Automatic isobath generalisation

By integrating cartographic constraints in a surface-based approach

Willem van Opstal

Martijn Meijers (1st mentor)

Ravi Peters (2nd mentor)

March 17th 2020

Contents

- Motivation and Problem statement
- Research objectives and Questions
- Related work

- Triangle region graph
- Methodology and Process

- Preliminary results

- Next stages

TU Delft

Isobath generalisation

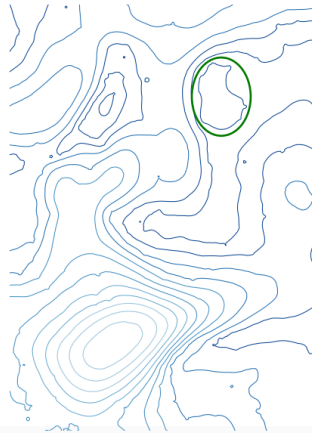
- Cartographic constraints
  - Morphology — Seabed shape
  - Legibility — Readability
  - Functional — Safety
  - Topology — Topology
- Currently done manually
- Automation brings:
  - Economic benefits
  - Safety benefits

TU Delft

Problem statement

- Incompatible constraints
  - Chart scales
  - Smoother lines > less morphology
  - Increasing line separation > less morphology
  - Masking safe waters

- Automated process does not exist yet
  - Complex decisions
  - Cartographers insight
  - or Relation with data is destroyed



Research objectives

- Integrate *all* constraints in the process
- In such way, all constraints are valid and thus the information is not over-generalised

  - Eliminate human interference
  - Quantify generalisation constraints
  - Evaluate metrics directly within the process
  - Apply operators locally, rather than globally

TU Delft

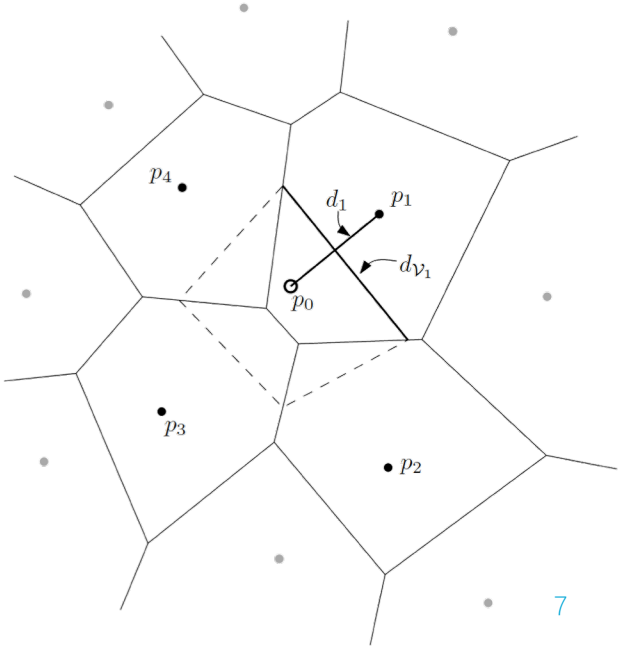
Research questions

- To what extent can we locally steer generalisation operators to account for cartographic constraints, in a surface-based isobath generalisation method?
  - What are the **minimum legibility constraints** for navigational isobaths, cartographic and legally at different chart scales?
  - How can we quantify the cartographic constraints into **local surface metrics**?
  - What is the effect of applying different **local operators** on the global surface, and how can this be exploited?
  - What are valid and realistic assumptions on **input data** in the field of application?
  - How can the extracted features be **validated** and does the method perform better than available alternatives?

