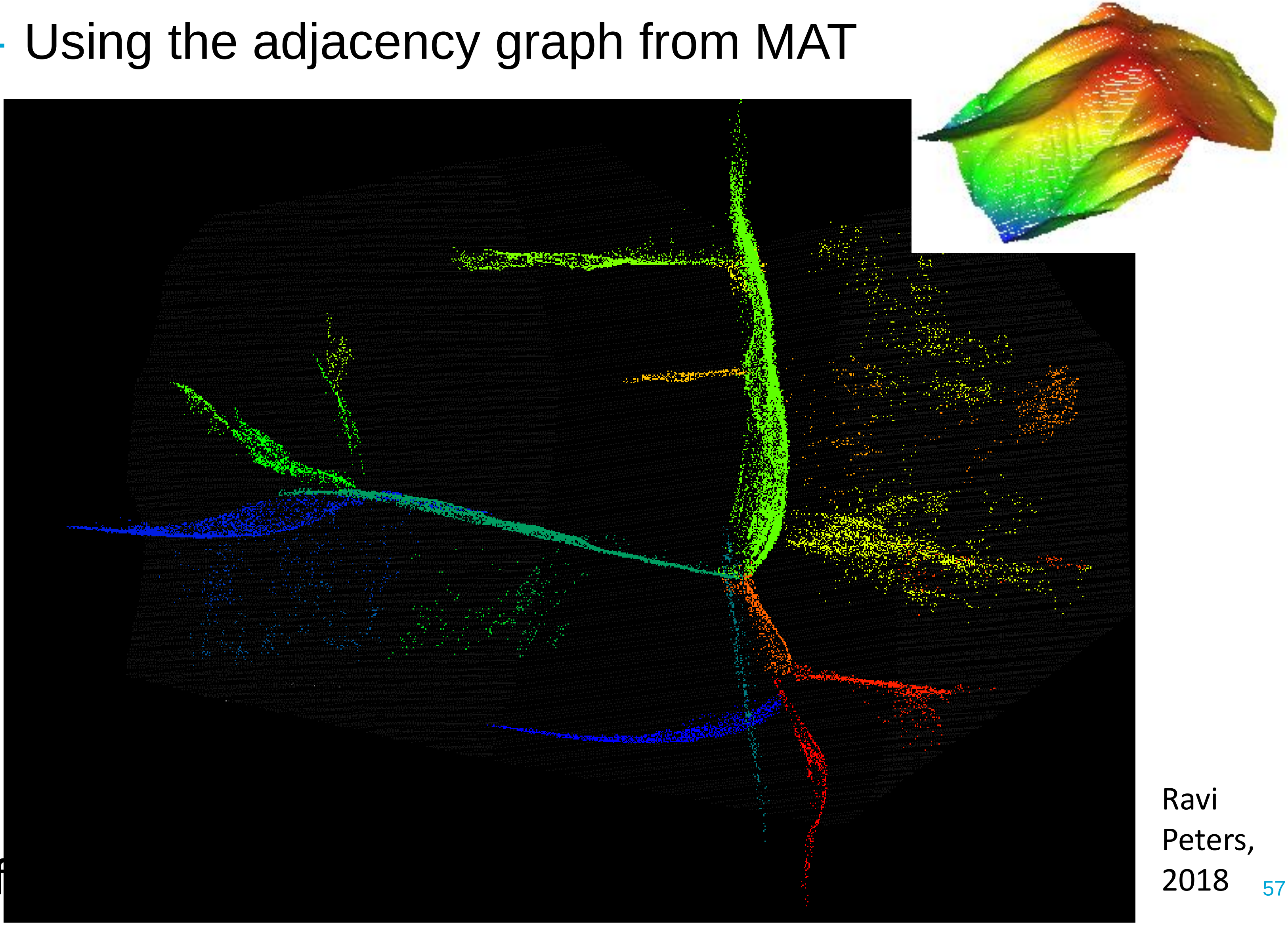
Methodology overview


① Estimate normal vector
Input point cloud
② MAT process
Medial atoms
+
Plane terrain points
Segment into medial sheets
Adjacency relationship
③ Extract candidate points
④ Generate polylines (two options)
⑤ The topology of breaklines
⑥ Polyline simplification and smoothing
⑦
Save to file