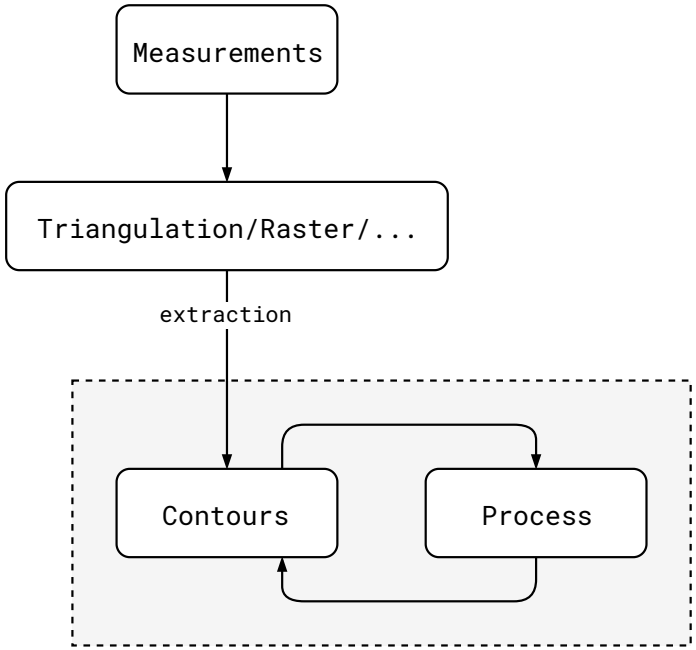
TU Delft

Related work

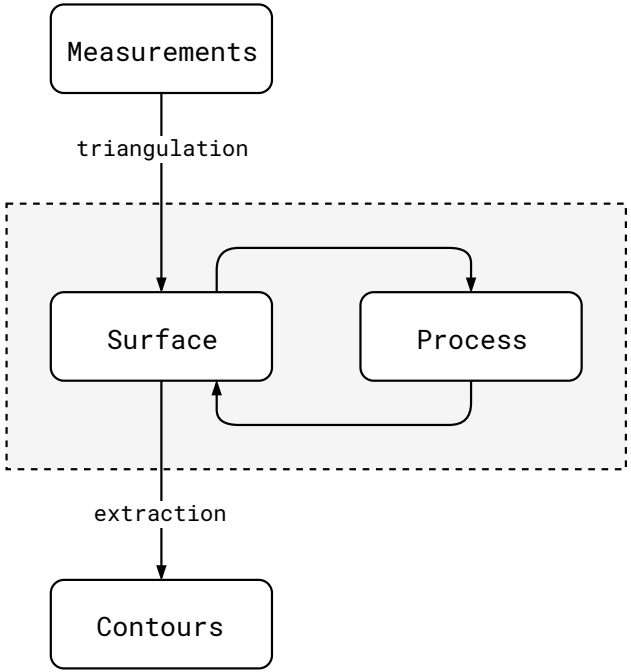
- Line-based generalisation
  - MAS > splines
- Surface-based generalisation
  - Navigational surface
  - Voronoi-based approach



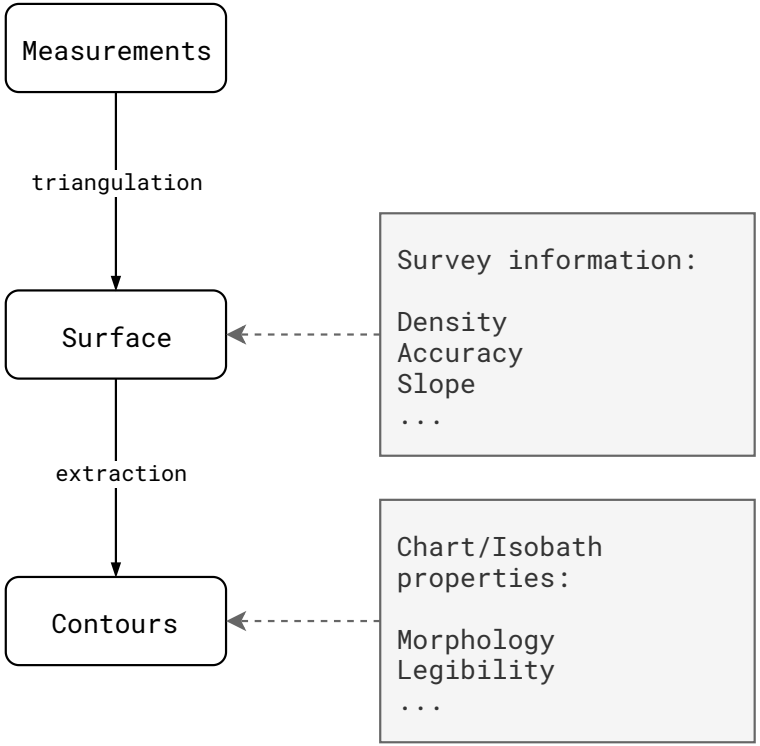
Line-based approach



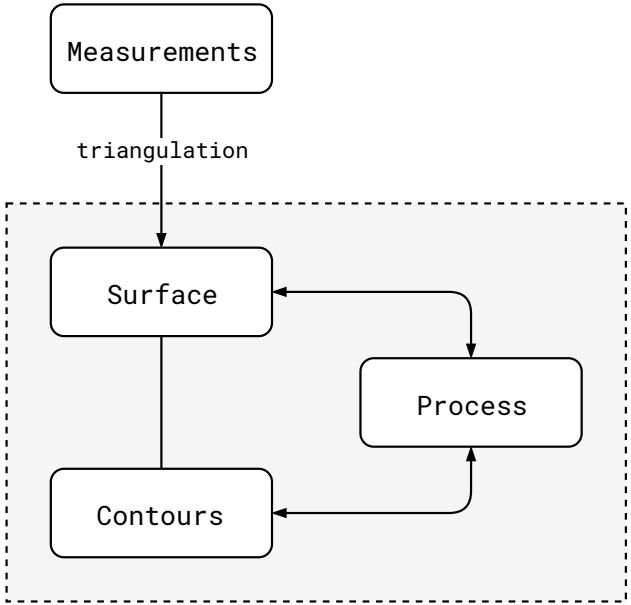
Surface-based approach



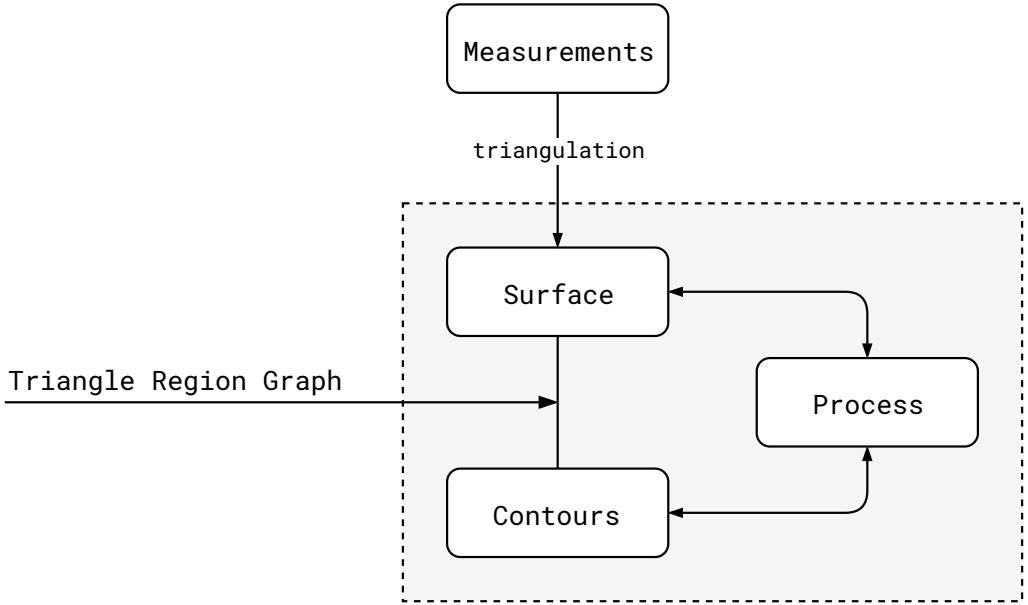
Information difference



Combined