TUDelft

Methodology: the topology of breaklines

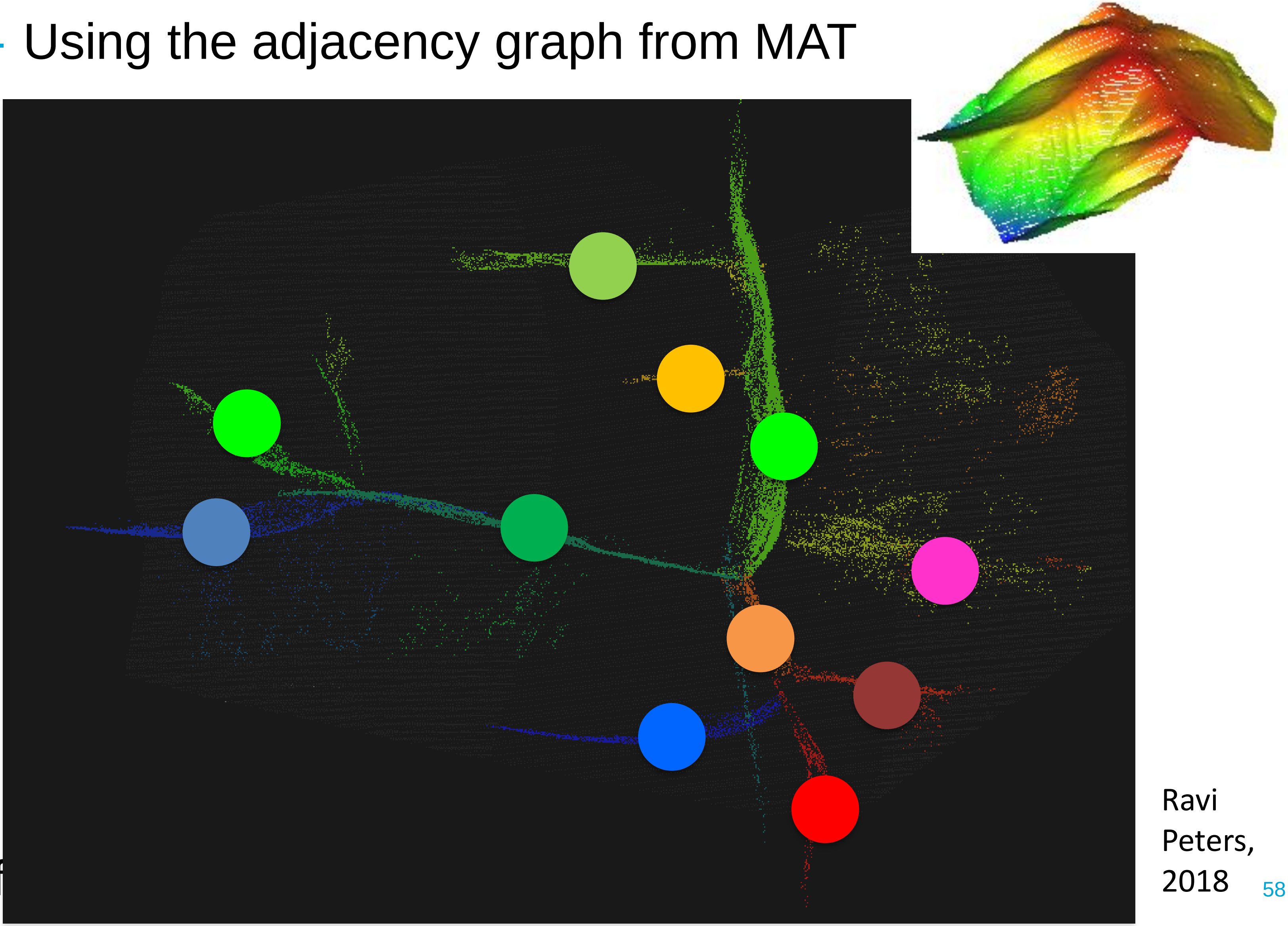
- Using the adjacency graph from MAT



Ravi Peters, 2018

Methodology: the topology of breaklines

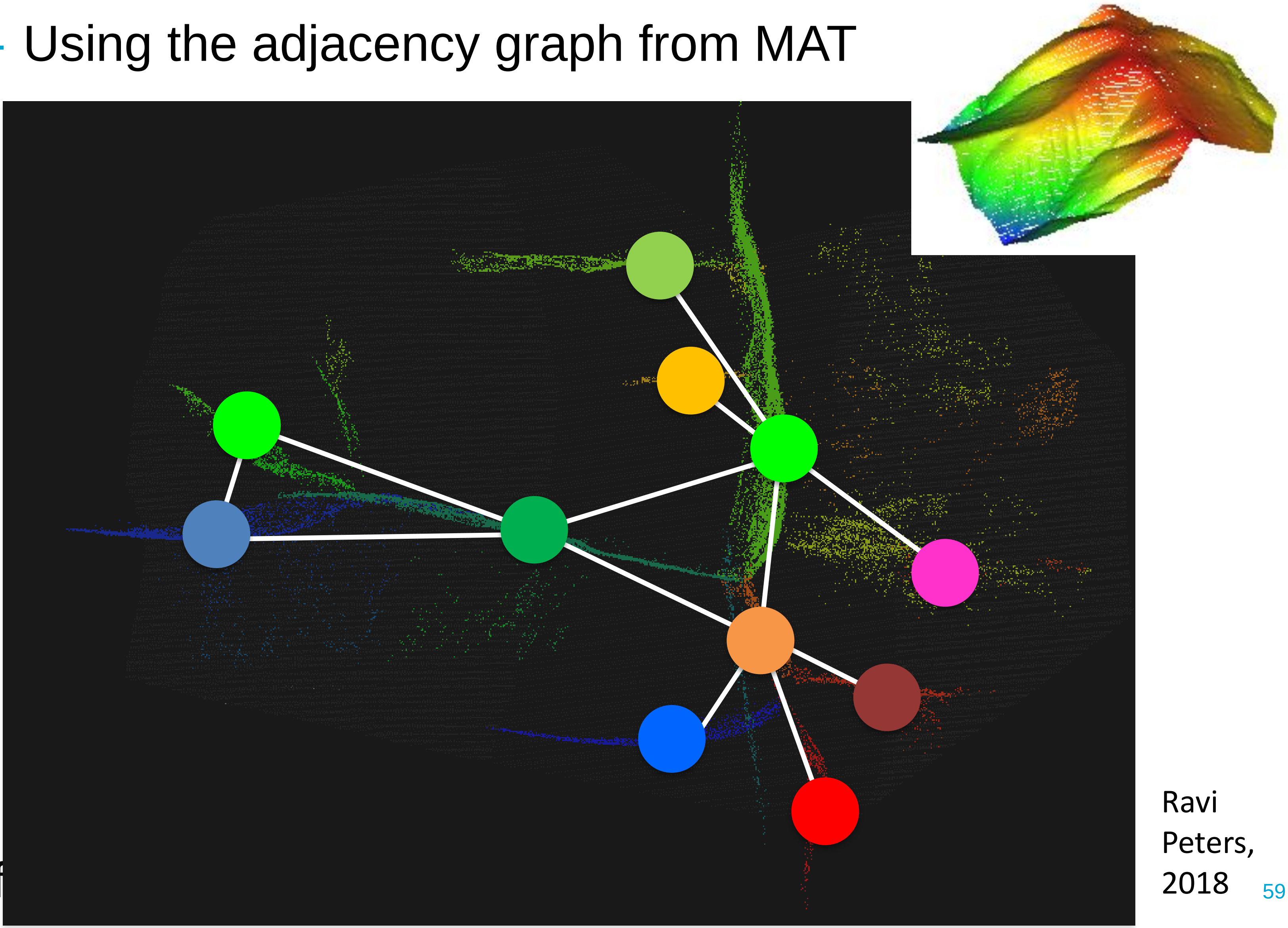
- Using the adjacency graph from MAT



Ravi Peters, 2018

Methodology: the topology of breaklines

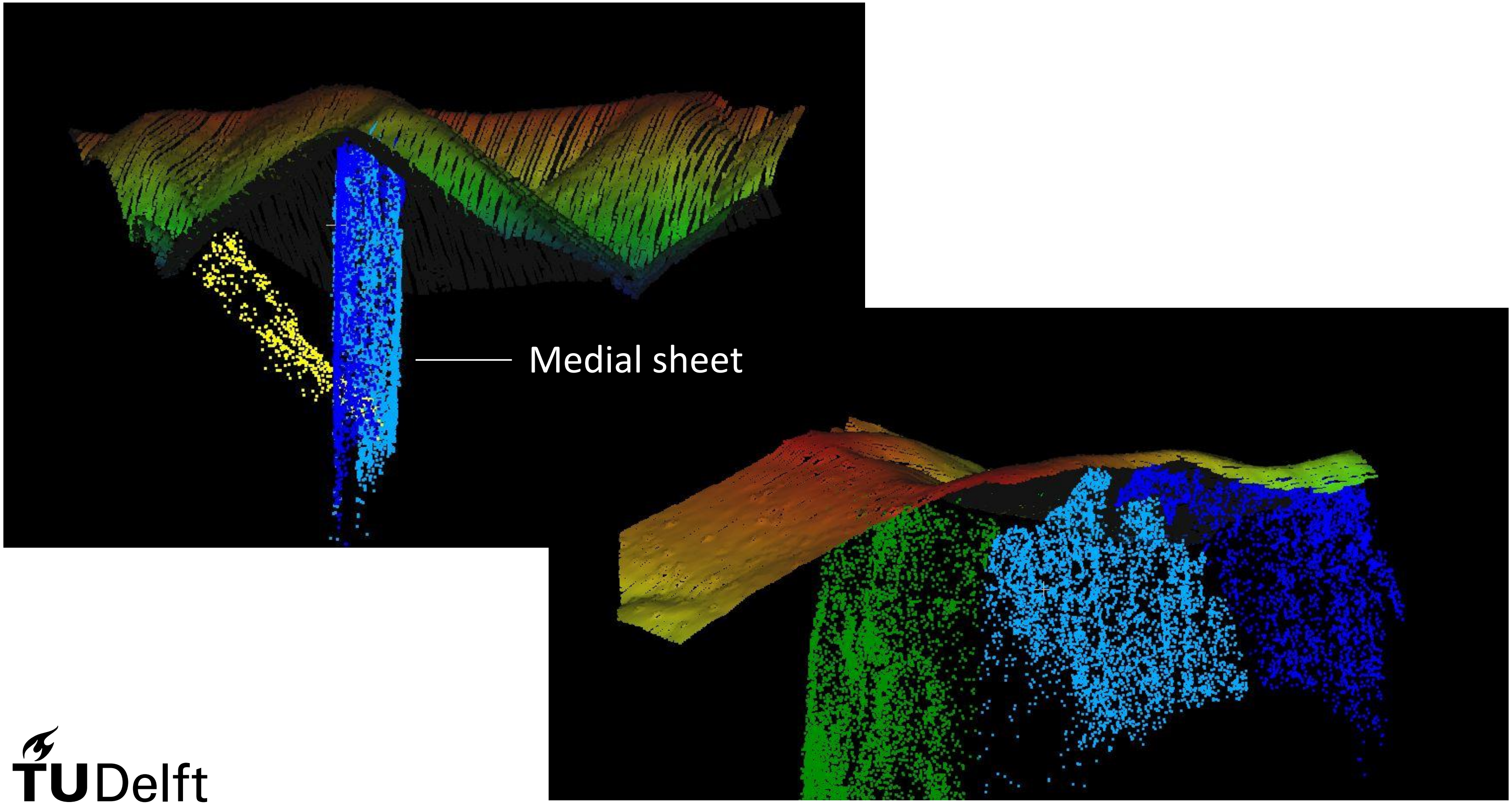
- Using the adjacency graph from MAT



Ravi Peters, 2018

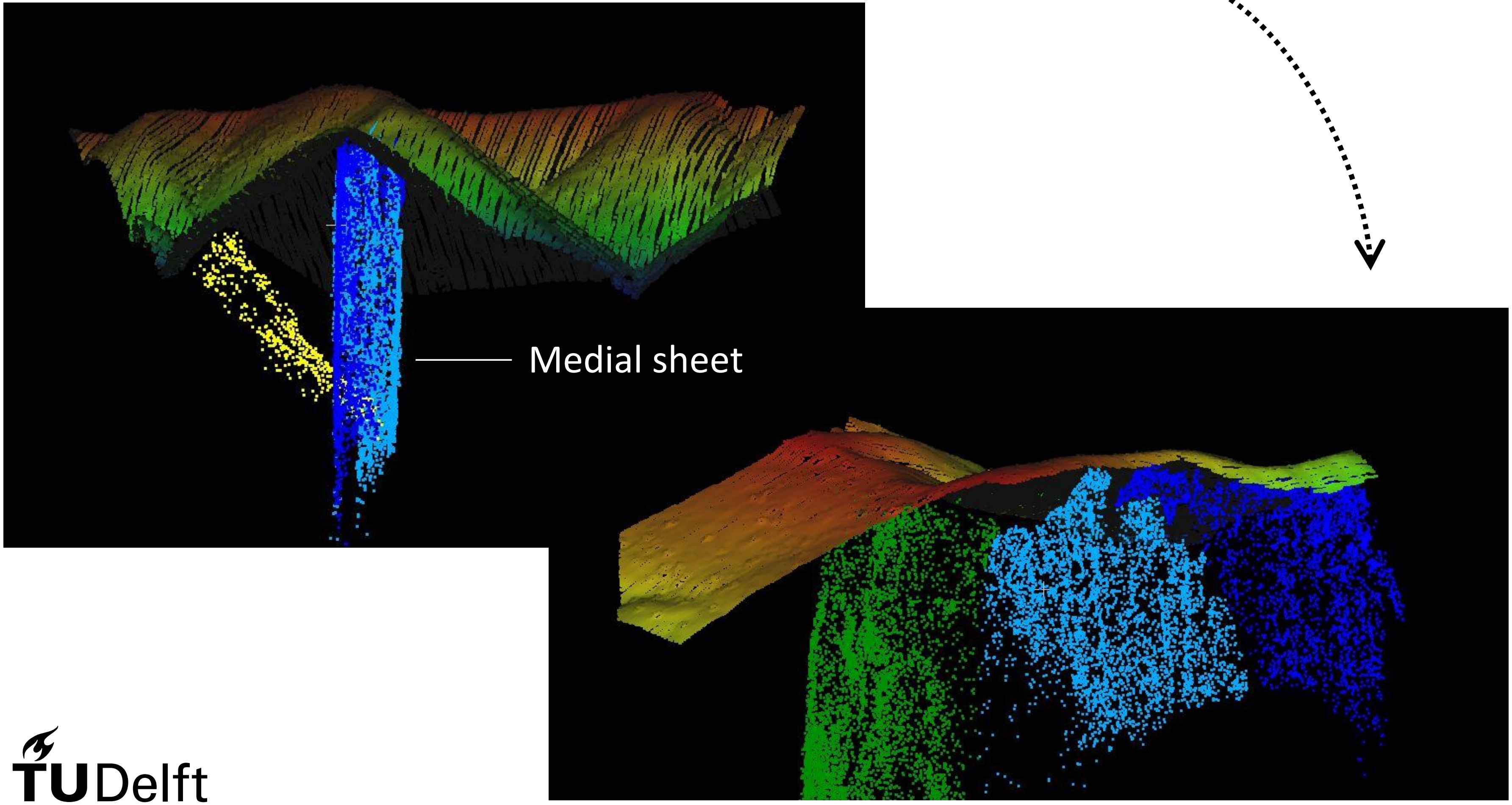
Methodology: the topology of breaklines

- Using the adjacency graph from MAT



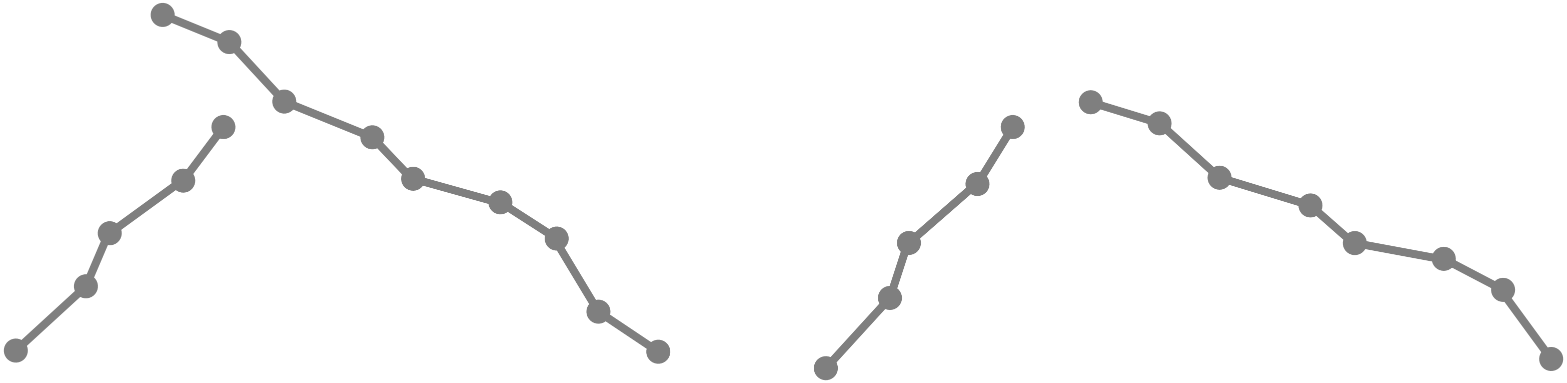
Methodology: the topology of breaklines

- Using the adjacency graph from MAT
- Only the surface adjacency is required



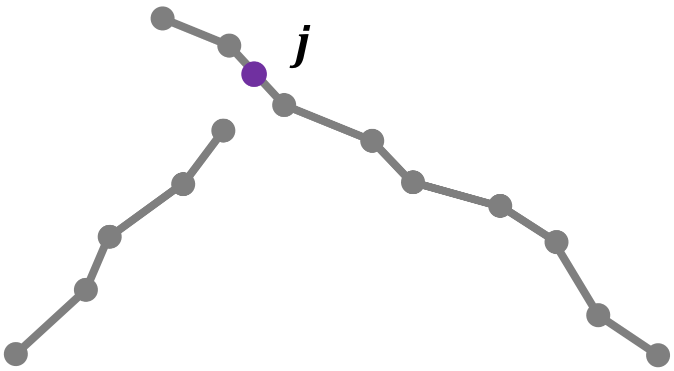
Methodology: the topology of breaklines

- connecting two breaklines

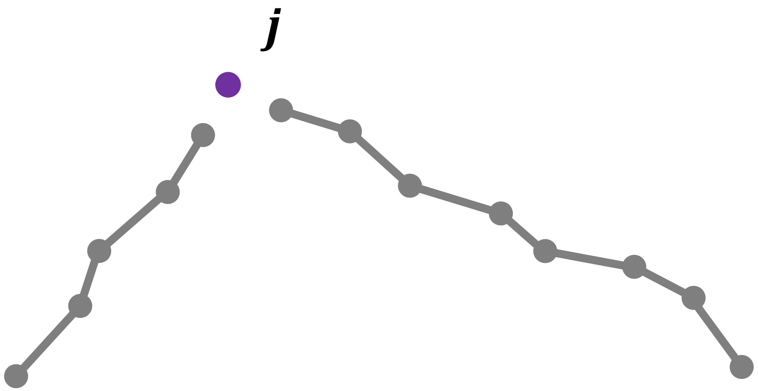


Methodology: the topology of breaklines

- connecting two breaklines



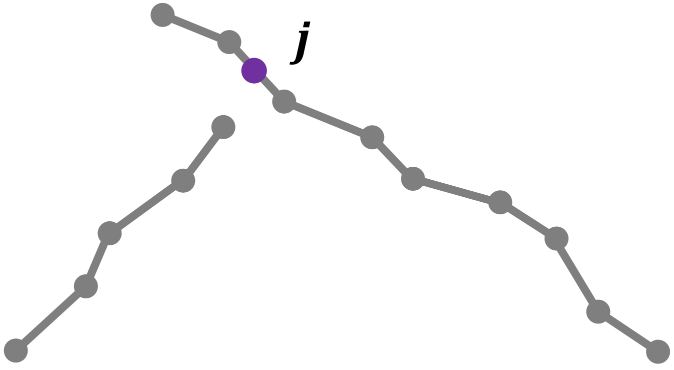
Junction point j in the polyline



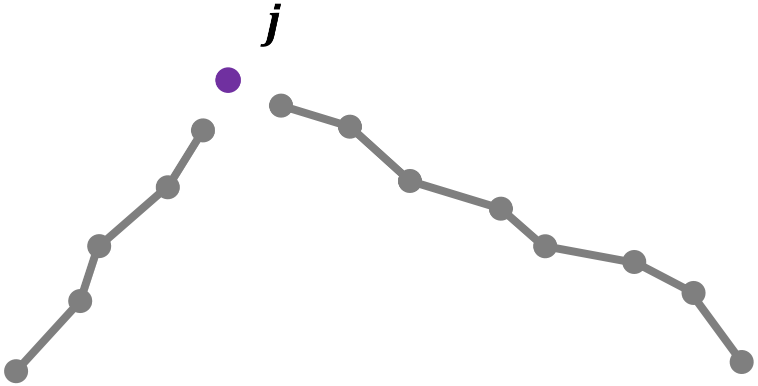
Junction point j out side the polyline

Methodology: the topology of breaklines

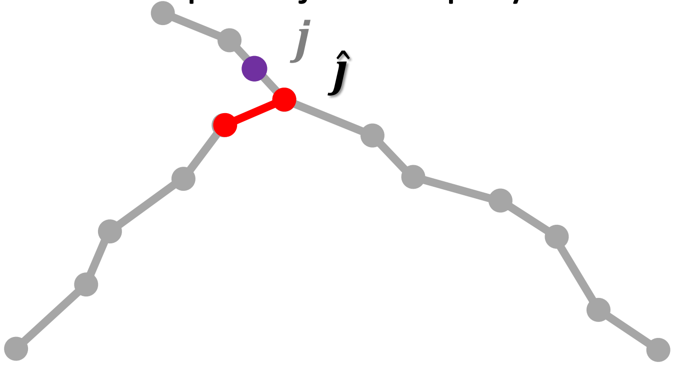
– connecting two breaklines



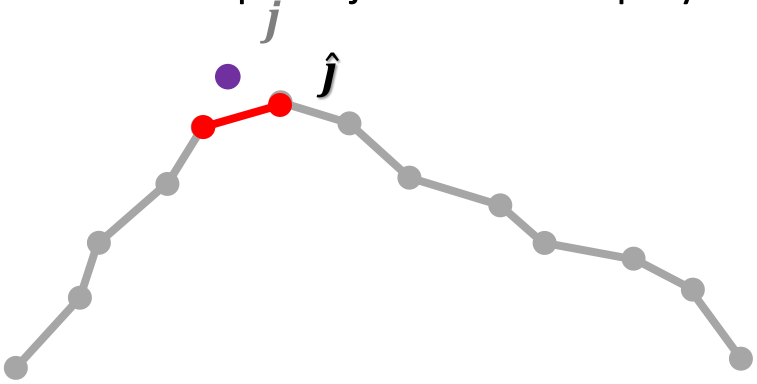
Junction point j in the polyline



Junction point j out side the polyline

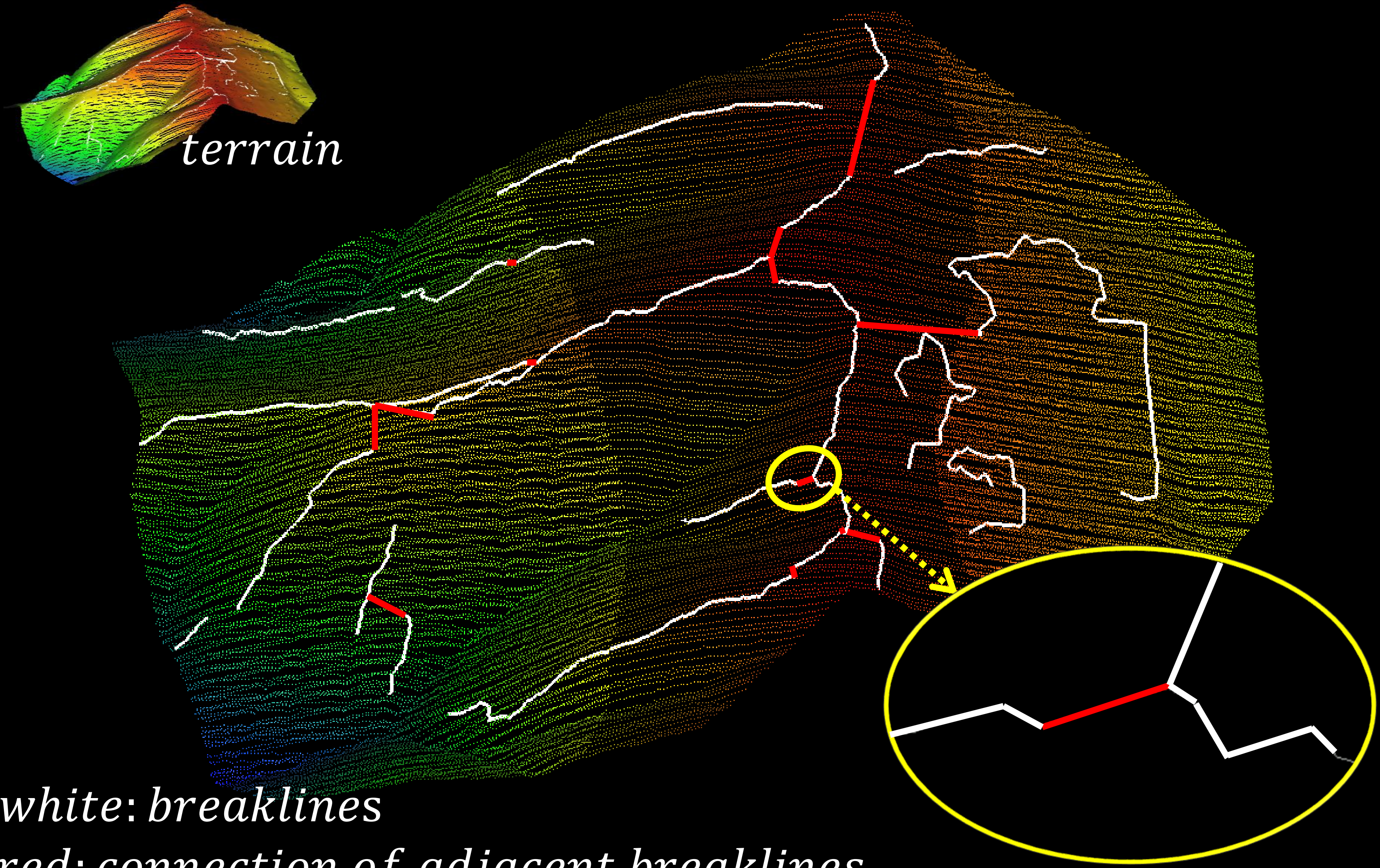


connection &
estimated junction $\hat{j}$



connection &
estimated junction $\hat{j}$

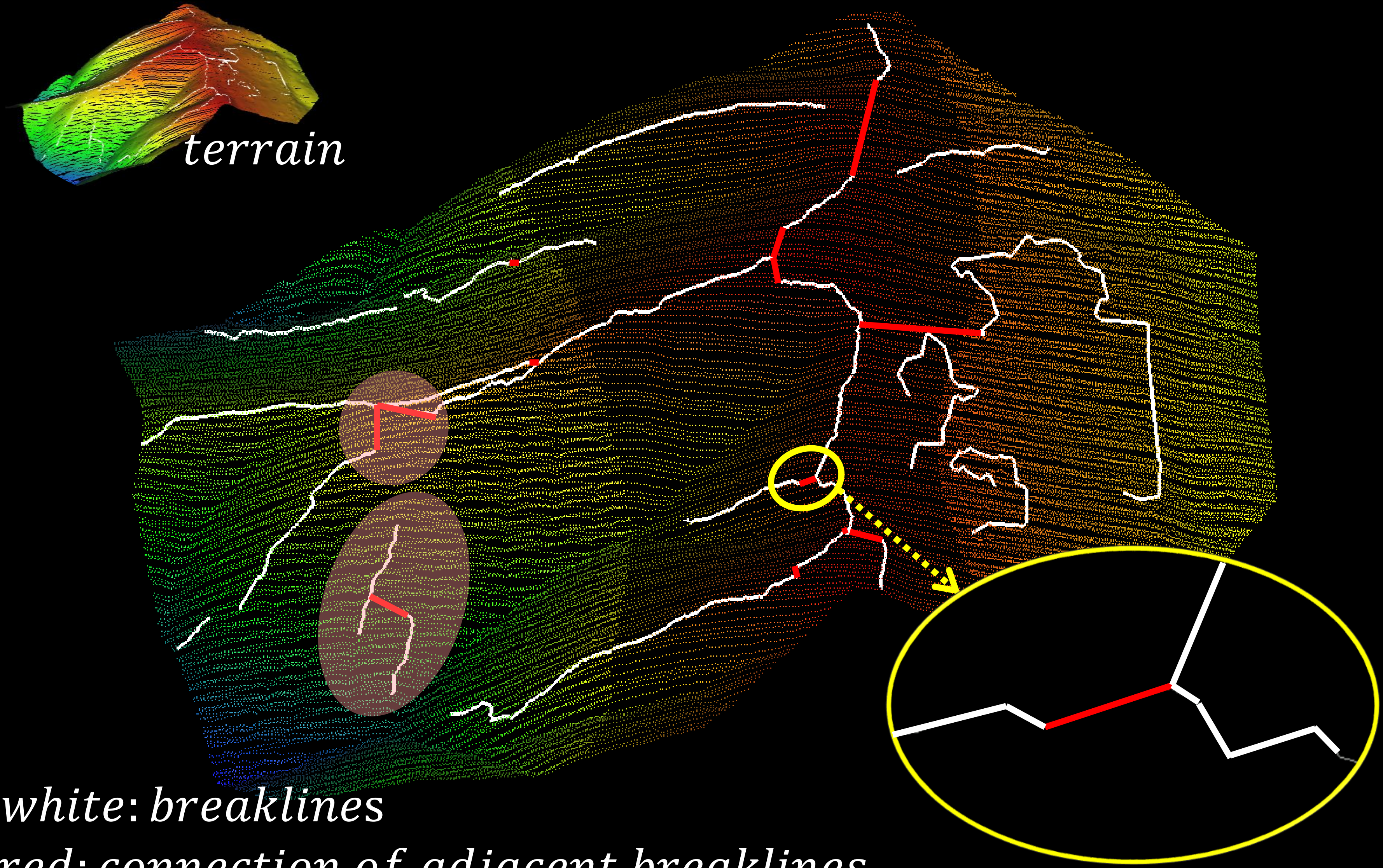
Methodology: the topology of breaklines



white: *breaklines*

red: *connection of adjacent breaklines*

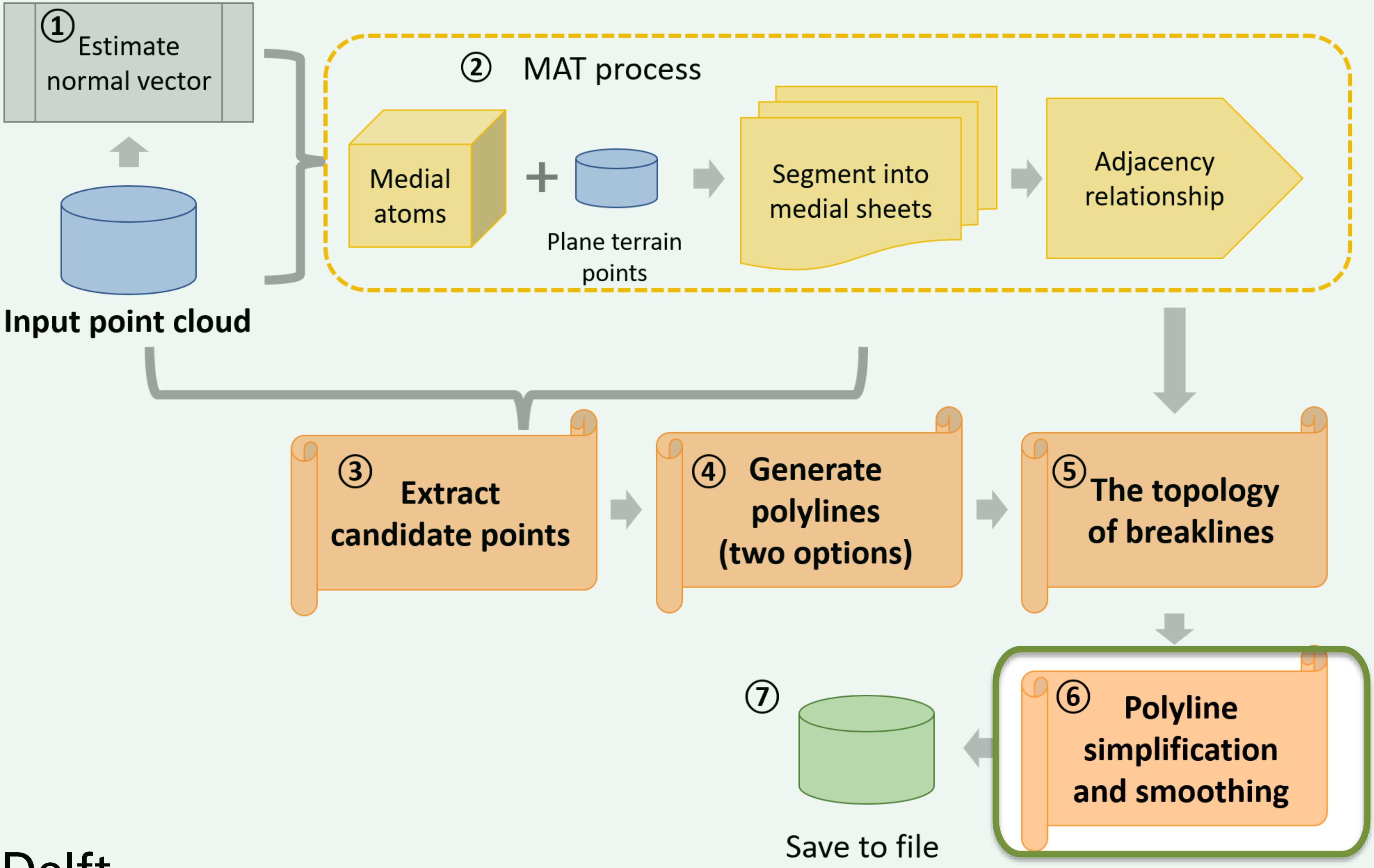
Methodology: the topology of breaklines



white: *breaklines*

red: *connection of adjacent breaklines*

Methodology overview



① Estimate normal vector

Input point cloud

② MAT process

Medial atoms

+

Plane terrain points

Segment into medial sheets

Adjacency relationship

③ Extract candidate points

④ Generate polylines (two options)

⑤ The topology of breaklines

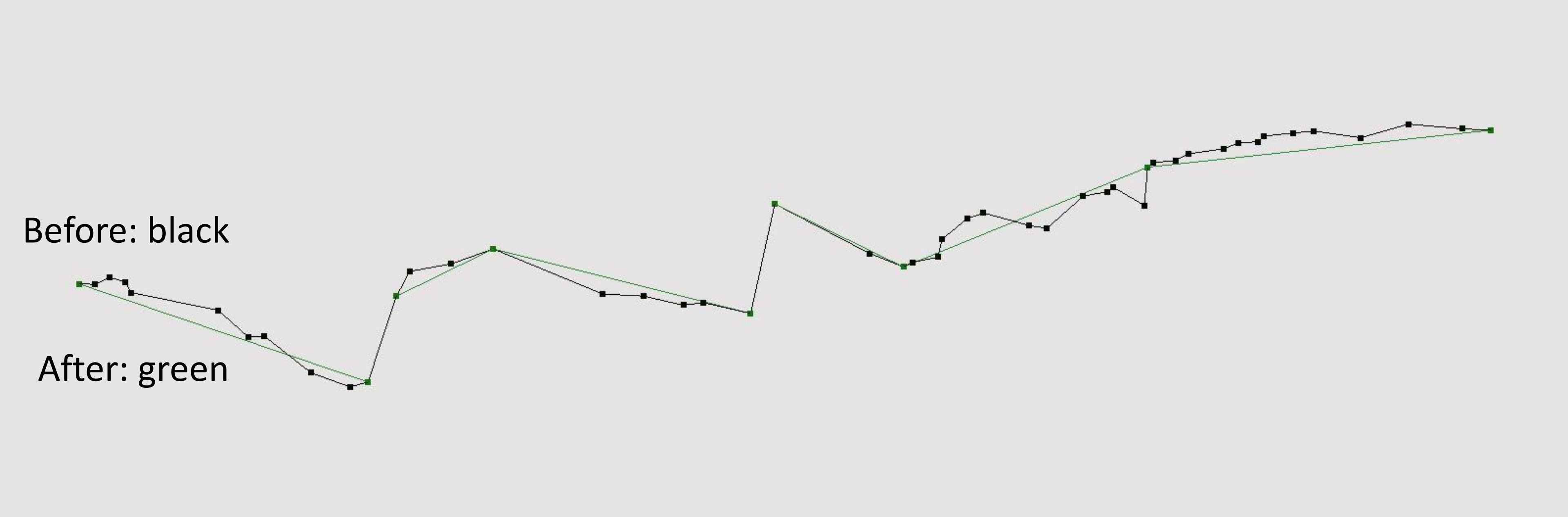
⑥ Polyline simplification and smoothing

⑦

Save to file

Methodology: polyline simplification and smoothing

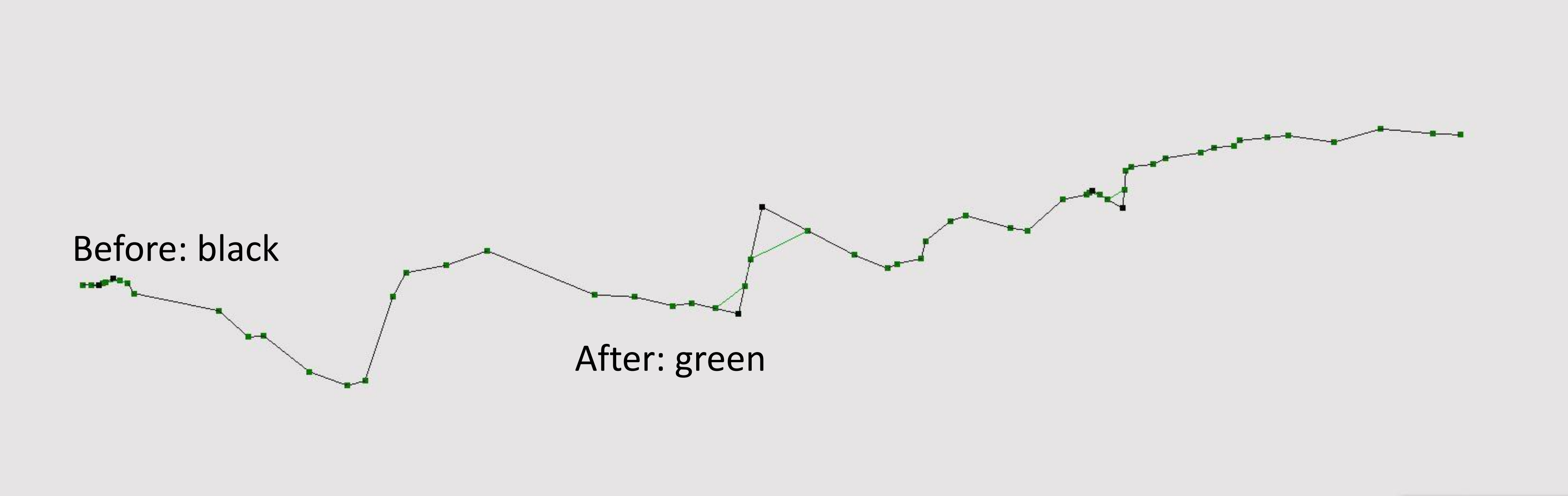
- Simplification: remove less important points*



*by Visvalingam’s algorithm

Methodology: polyline simplification and smoothing

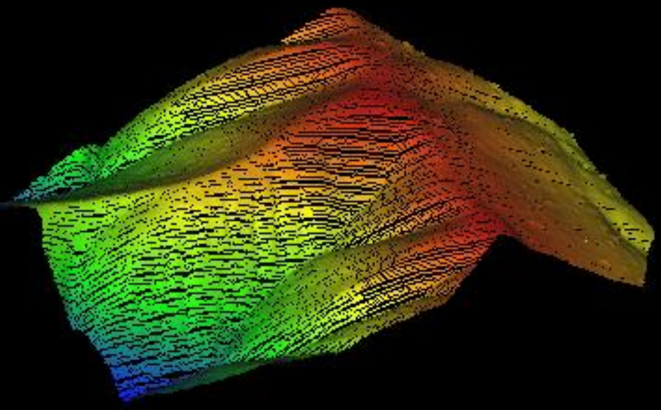
– Smoothing: adding points to remove sharp angle



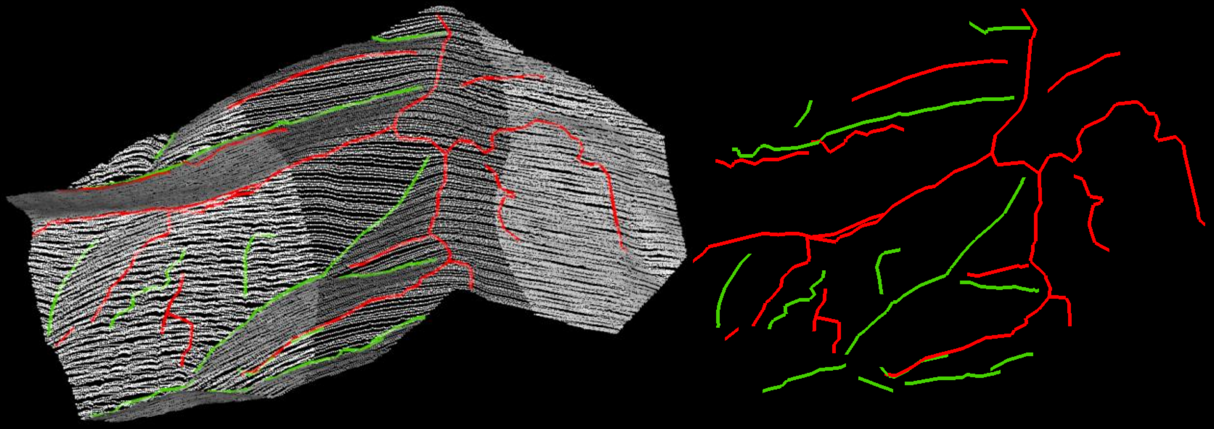
Result & analysis

Result1 (red ridge; green valley)

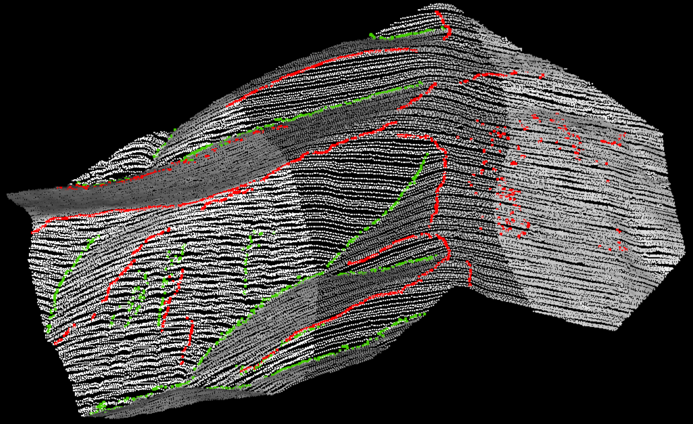
Point cloud



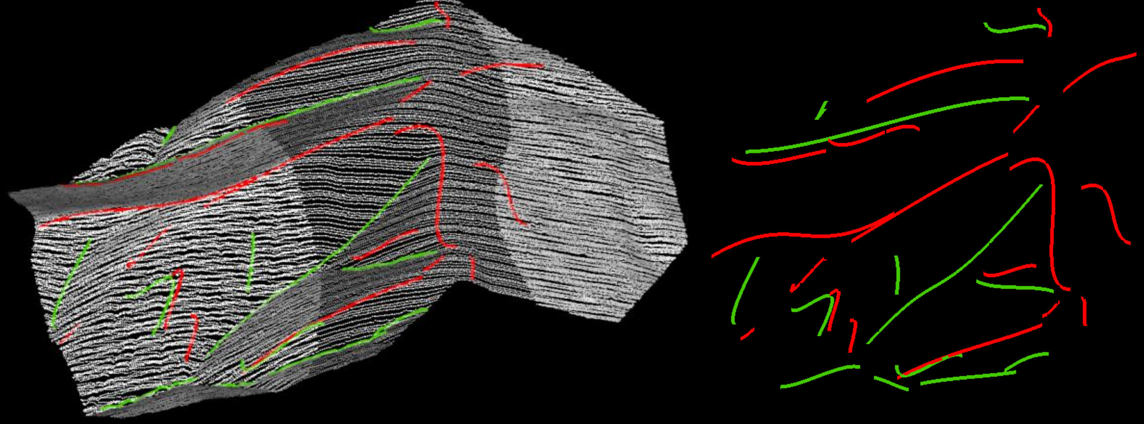
Breaklines with graph theory



Candidate points

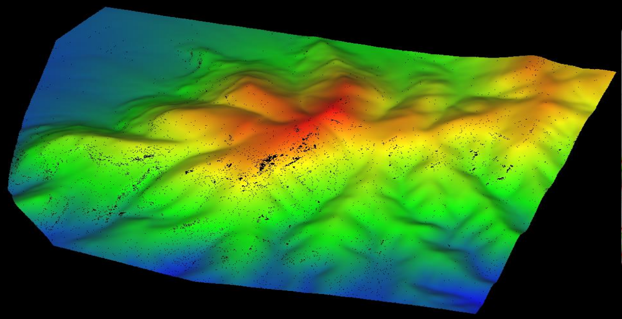


Breaklines with polynomial fit

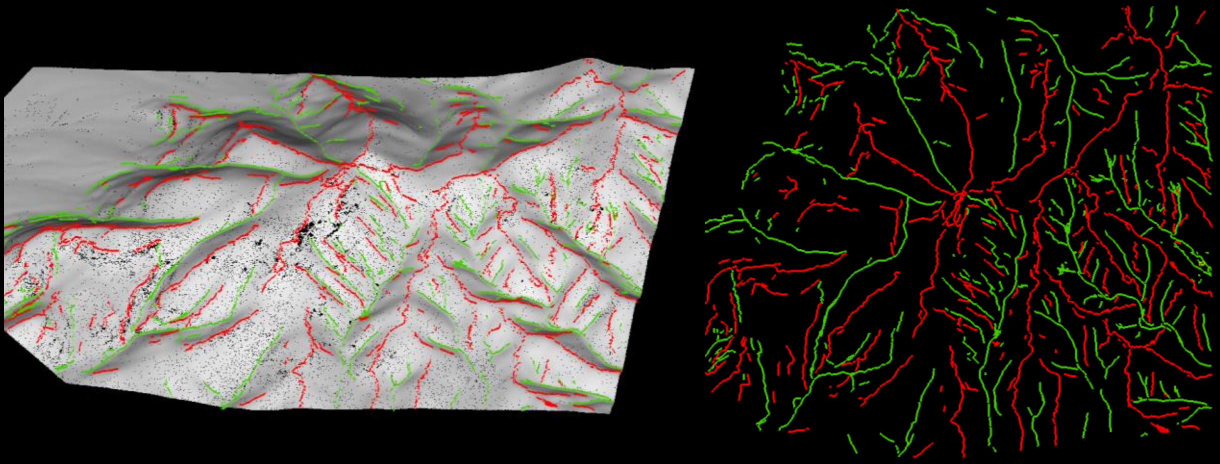


Result2 (red ridge; green valley)