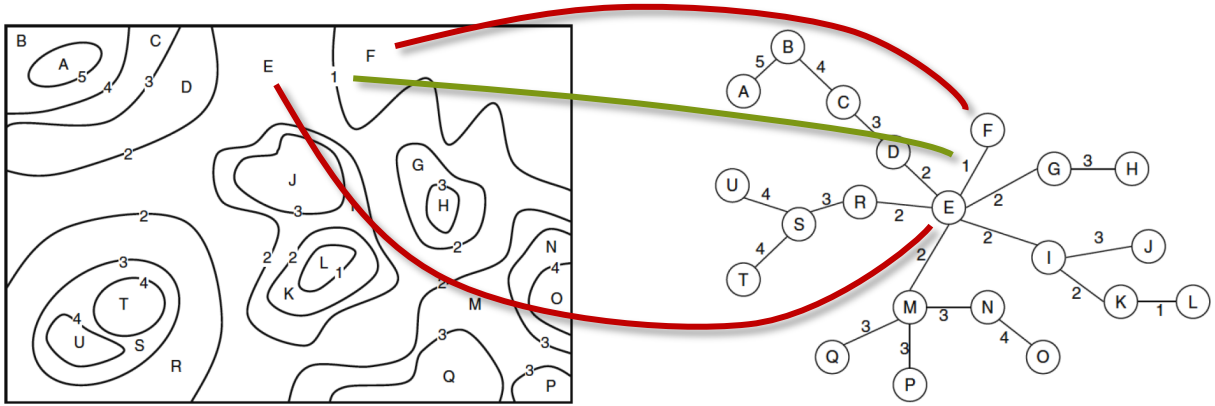
Combined


Measurements
triangulation
Surface
Triangle Region Graph
Process
Contours

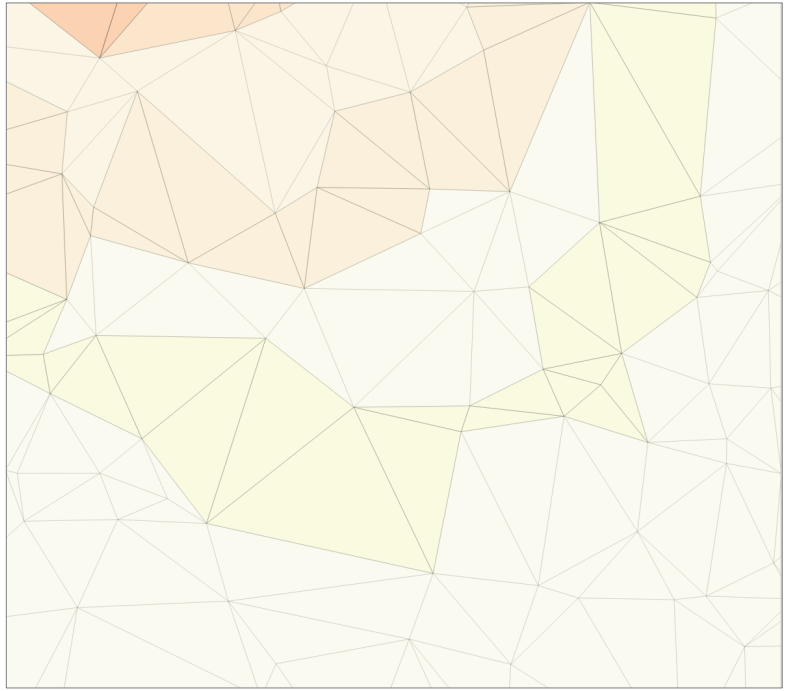
Triangle Region Graph

Triangle region graph (TRG)