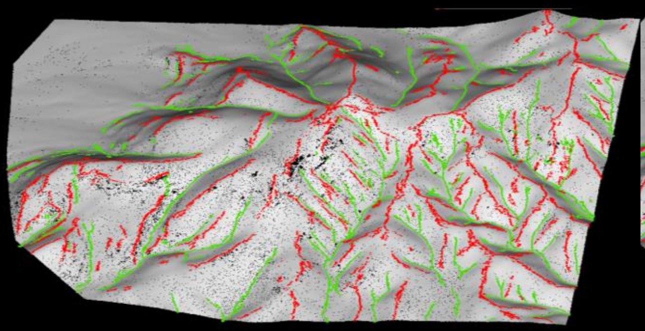
Point cloud



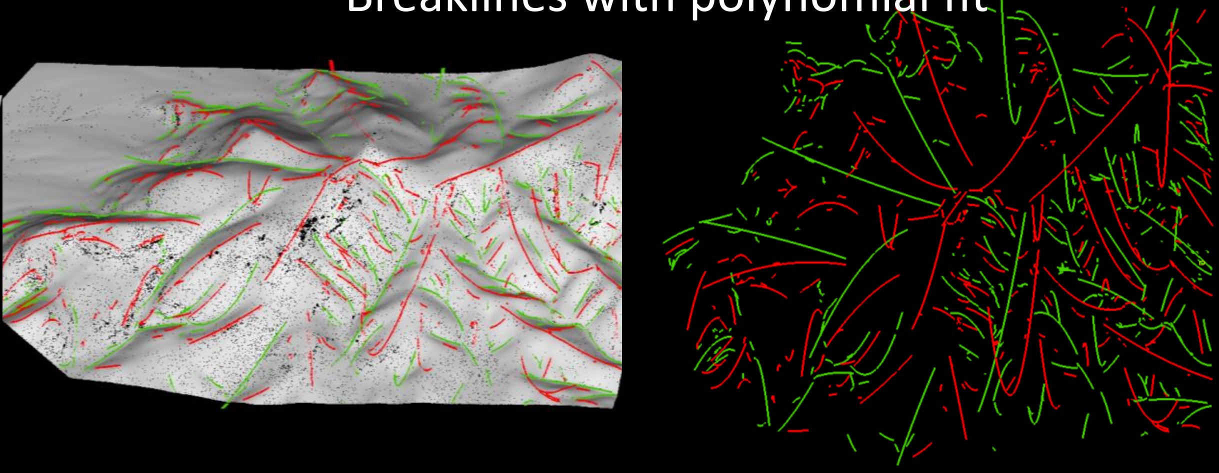
Breaklines with graph theory



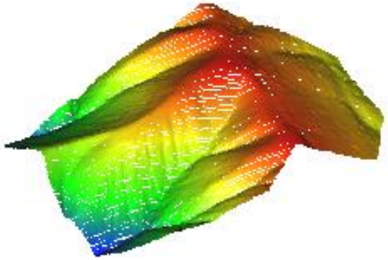
Candidate points

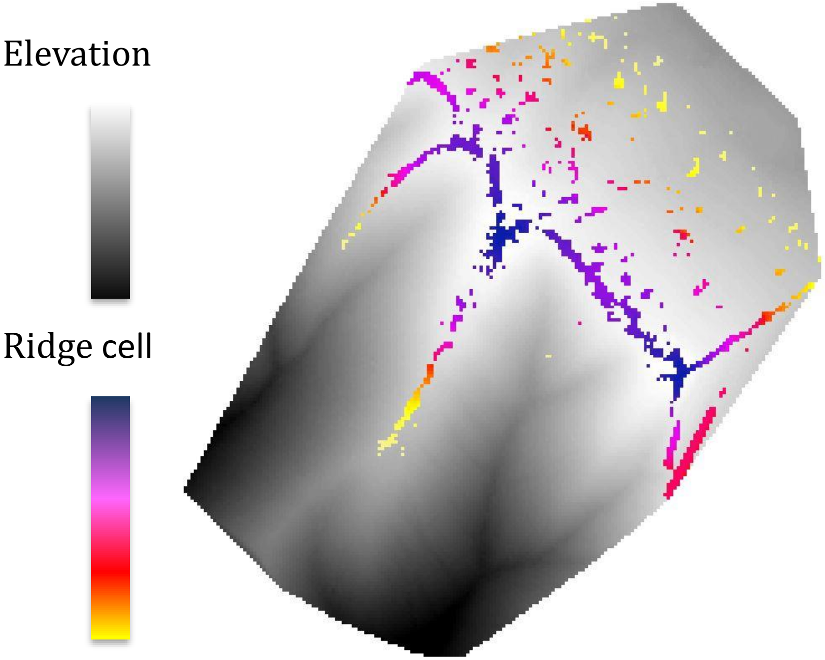


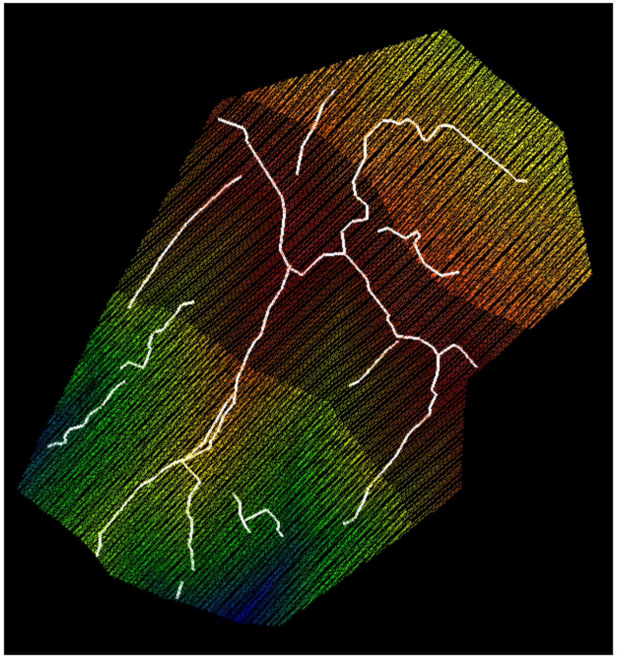
Breaklines with polynomial fit



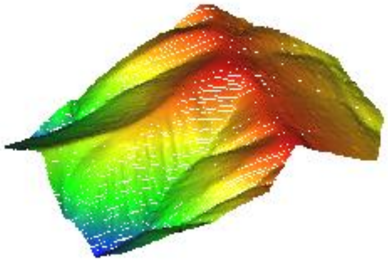
Comparing with ArcMap: ridge

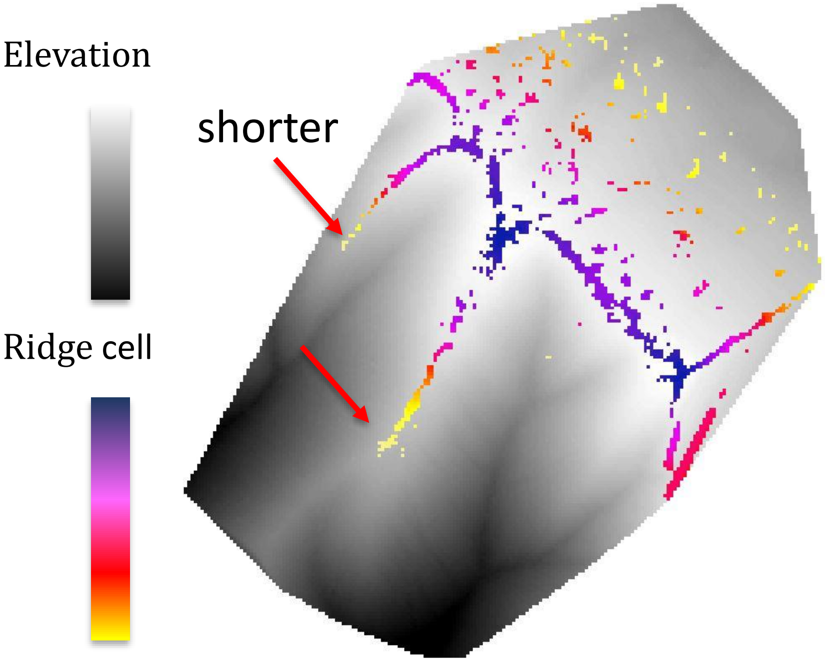




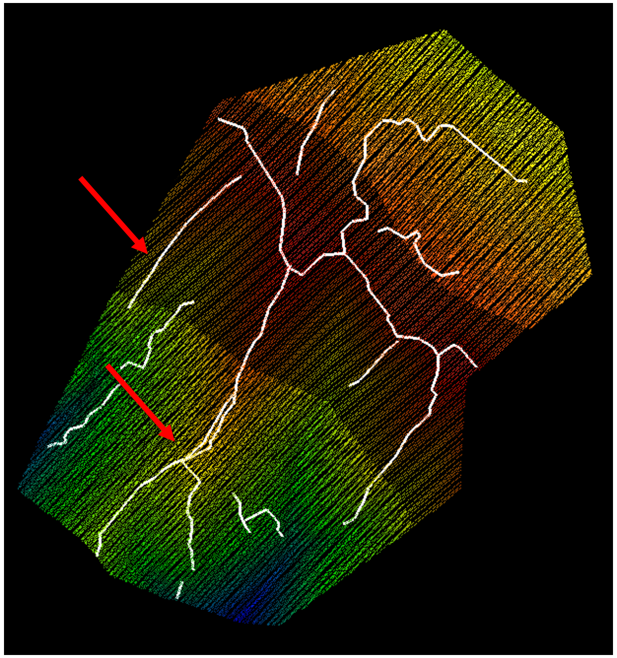


Comparing with ArcMap: ridge



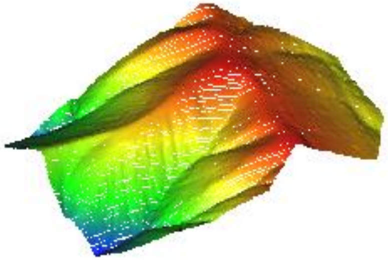


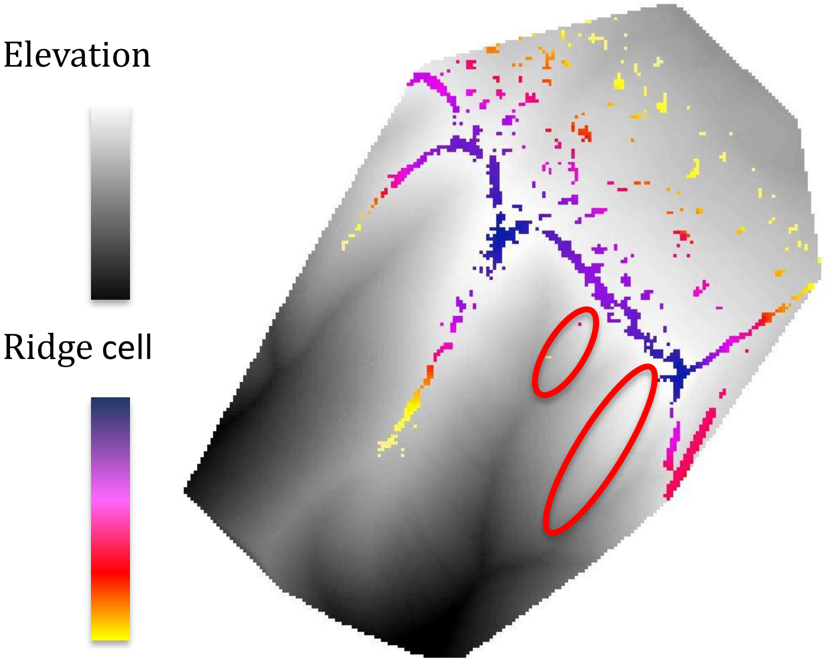
Cells → points
Incomplete



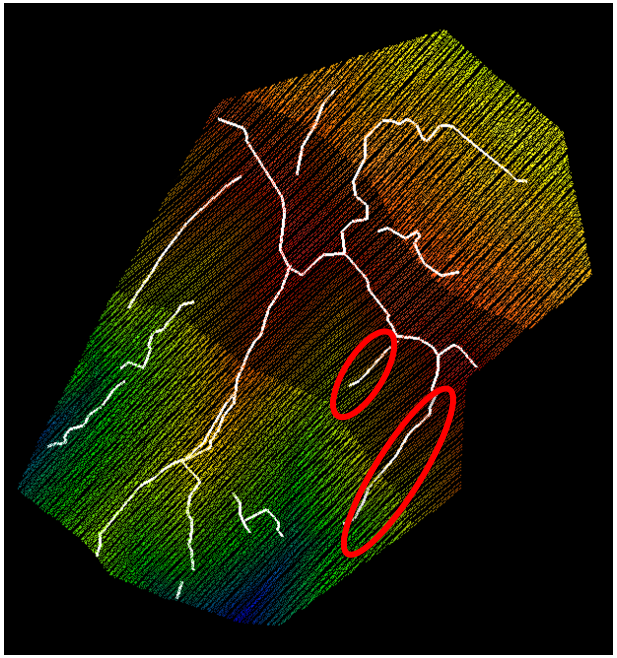
3D polylines + topology √
Complete √

Comparing with ArcMap: ridge



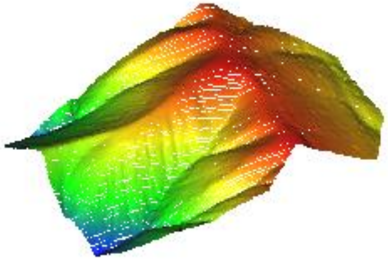


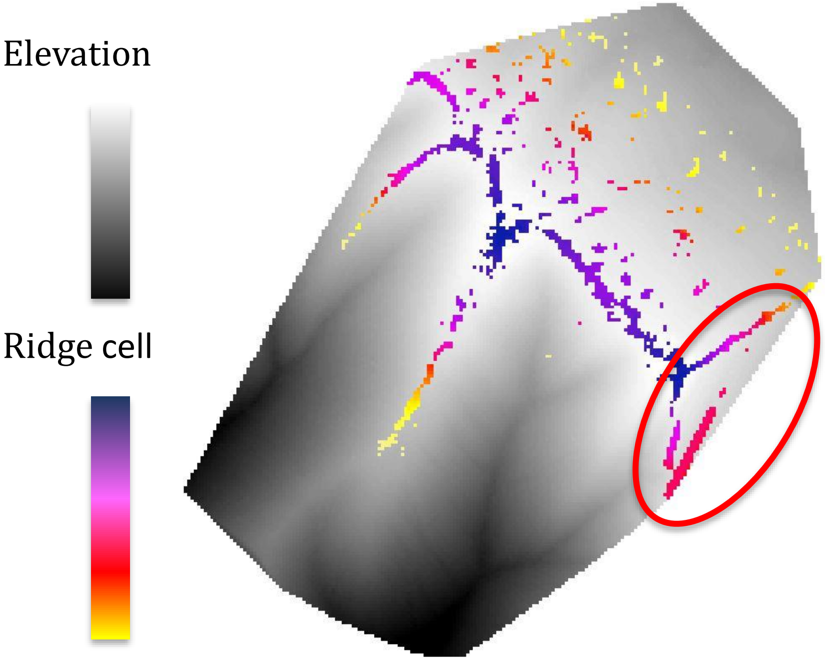
Cells → points
Incomplete
Missing ridges

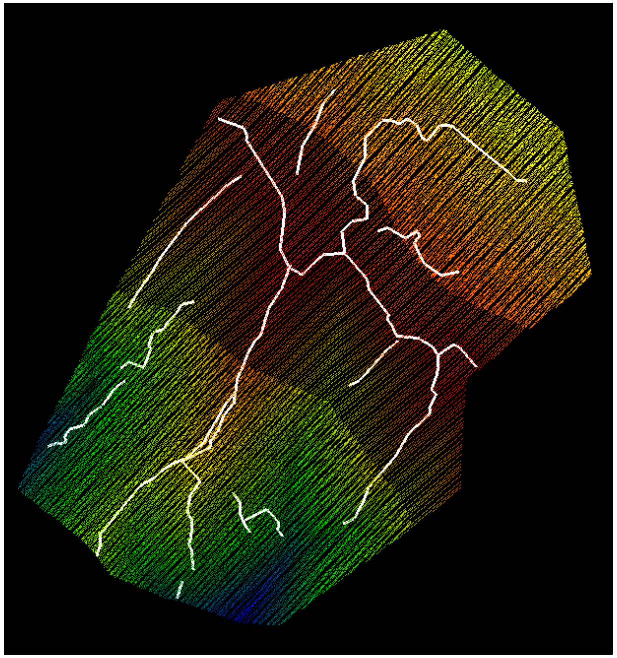


3D polylines + topology √
Complete √
Missing small ridges

Comparing with ArcMap: ridge







Cells → points
Incomplete
Missing ridges
Effected by spatial interpolation

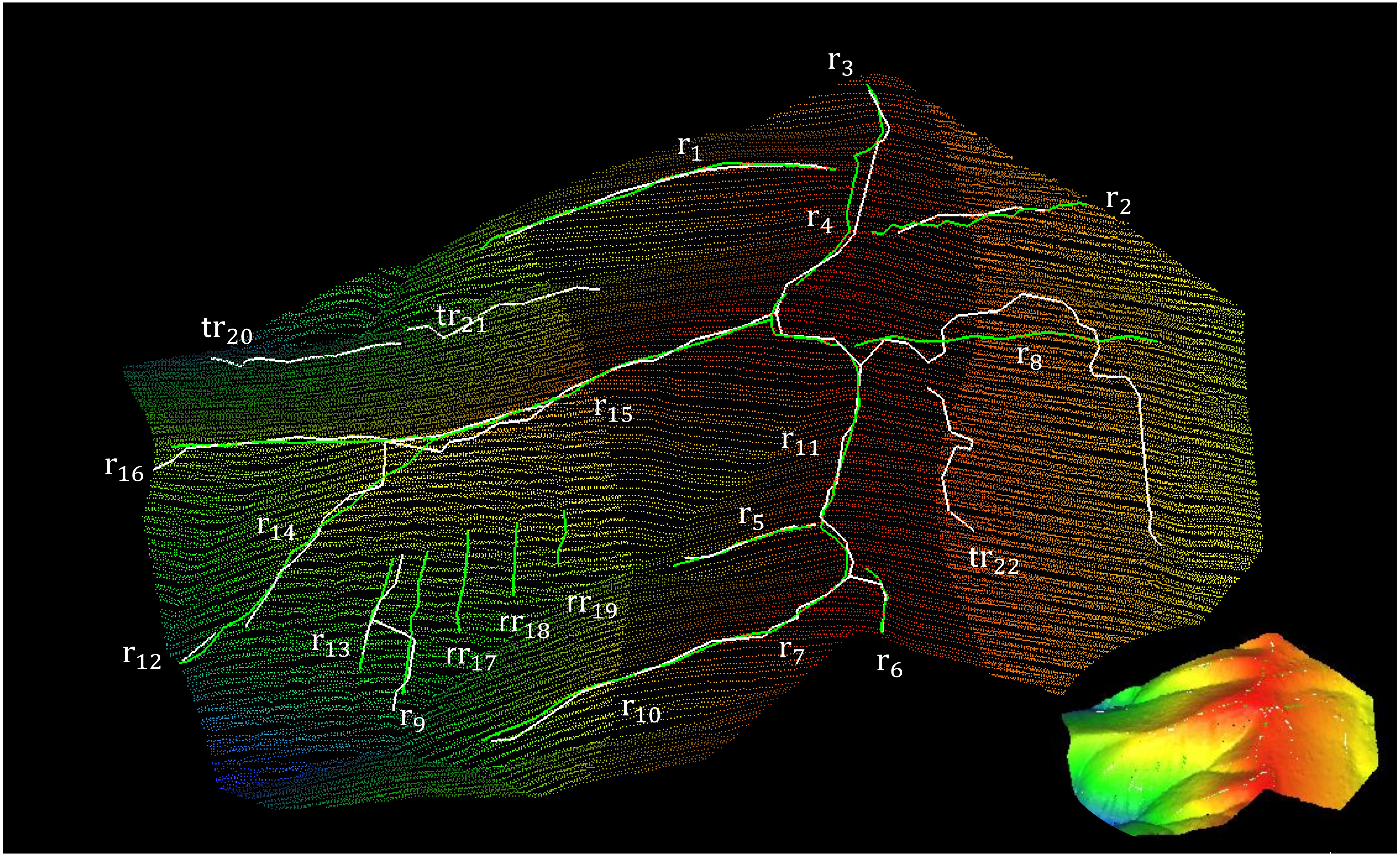
3D polylines + topology √
Complete √
Missing small ridges

Validation:

- If the breaklines are correctly identified?

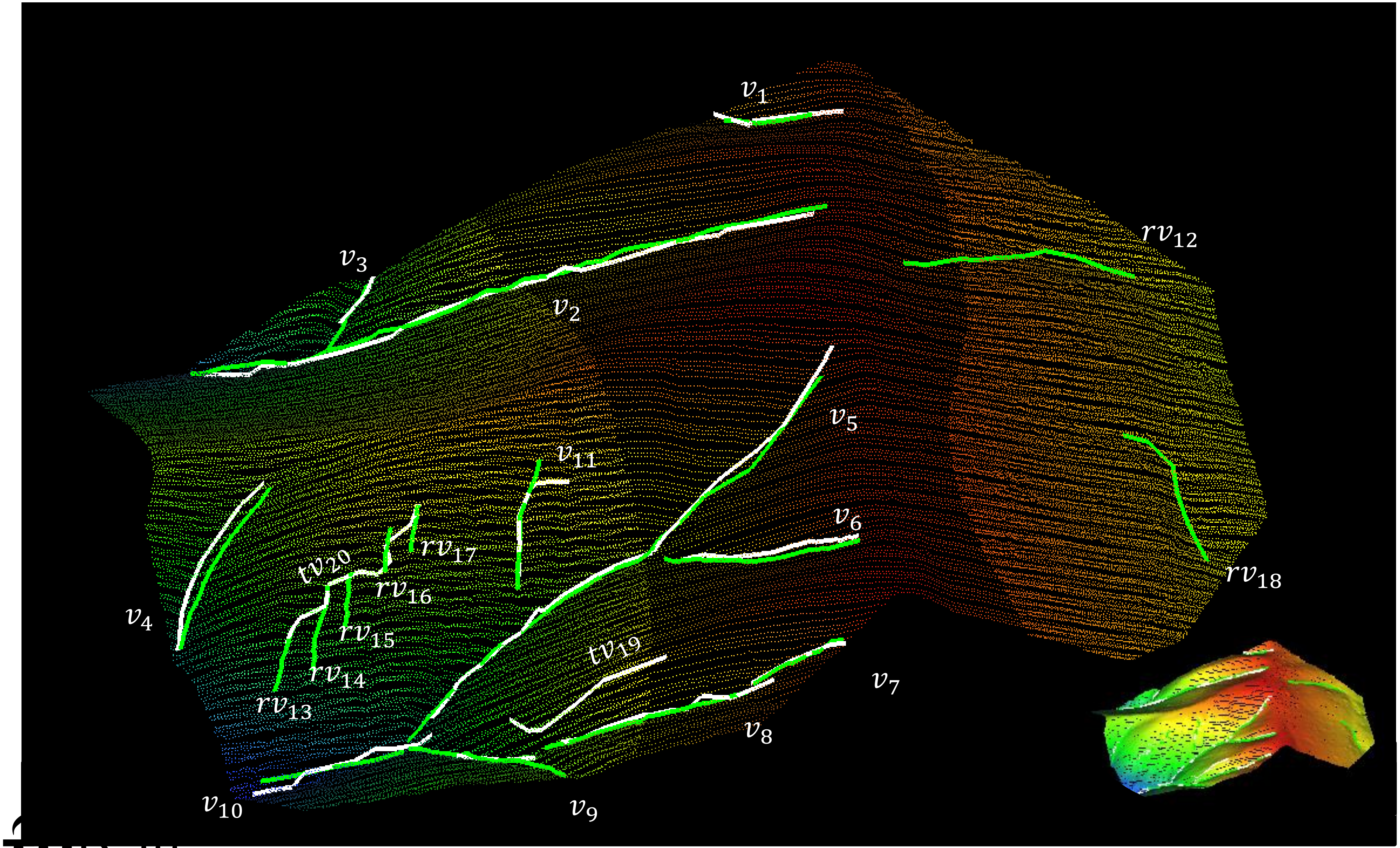
- How is the accuracy and precision of the correctly identified breaklines?

Validation: ridge



Green: reference ridge; white: generated ridges

Validation: valley

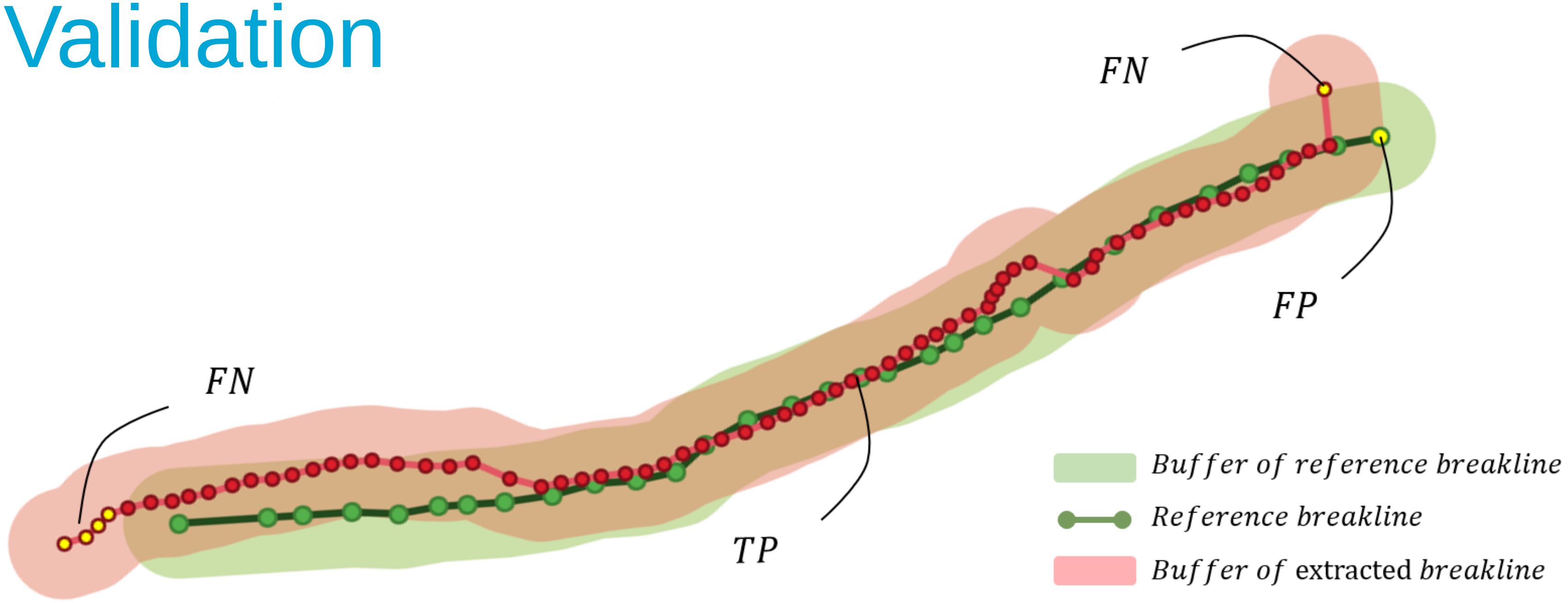


Green: reference valleys; white: generated valleys

Validation

- If the breaklines are correctly identified?
  - 16/19 ridges and 11/13 valleys (27/32 breaklines) are correctly identified

Validation



Accuracy:

$$correctness = \frac{TP}{TP + FP \times scalar} \qquad quality = \frac{TP}{TP + FN + FP \times scalar}$$

$$completeness = \frac{TP}{TP + FN} \qquad \left(scalar = \frac{TP + FN}{NR} \right)$$

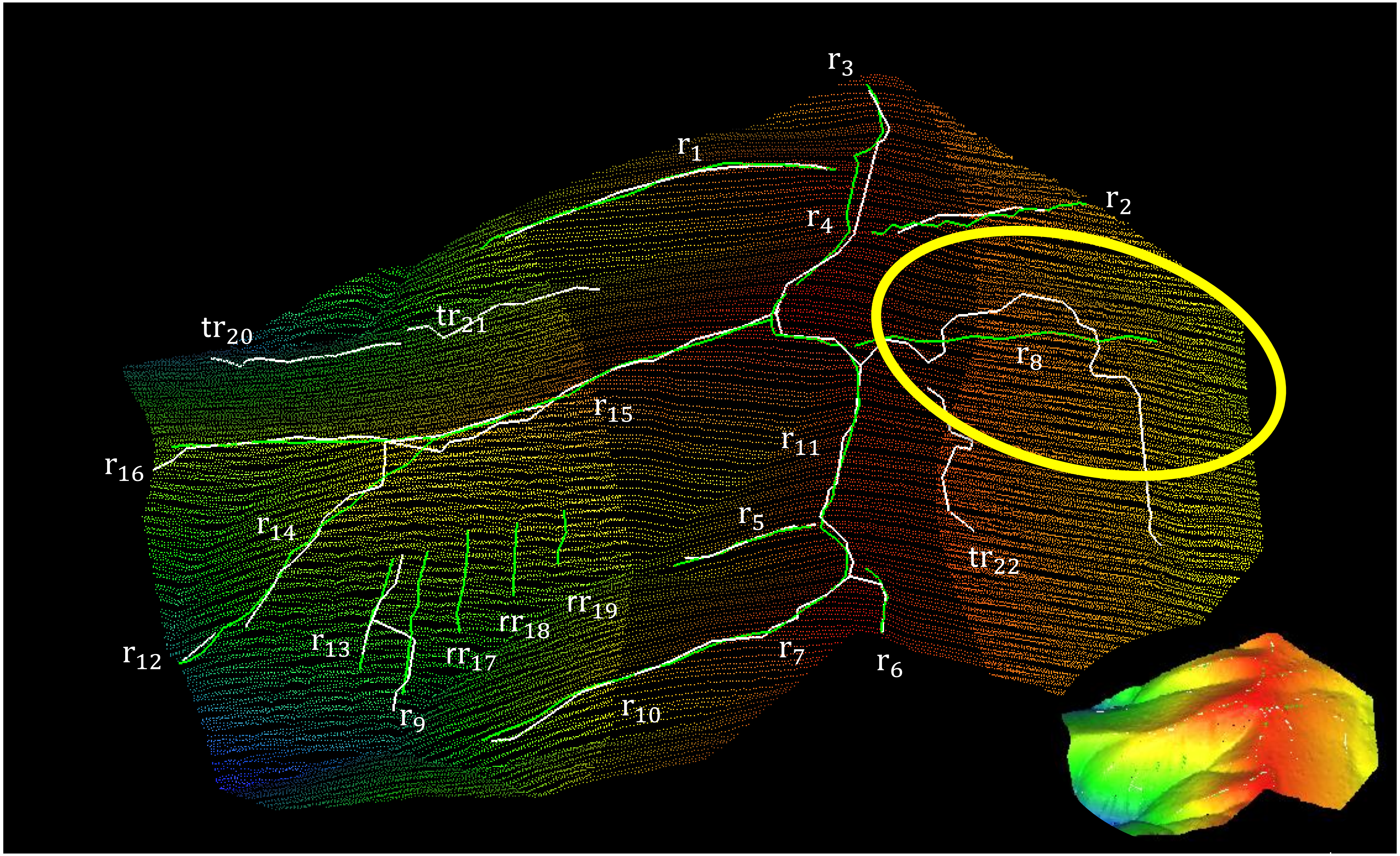
Precision:

$$p_l = \frac{\sum d(v, r_l)}{the\ length\ of\ l} \qquad p_v = \frac{\sum d(v, r_l)}{TP}$$