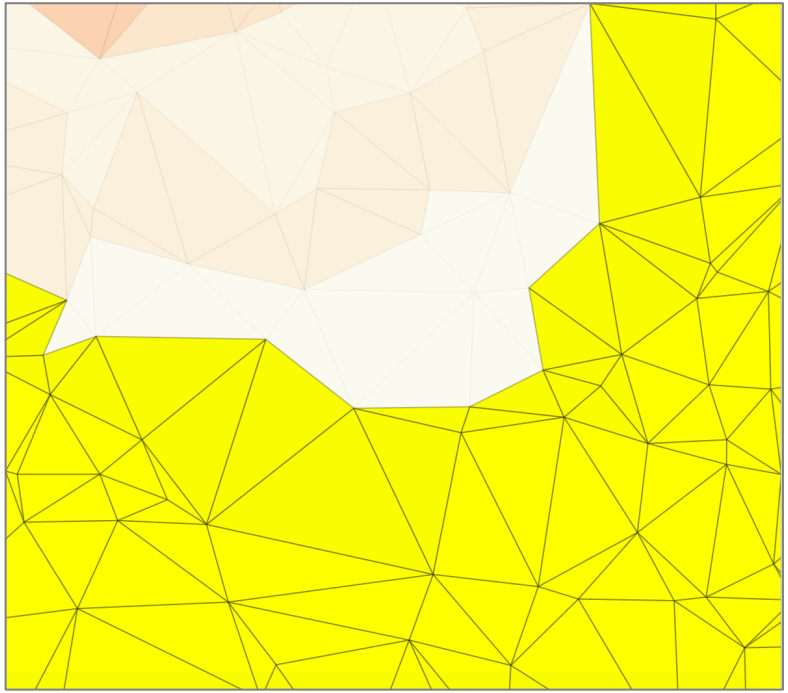
- Links together:
  - Inter-isobaths areas (depth areas > ENCs)
  - Isobaths (e.g. separation)
  - Triangulation (survey data)