Validation: ridge

TP pair	scalar	correctness	quality	completeness	polyline precision	vertex precision
r1	100%	100.00%	100.00%	100.00%	0.06	0.46
r2	21.74%	88.46%	88.46%	100.00%	0.09	0.94
r3	85.71%	100.00%	100.00%	100.00%	0.26	0.70
r4	28.57%	100.00%	100.00%	100.00%	0.15	1.44
r5	54.55%	100.00%	100.00%	100.00%	0.10	0.49
r6	44.44%	100.00%	100.00%	100.00%	0.27	1.09
r7	33.33%	100.00%	100.00%	100.00%	0.20	0.78
r8	191.67%	89.87%	68.23%	73.91%	1.23	4.62
r9	66.67%	71.43%	62.50%	83.33%	0.26	1.48
r10	52.38%	100.00%	100.00%	100.00%	0.13	0.85
r11	64.00%	100.00%	100.00%	100.00%	0.18	0.86
r12	40.00%	100.00%	100.00%	100.00%	0.07	0.42
r13	83.33%	100.00%	100.00%	100.00%	0.19	0.84
r14	80.00%	93.22%	85.94%	91.67%	0.20	0.99
r15	100%	100.00%	61.54%	61.54%	0.18	1.10
r16	76.47%	100.00%	92.31%	92.31%	0.16	0.93
average		96%	91.19%	93.92%	0.23	1.12

Validation: ridge



Green: reference ridge; white: generated ridges

Validation: valley

TP pair	scalar	correctness	quality	completeness	polyline precision	vertex precision
v1	83.33%	100%	80.00%	80.00%	0.32	1.37
v2	75.76%	100%	100%	100%	0.25	1.28
v3	30.00%	90.91%	90.91%	100%	0.17	1.16
v4	72.73%	100%	100%	100%	0.36	1.58
v5	73.33%	96.63%	92.38%	95.45%	0.18	0.94
v6	146.15%	100%	100%	100%	1.03	2.17
v7	44.44%	100%	100%	100%	0.13	0.62
v8	76.47%	100%	84.62%	84.62%	0.21	0.78
v9	17.65%	77.27%	77.27%	100%	0.06	0.69
v10	100.00%	100%	88.89%	88.89%	0.49	1.68
v11	66.67%	100%	83.33%	83.33%	0.05	0.25
average		97%	90.67%	93.84%	0.30	1.14

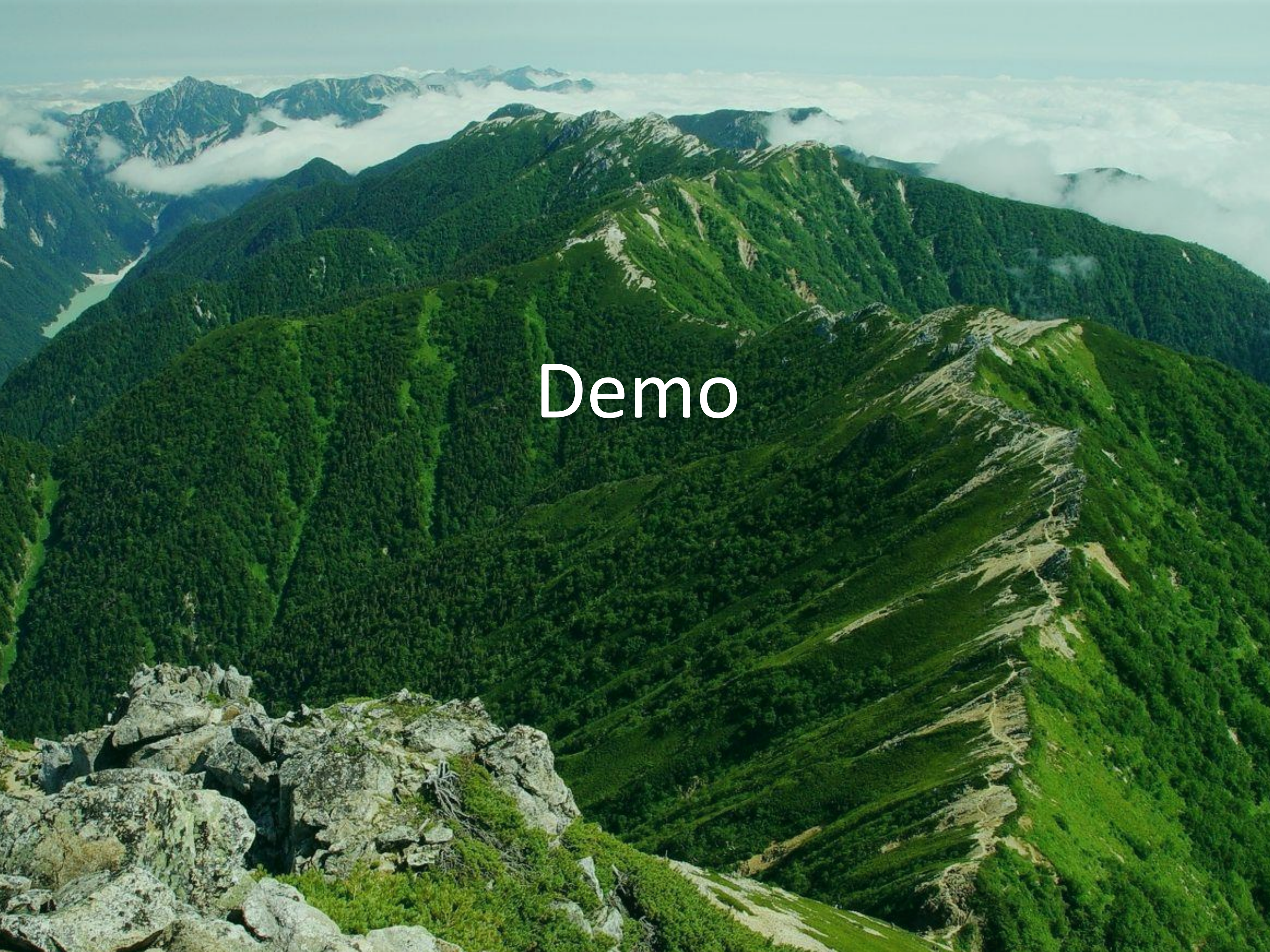
Validation

- If the breaklines correctly identified?
  - 16/19 ridges and 11/13 valleys (27/32 breaklines) are correctly identified
- How is the accuracy and precision of the correctly identified breaklines?
  - accurate and precise

	correctness	quality	completeness	polyline precision	vertex precision
ridge	96%	91.19%	93.92%	0.23	1.12
valley	97%	90.67%	93.84%	0.30	1.14

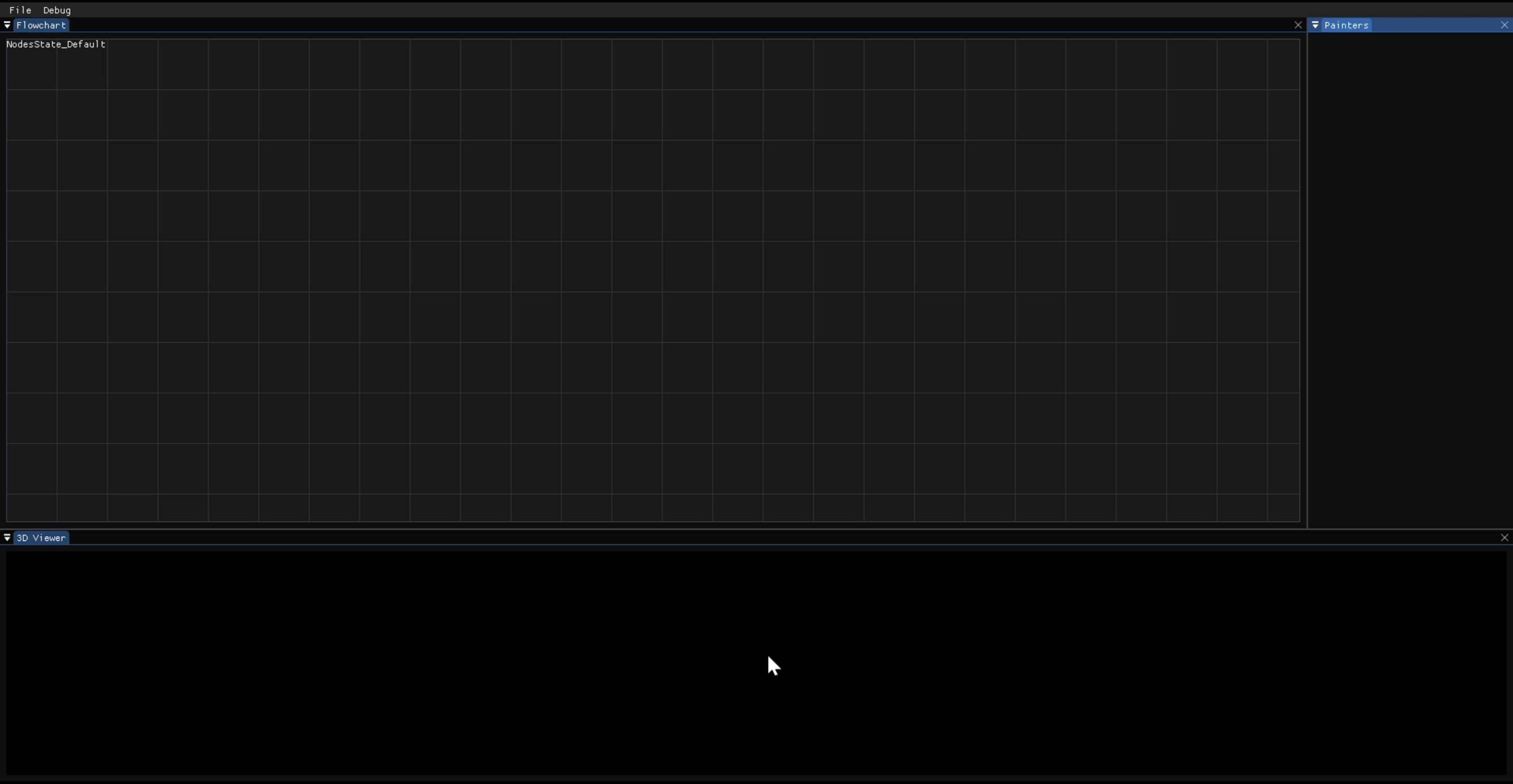
Point density: 2pts/m^2

TUDelft

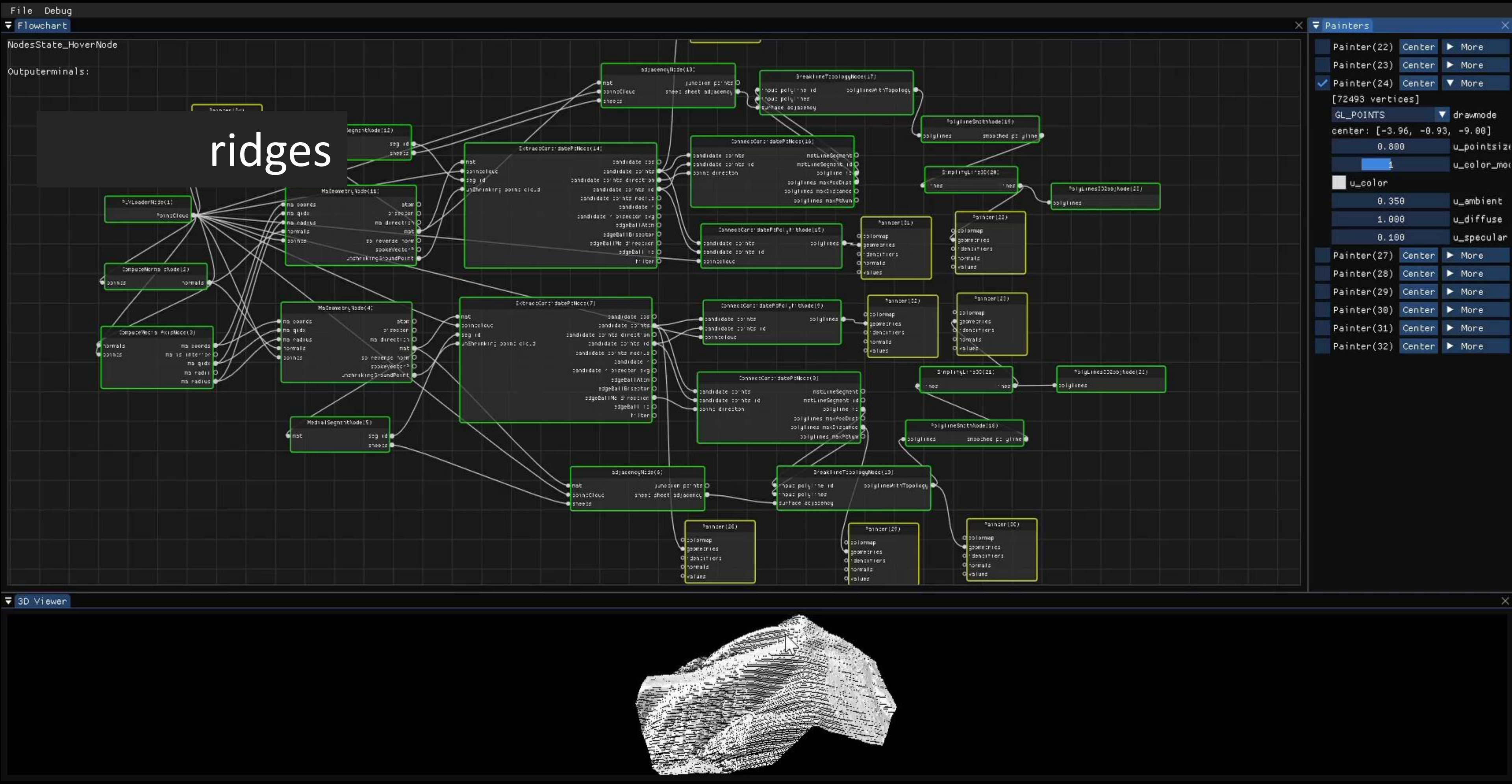


Demo

Demo


File
Debug
Flowchart
NodesState_Default
Painters
3D Viewer

Demo (ridge: red; valley: green)



ridges

Painters

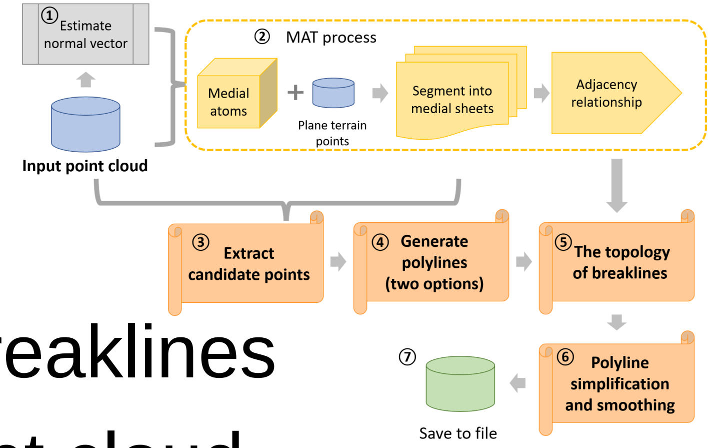
□ Painter(22)	Center	▶ More
□ Painter(23)	Center	▶ More
✓ Painter(24)	Center	▼ More

[72493 vertices]
GL_POINTS ▼ drawmode
center: [-3.96, -0.93, -9.00]
0.800 u_pointsize
1 u_color_mod
□ u_color
0.350 u_ambient
1.000 u_diffuse
0.100 u_specular

□ Painter(27)	Center	▶ More
□ Painter(28)	Center	▶ More
□ Painter(29)	Center	▶ More
□ Painter(30)	Center	▶ More
□ Painter(31)	Center	▶ More
□ Painter(32)	Center	▶ More

Conclusion

Conclusion



① Estimate normal vector
Input point cloud
② MAT process
Medial atoms
+
Plane terrain points
Segment into medial sheets
Adjacency relationship
③ Extract candidate points
④ Generate polylines (two options)
⑤ The topology of breaklines
⑥ Polyline simplification and smoothing
⑦
Save to file

- A new method to generate breaklines
- Directedly processing on point cloud
- long breaklines:
  - Correctly identified;
  - Accurate and precise
- small breaklines & planar area:
  - missing identified or incorrectly identified
  - topology error

Future work

- parameters:
  - User defined → estimate parameters
    →better result & automatically
  - Global → adapt to different sheet
    → eliminate error in small breaklines
- Topology
  - The junction points & connection
    → require better estimation
- Validation on more datasets

Tools and datasets used

- Language: C++
- Libraries: Geoflow, boost, CGAL, eigen, spline;
- Input data
  - Mountain: *OpenTopography* LiDAR point cloud data

- Source code: https://github.com/qq2012/geoflow-nodes

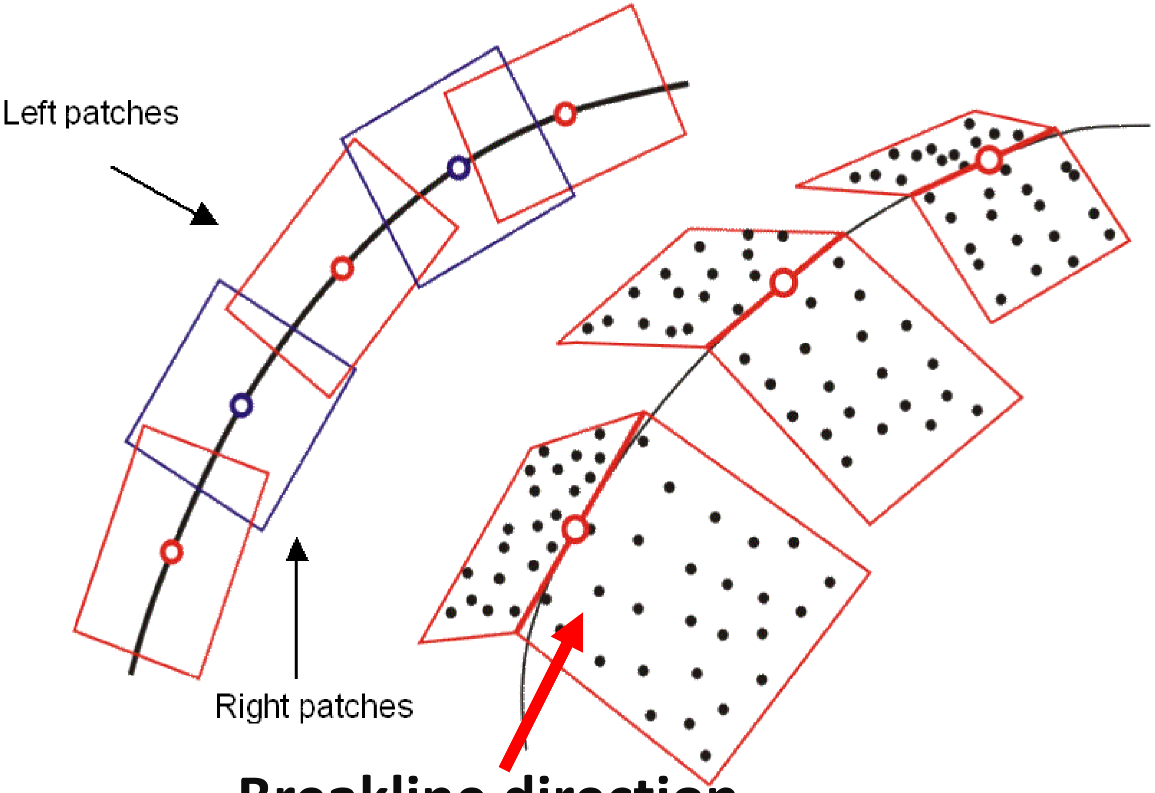
MAT : https://github.com/tudelft3d/geoflow-nodes
Boost: https://www.boost.org
CGAL: https://www.cgal.org
Eigen: http://eigen.tuxfamily.org
Spline: https://github.com/chen0040/cpp-spline
OpenTopography: https://opentopography.org/start

Thank you ^-^

Questions?