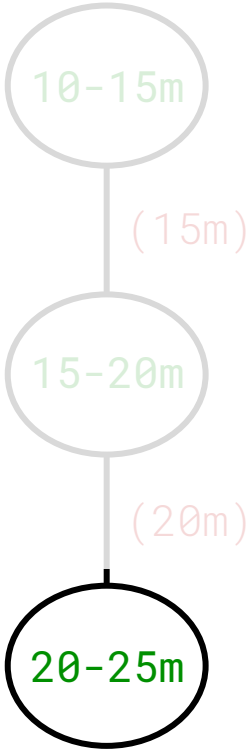


TRG Structure

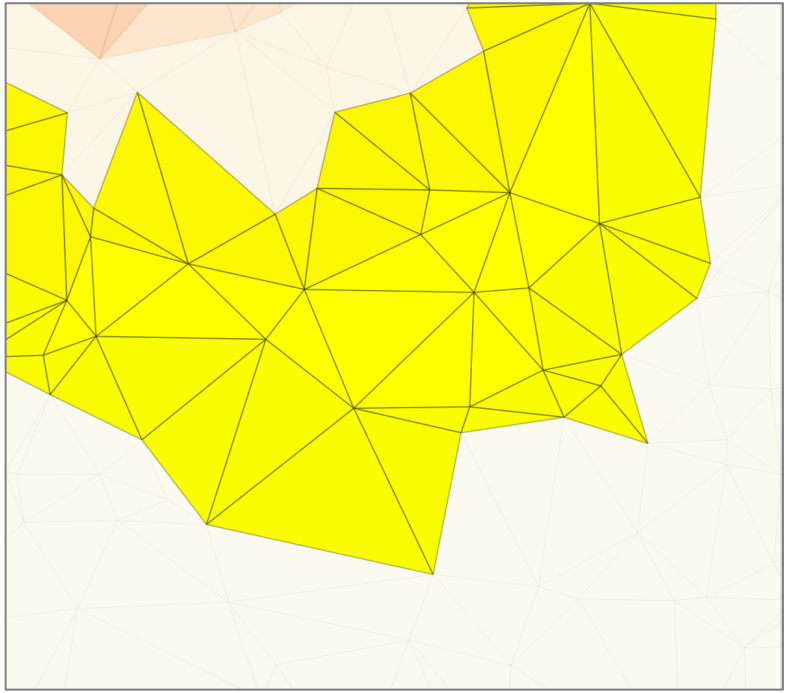


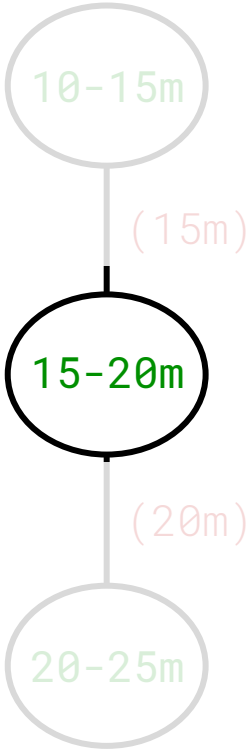
TRG Structure



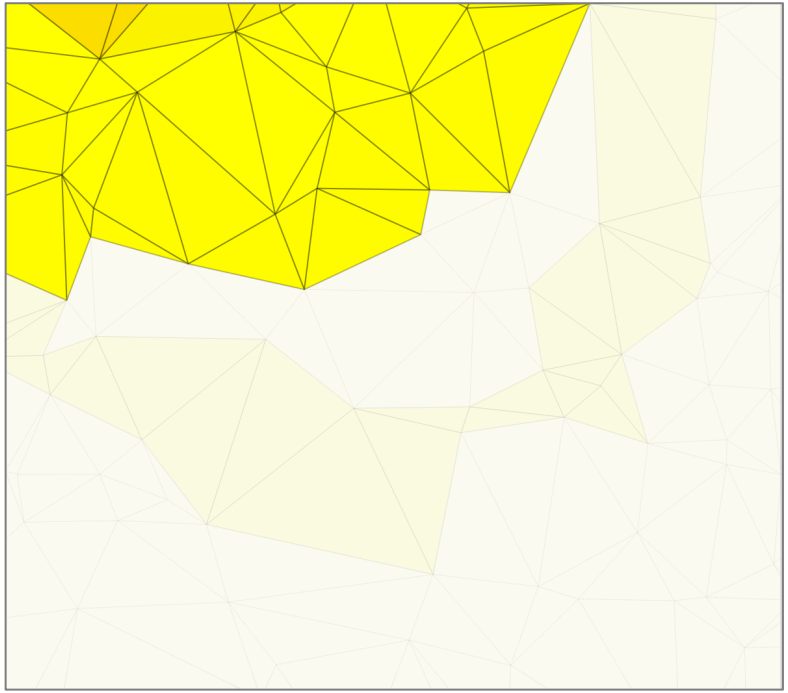


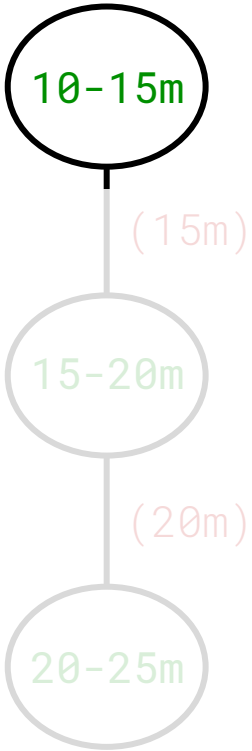