- Why Medial Axis Transform (MAT)?
  - Fully 3D, directly processing on point cloud
  - Provide useful geometry features
    - the radius corresponds to the curvature,
  - Medial sheet indicates the breaklines
    - interior & exterior → ridge & valley
    - One sheet indicates one breakline
  - Provide adjacency graph to estimate the topology of the breaklines

Existing method: Point cloud



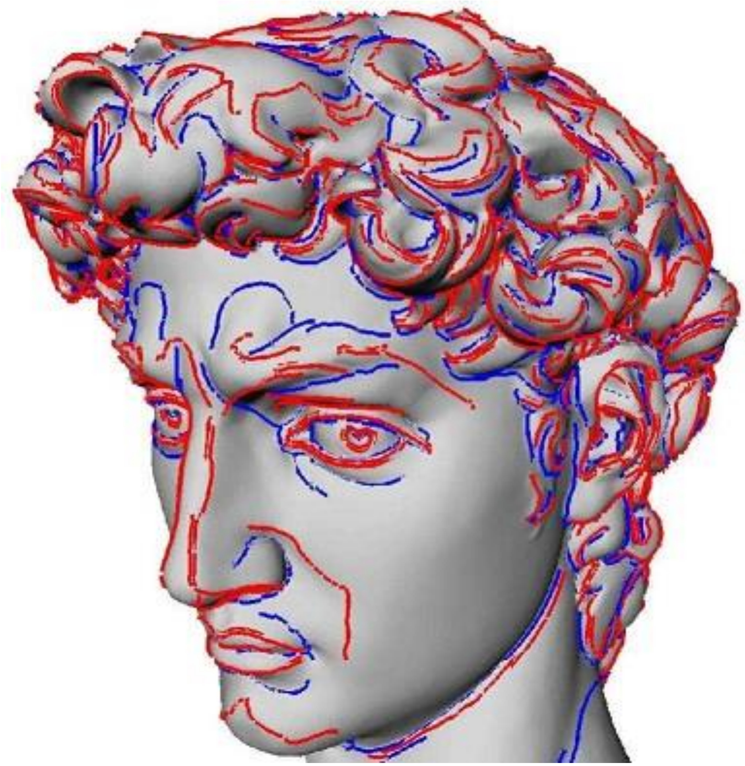
Breakline direction

- Need manual intervention for every breakline
- can not handle branching

Briese, 2004

Existing method: Mesh

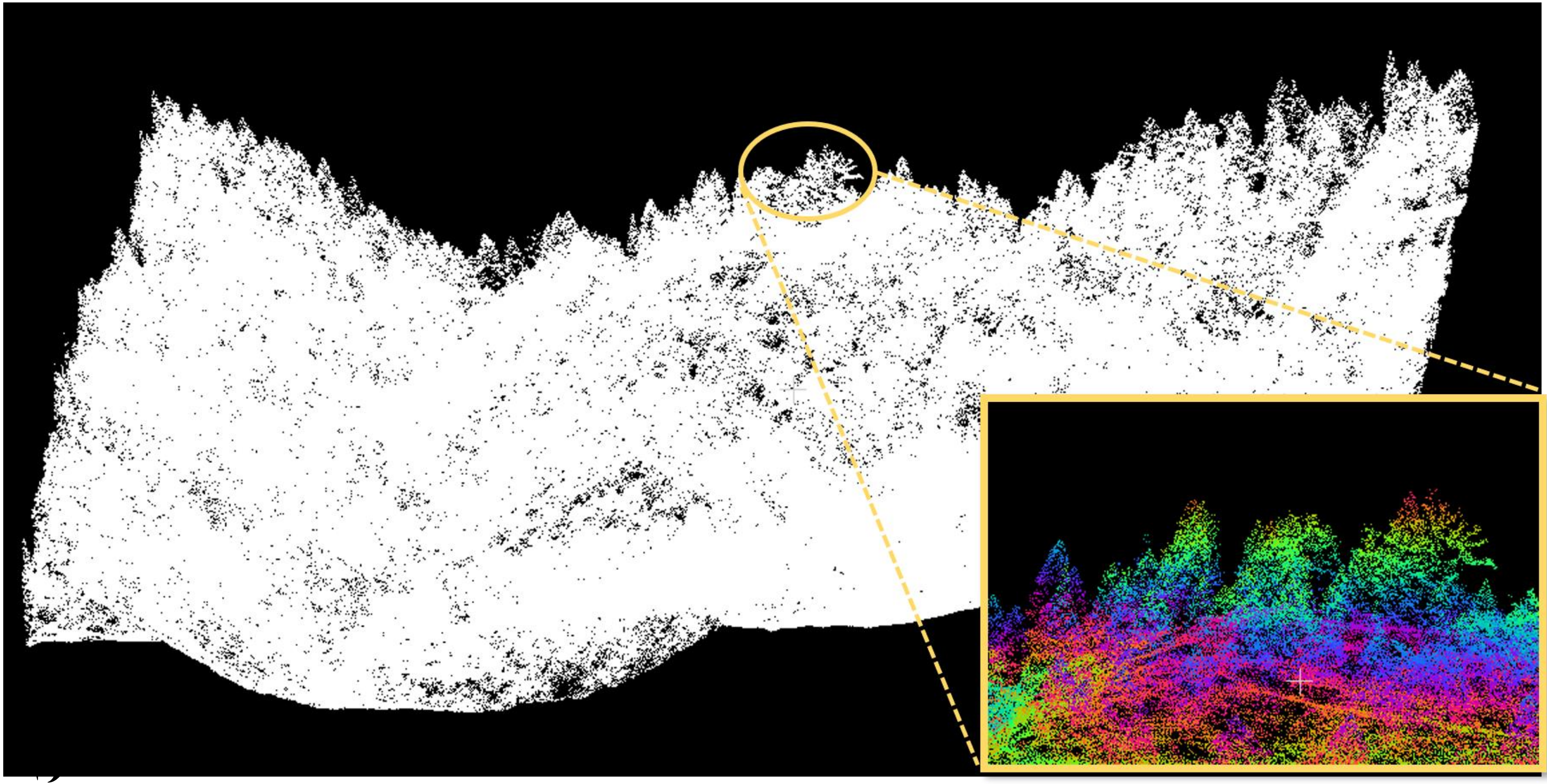
- Input: TIN mesh
- Estimate curvature for each vertices



Cgal: https://dwz.cn/7LwknapW

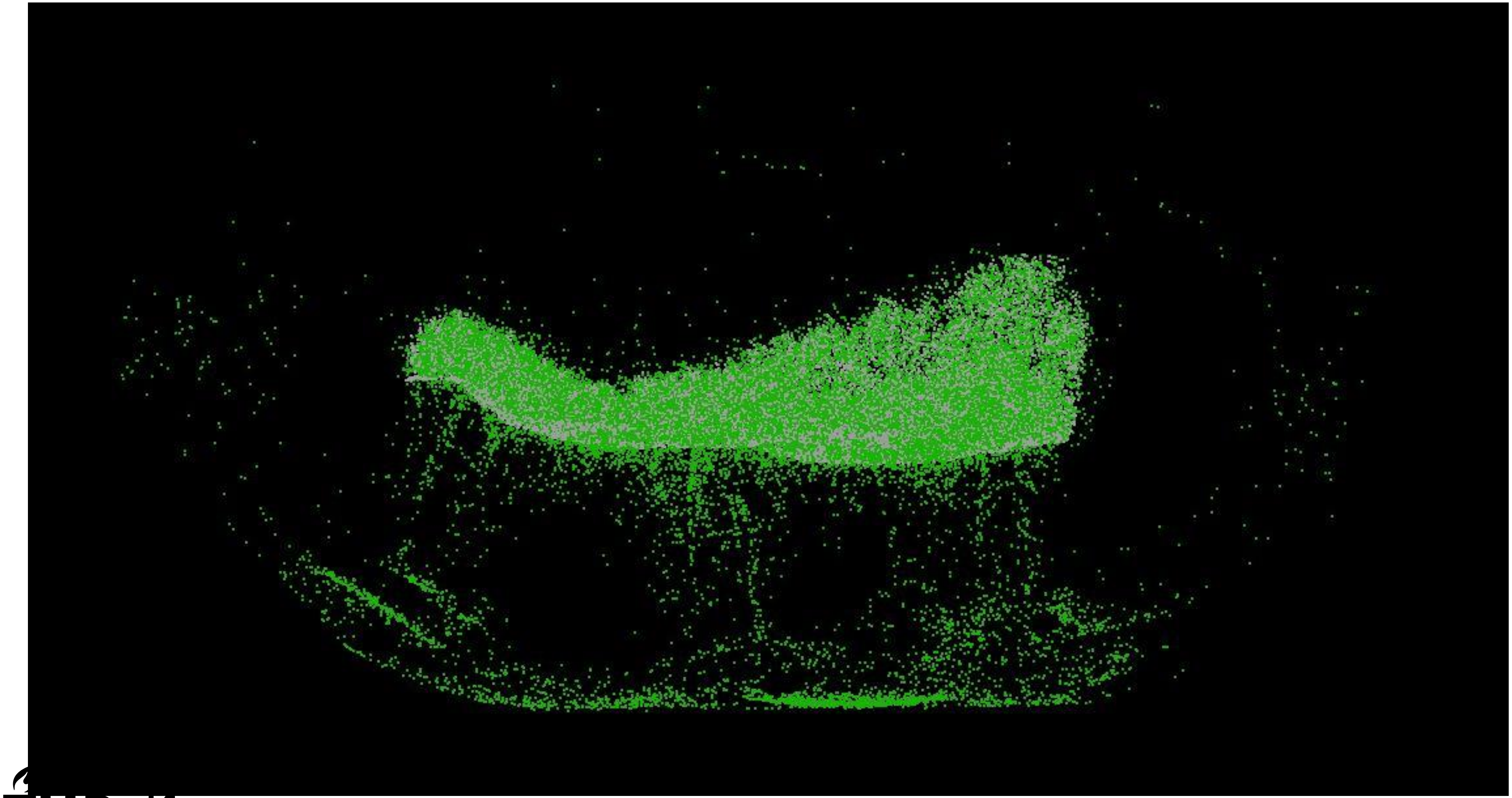
Scope

– Point cloud contains trees will not be considered



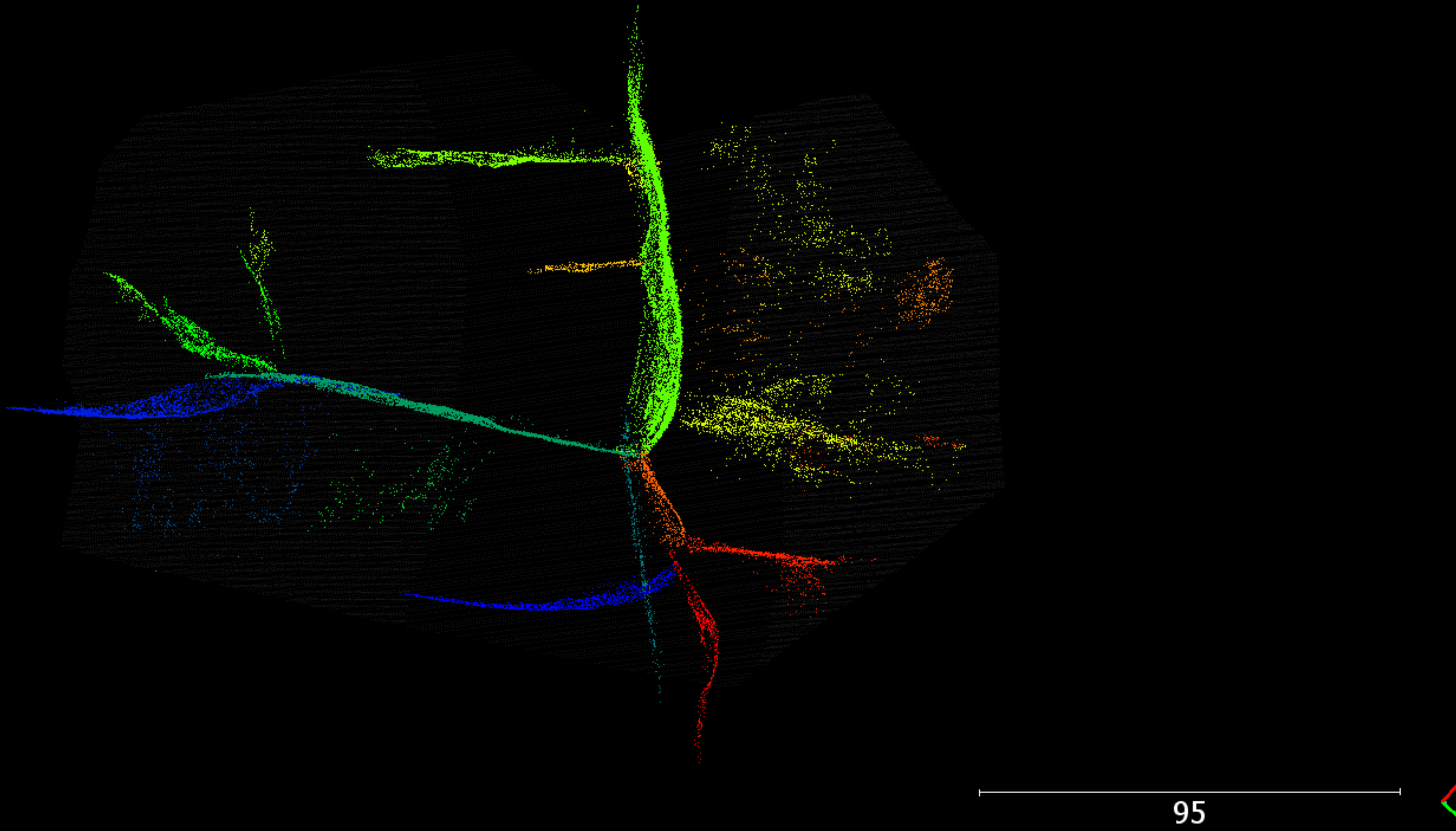
Scope

- Point cloud contains trees will not be considered

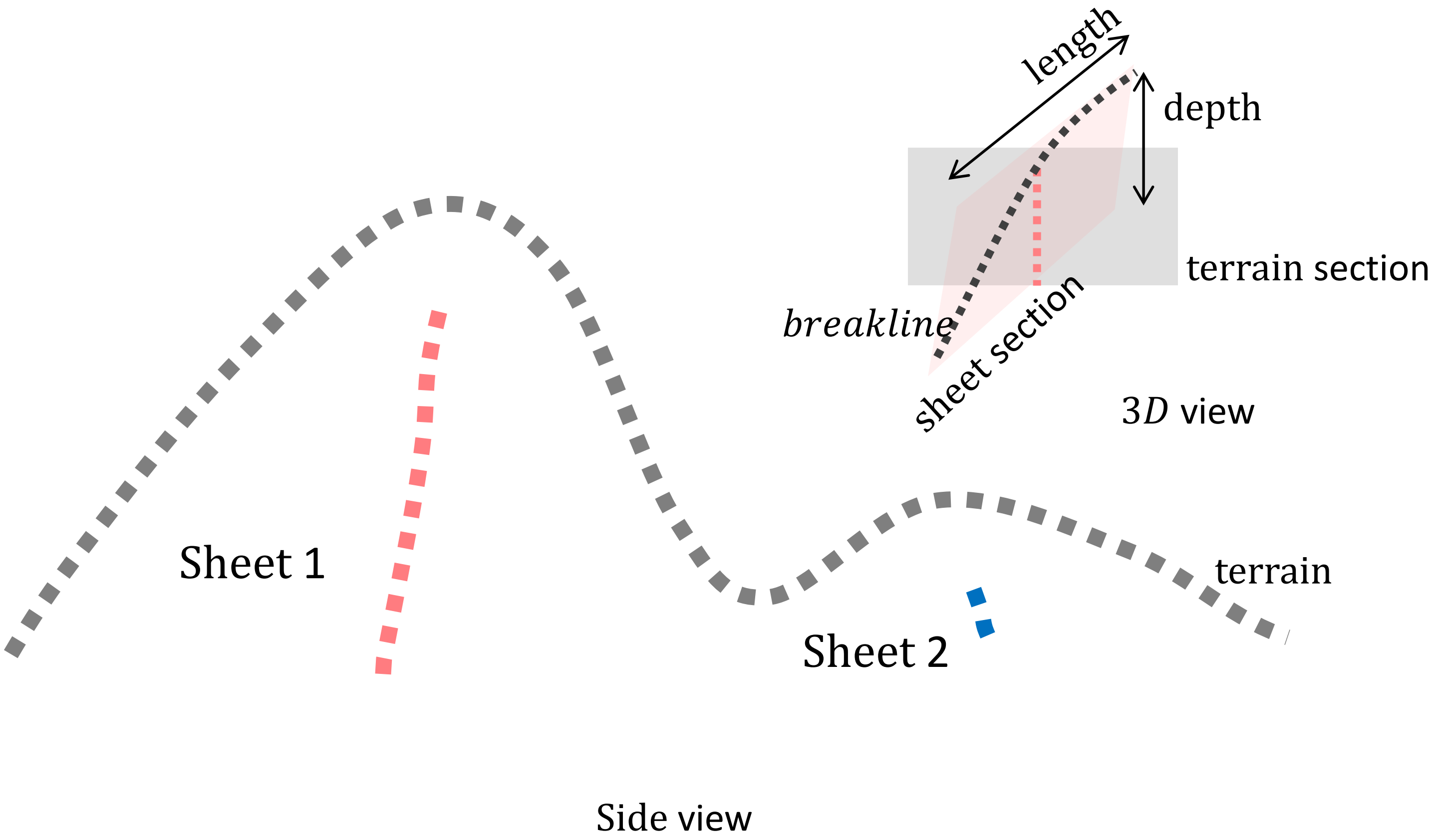


MAT segmentation

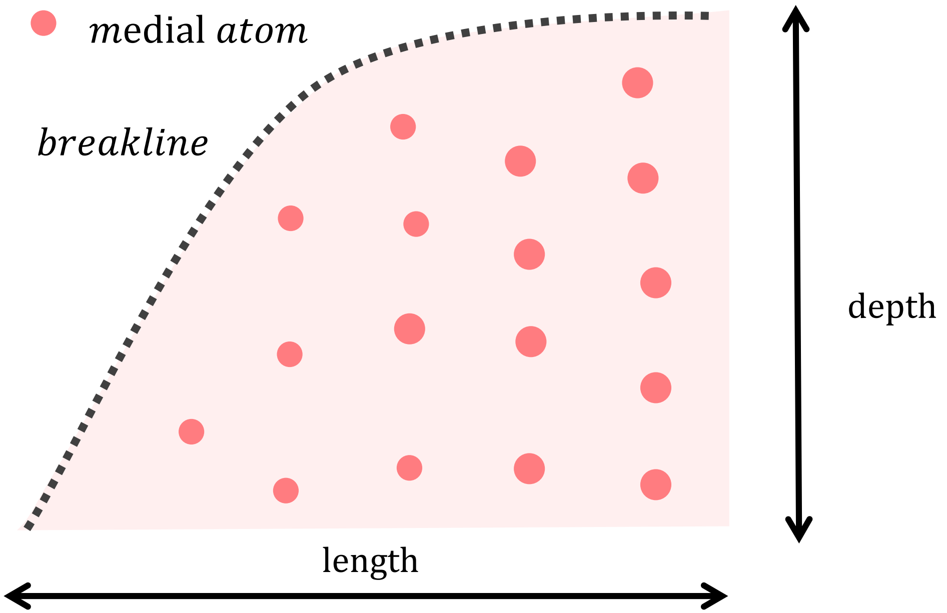
- Medial segmentation → medial sheet



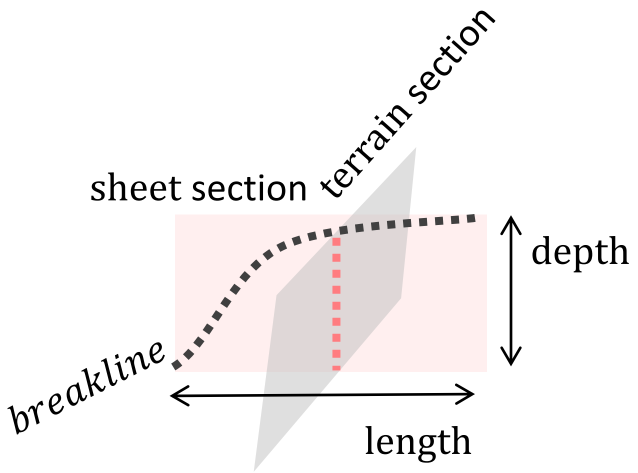
Medial sheet sideview 1



Medial sheet sideview 2

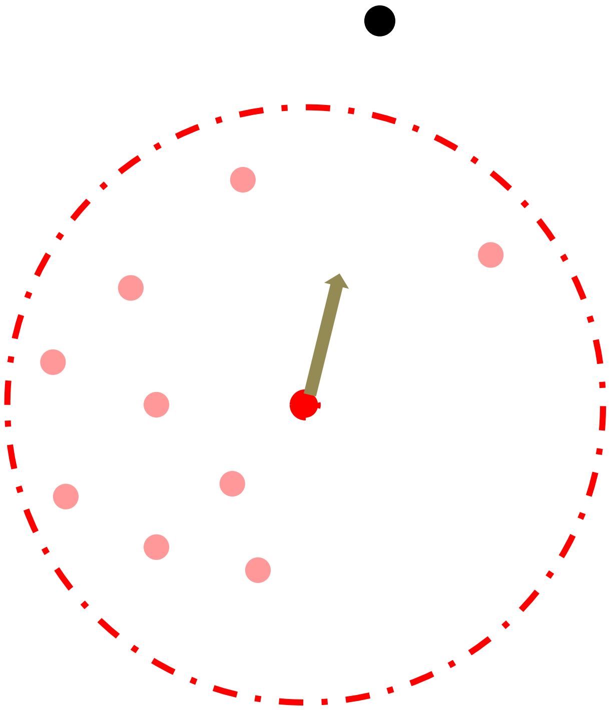


Side view

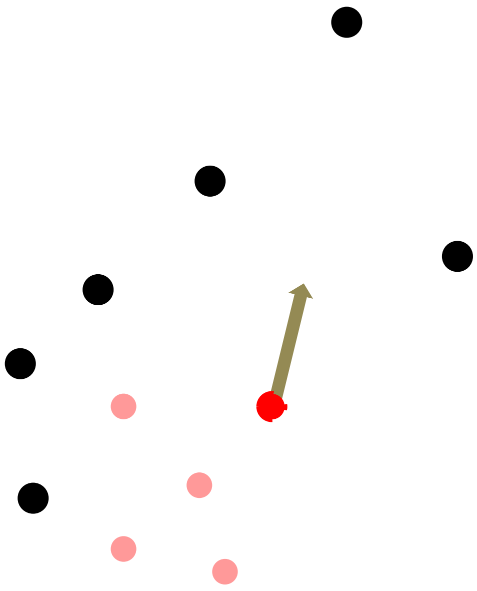


3D view

Methodology:
edge ball detection: neighbor search



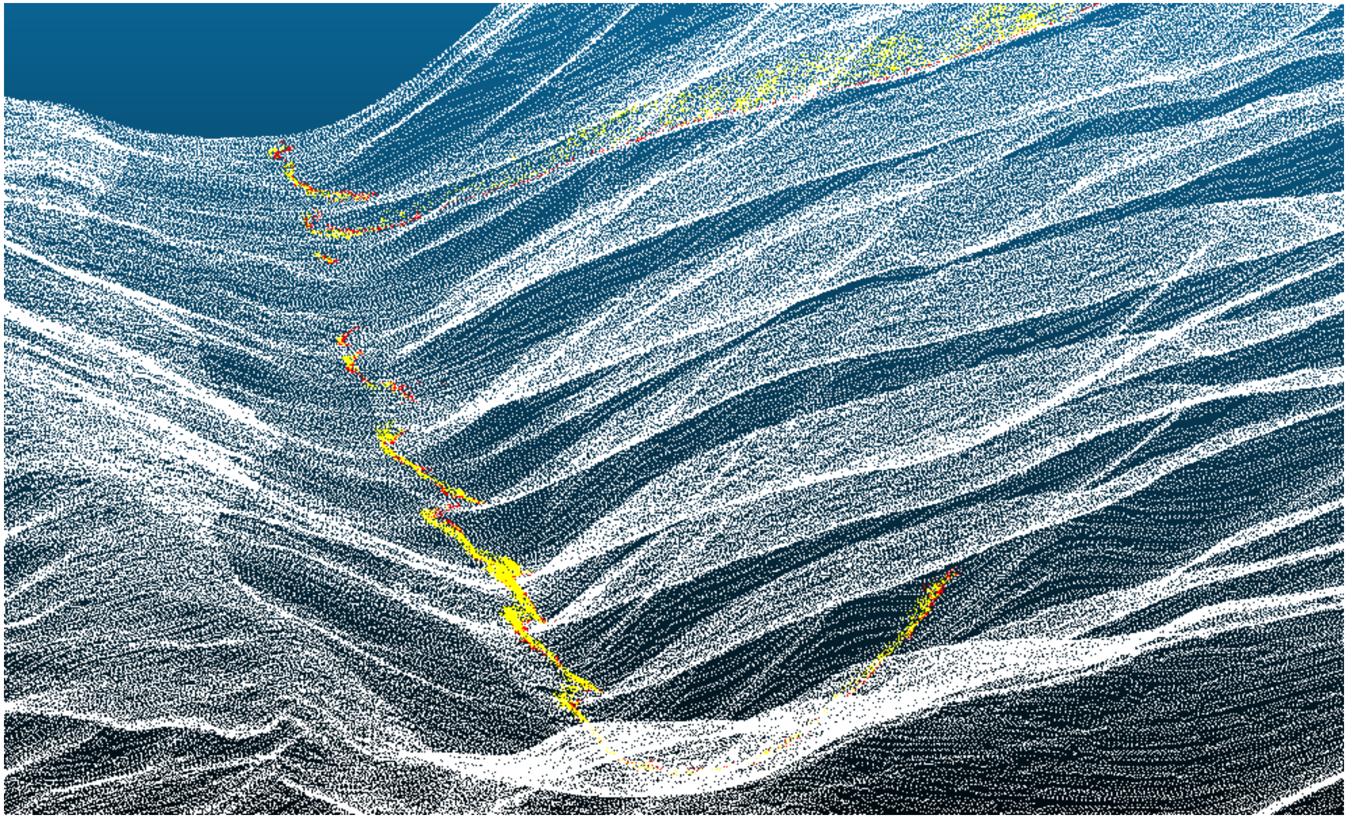
radius neighbour search



5-nearest-neighbour search

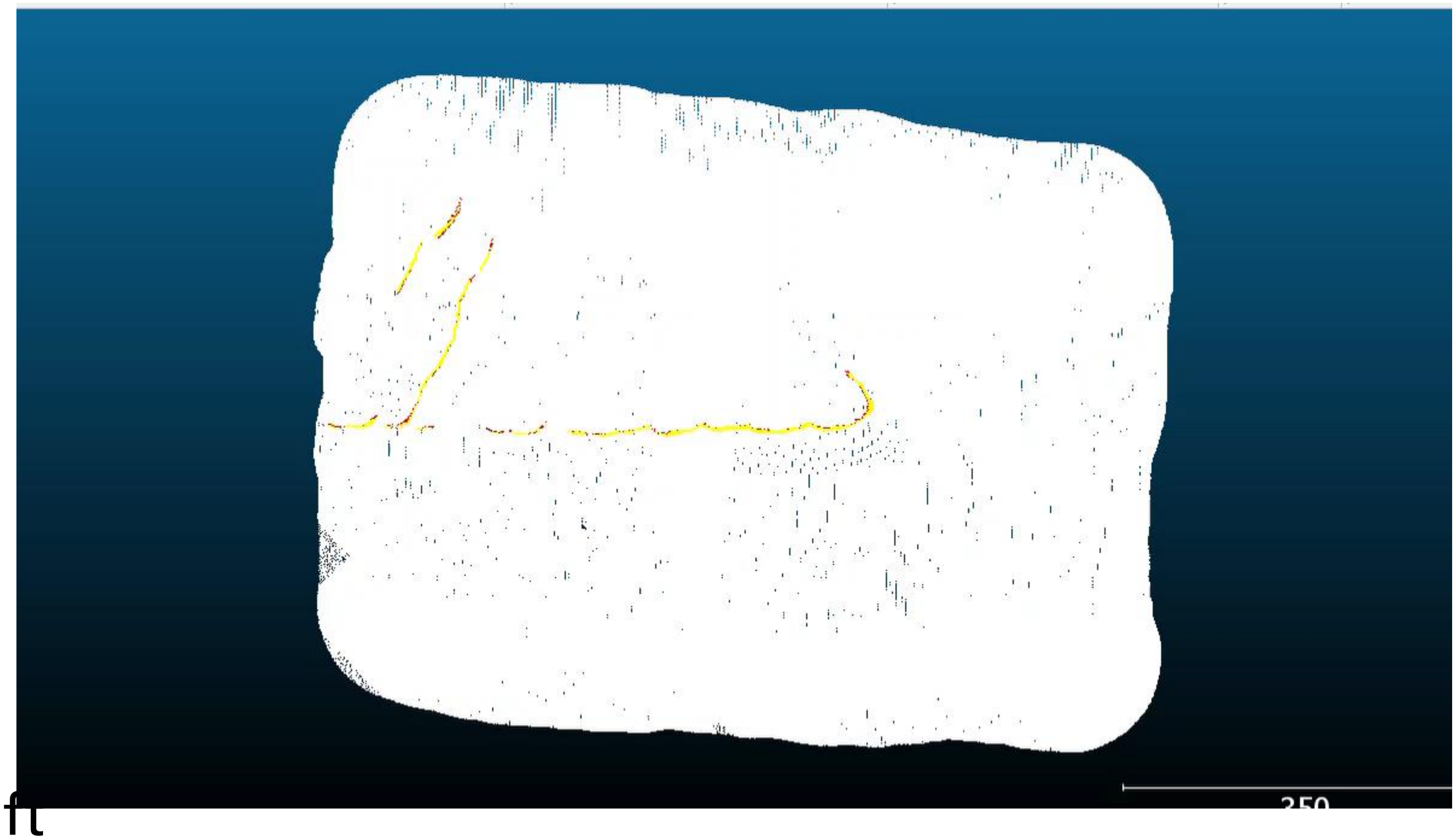
Methodology:
edge ball detection: neighbor search

- Extract candidate points
  - KNN (yellow) vs search radius (red)



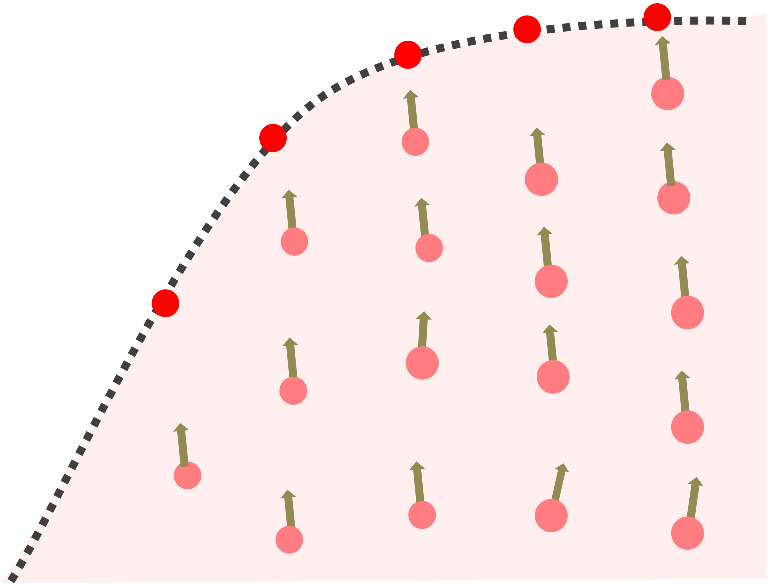
Methodology:
edge ball detection: neighbor search

- Extract candidate points
  - KNN (yellow) vs search radius (red)



Methodology: extracting candidate points

- calculating candidate point coordinates (X, Y, Z)



For each sheet's edge points P：

Find k-nearest-neighbor

$$\mathrm{R} = \sum r_i$$

$$\vec{b}_{avg} = \sum (bisector_i \times \frac{r_i}{R})$$

$$X = P_x + r_P \cdot \vec{b}_{avg_x}$$

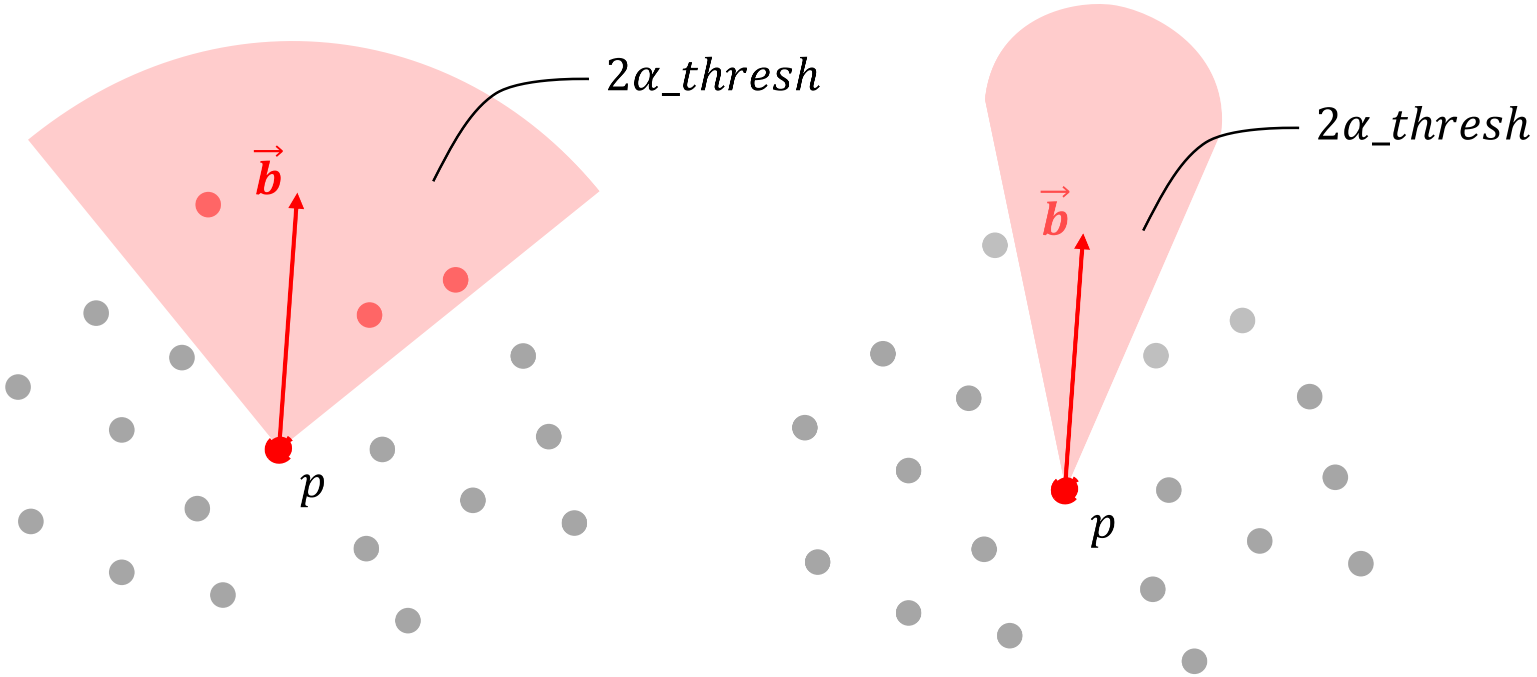
$$Y = P_y + r_P \cdot \vec{b}_{avg_y}$$

$$Z = P_z + r_P \cdot \vec{b}_{avg_z}$$

Methodology: extracting candidate points

- Detecting edge balls

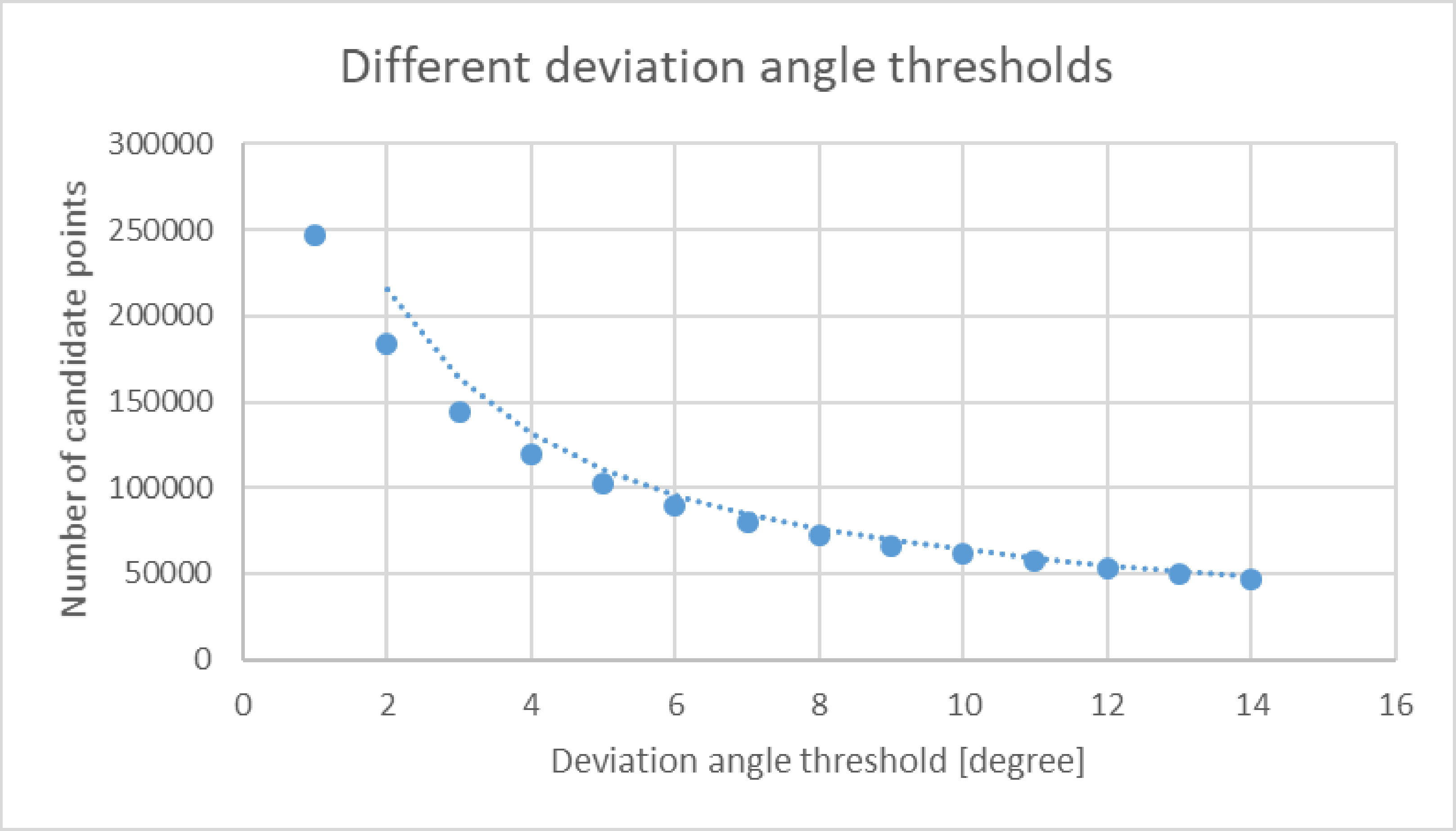
Small α vs large α



p is not an edge ball

p not an edge ball

Methodology: extracting candidate points

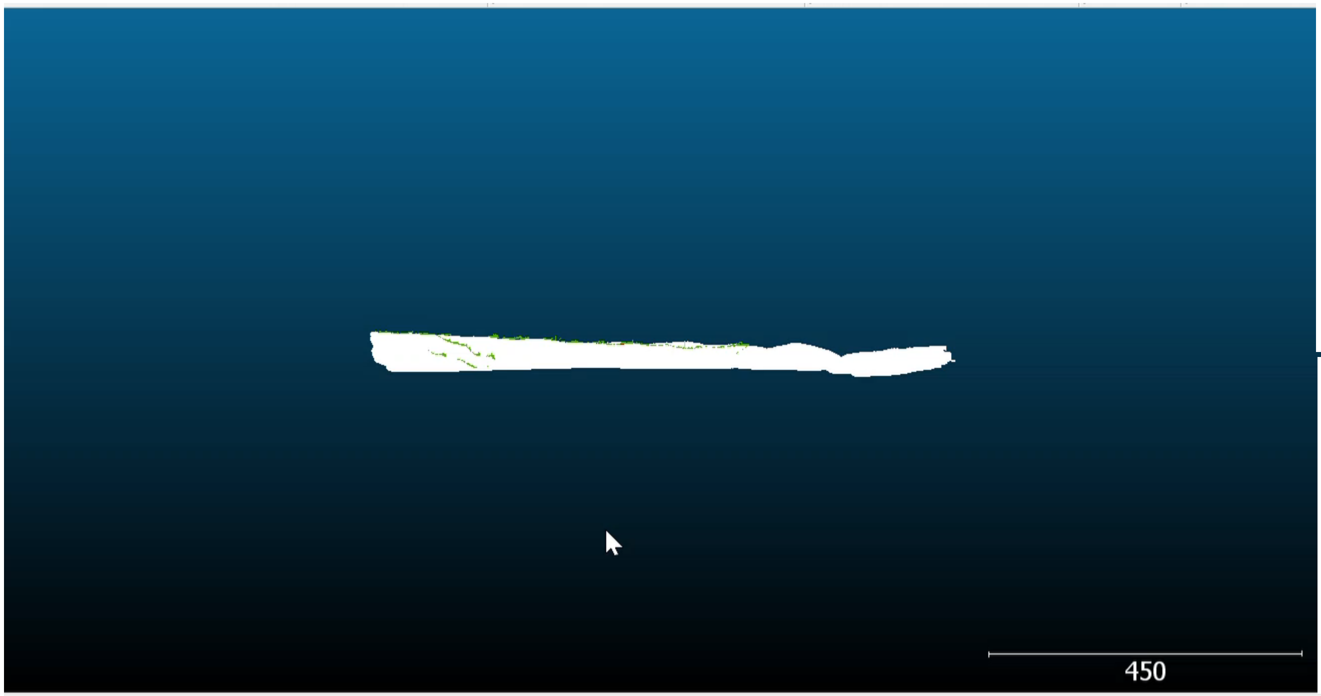


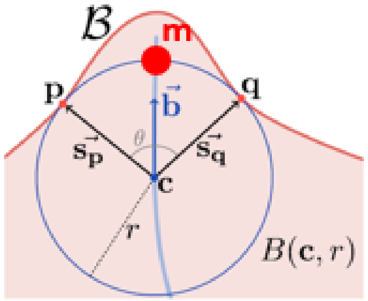
total number of medial atom is 336839.

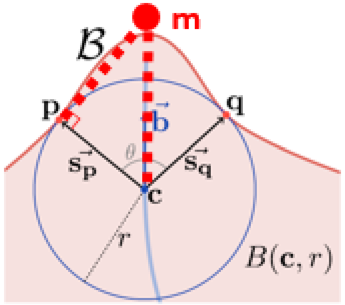
Methodology: extracting candidate points

- calculating candidate point coordinate (X, Y, Z)
  - Intersection of bisector and the medial ball (red)

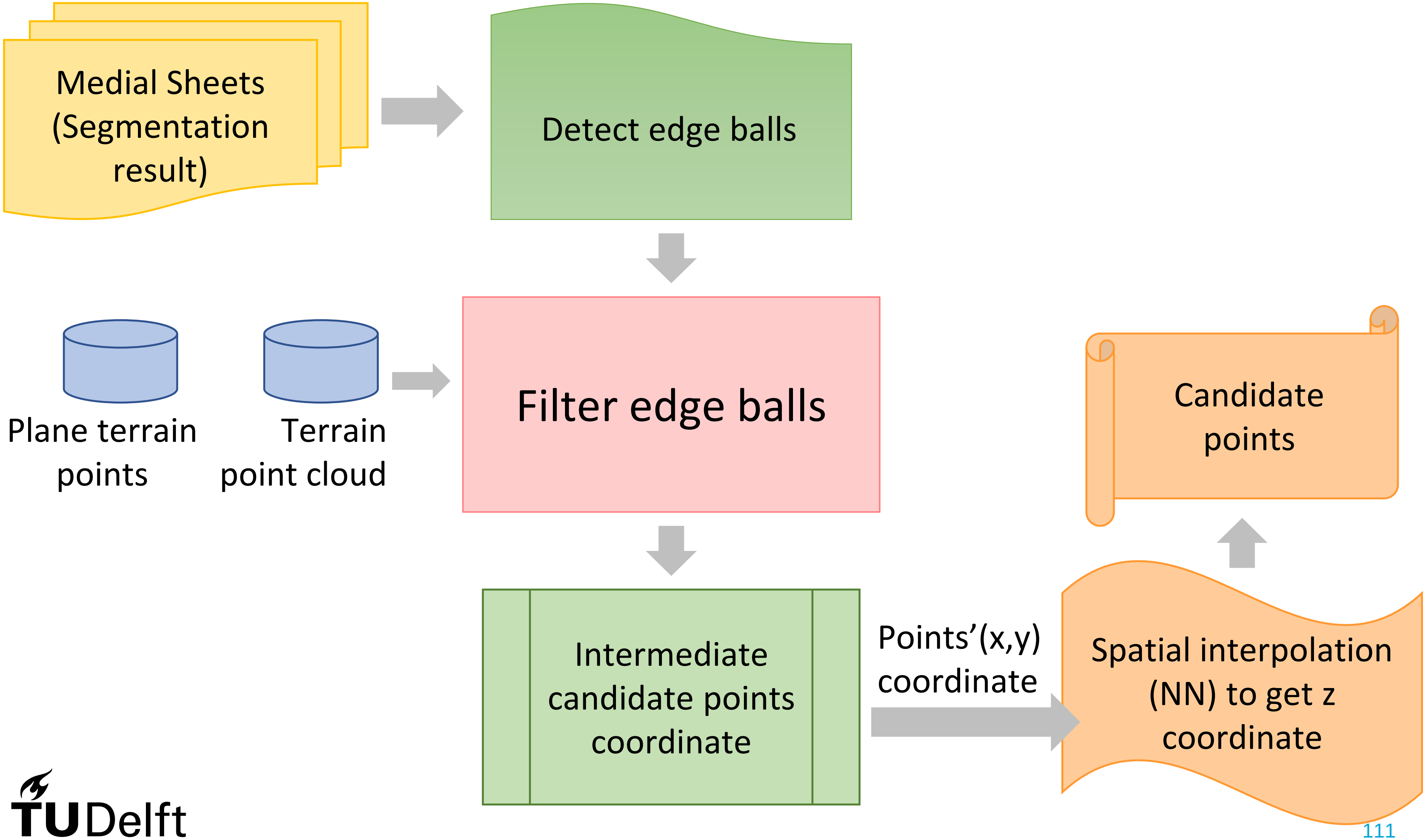
  Vs. Intersection of bisector and right- angle side (green)







Methodology: extracting candidate points



Medial Sheets (Segmentation result)

Detect edge balls

Plane terrain points

Terrain point cloud

Filter edge balls

Intermediate candidate points coordinate

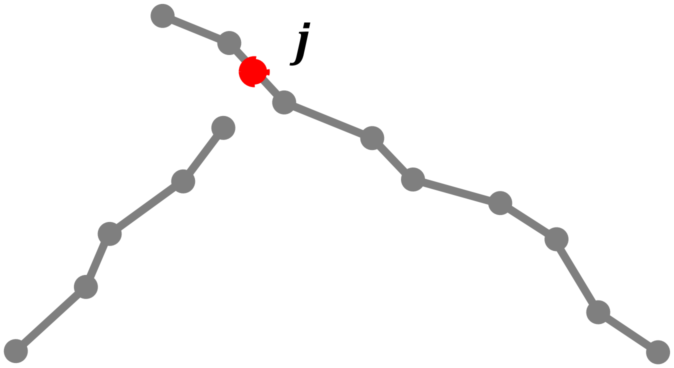
Points'(x,y) coordinate

Spatial interpolation (NN) to get z coordinate

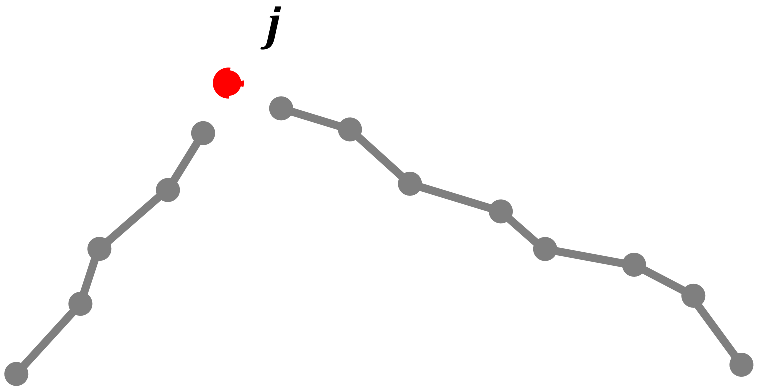
Candidate points

Methodology: connecting candidate points

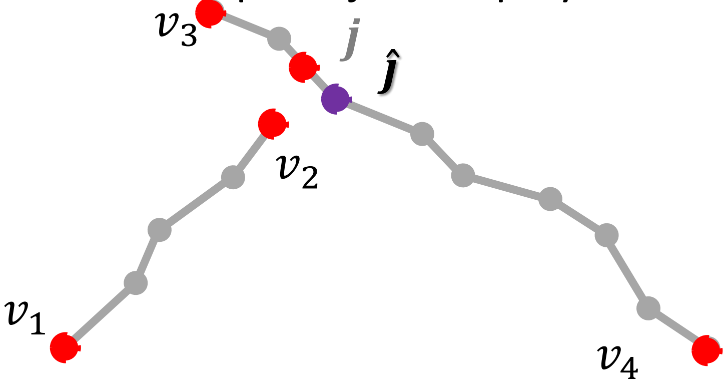
- Finding the topology of breaklines



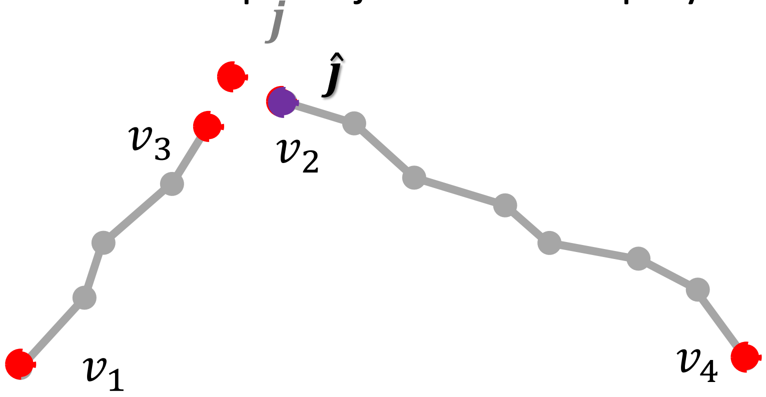
Junction point j in the polyline



Junction point j out side the polyline



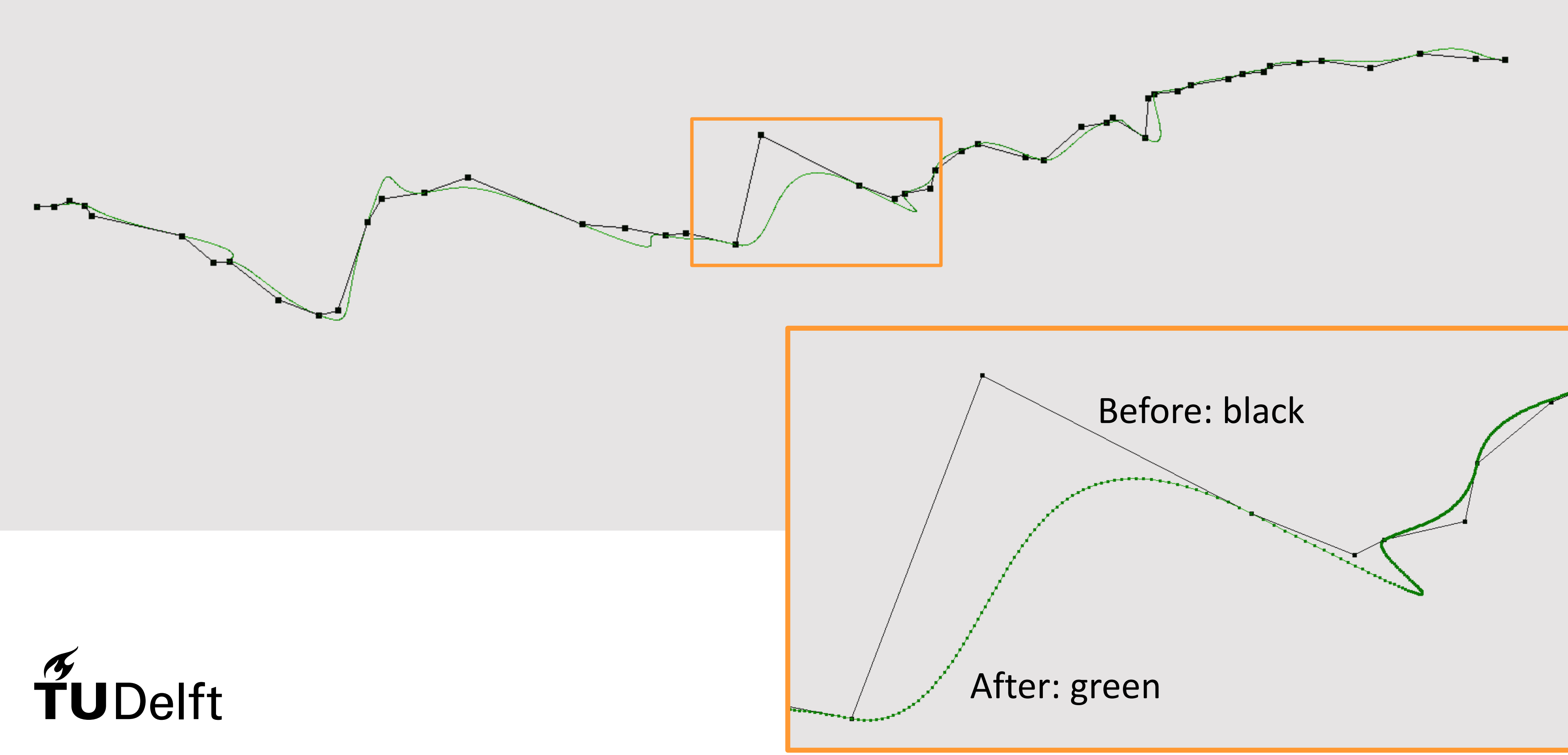
estimated junction $\hat{j}$



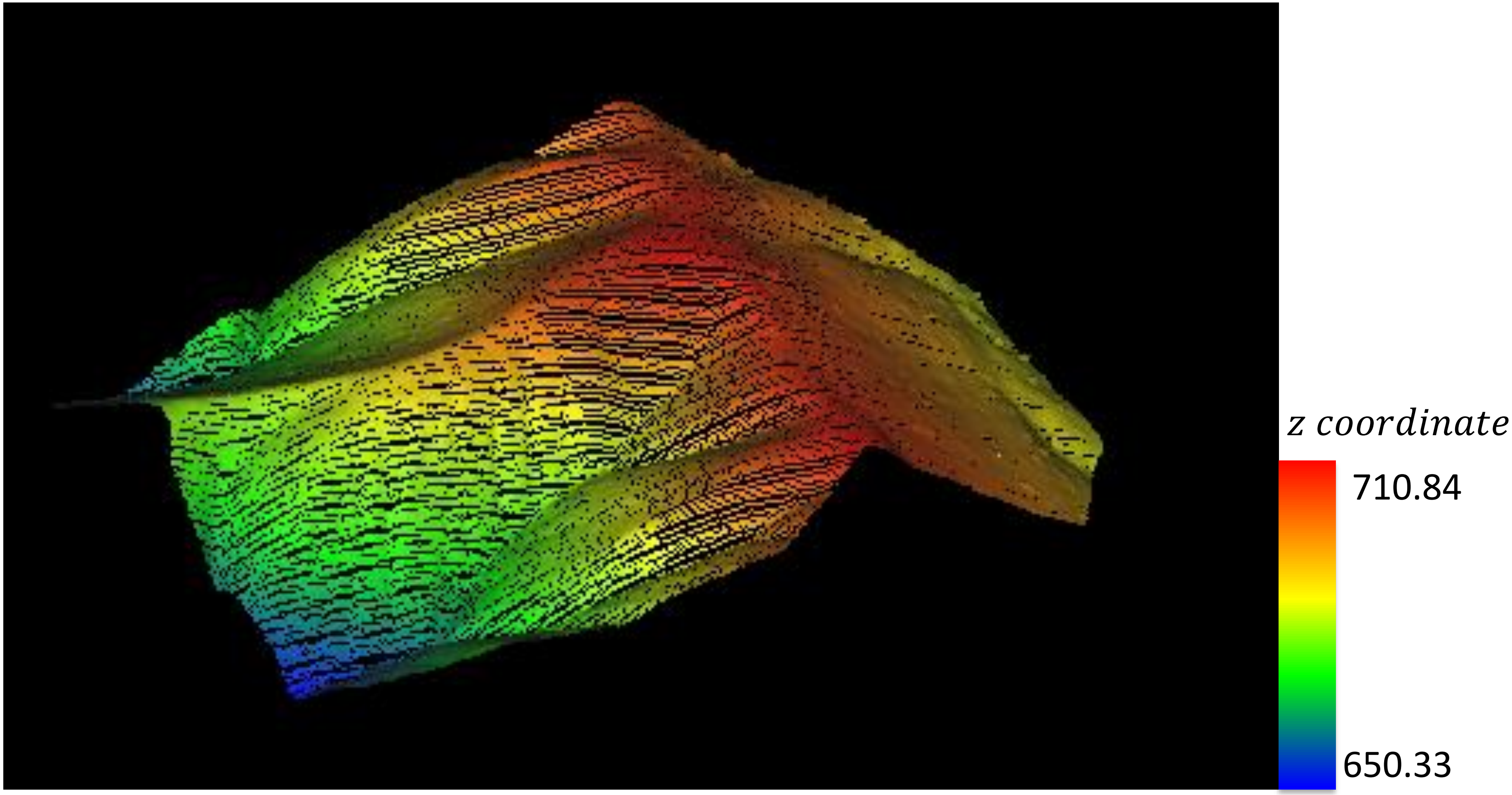
estimated junction $\hat{j}$

Methodology: polyline simplification and smoothing

– Smoothing by b-spline

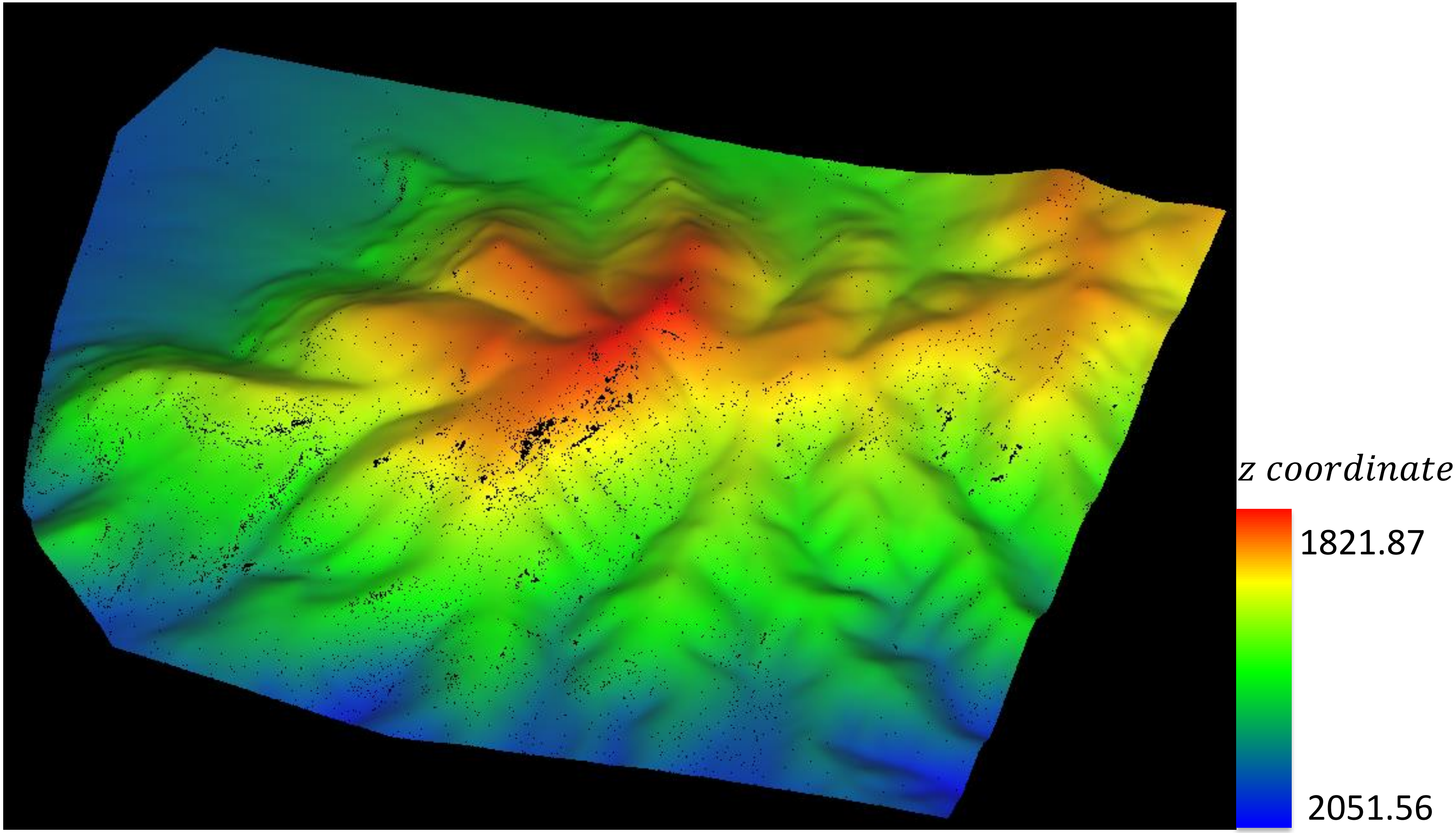


Data 1



Point count: 72493 Point density: 2pts/m^2 height difference: 60.51 m

Data 2



Point count: 2173787 Point density: 6pts/m^2 height difference: 229.69 m