TRG Structure



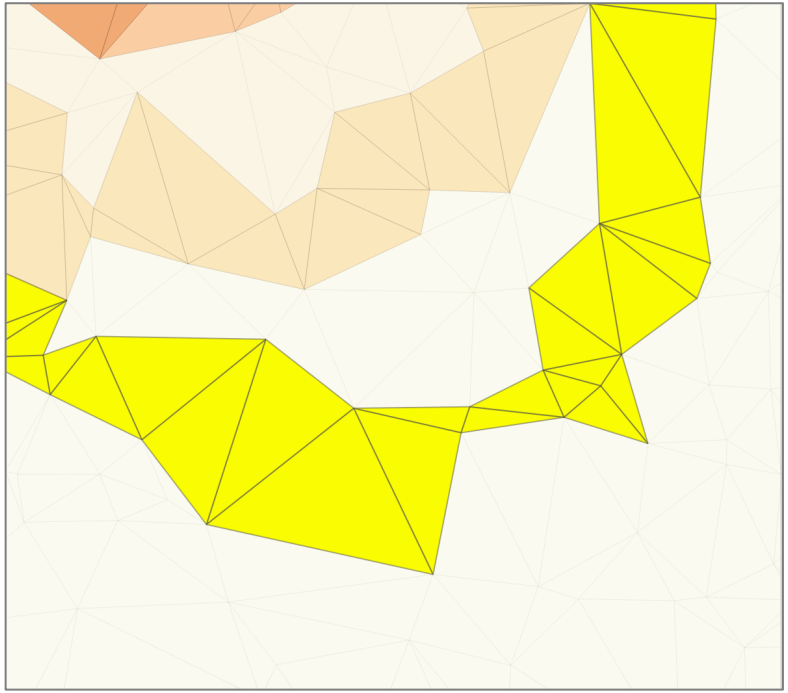


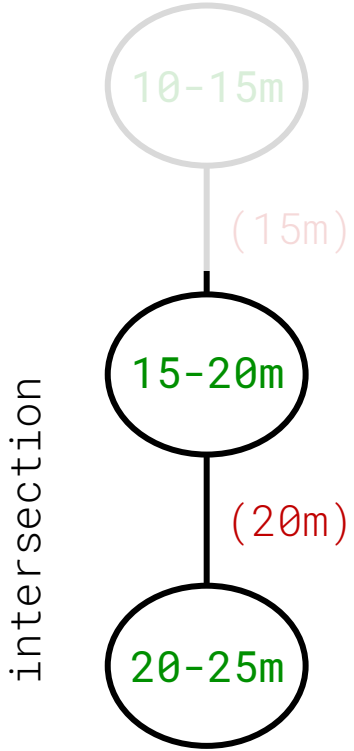
TRG Structure



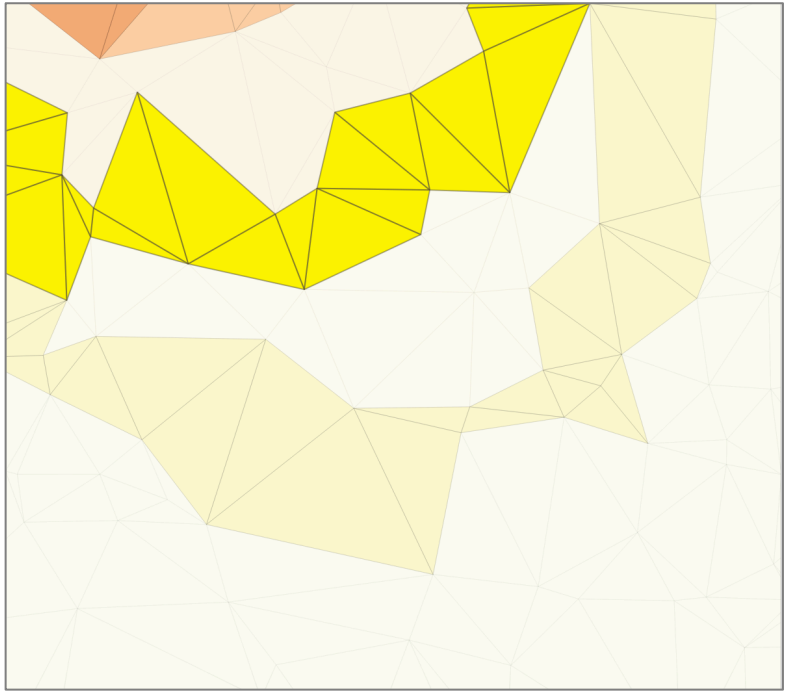


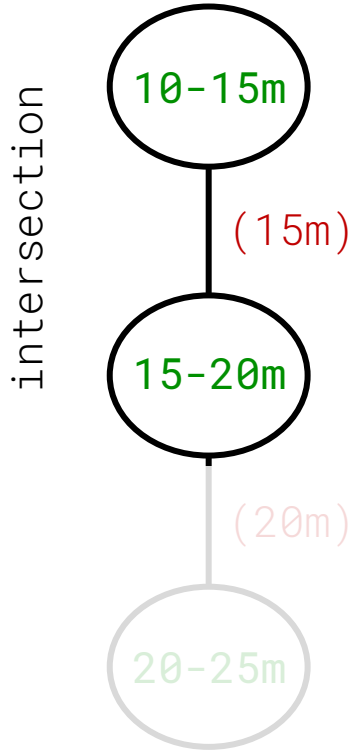
TRG Structure



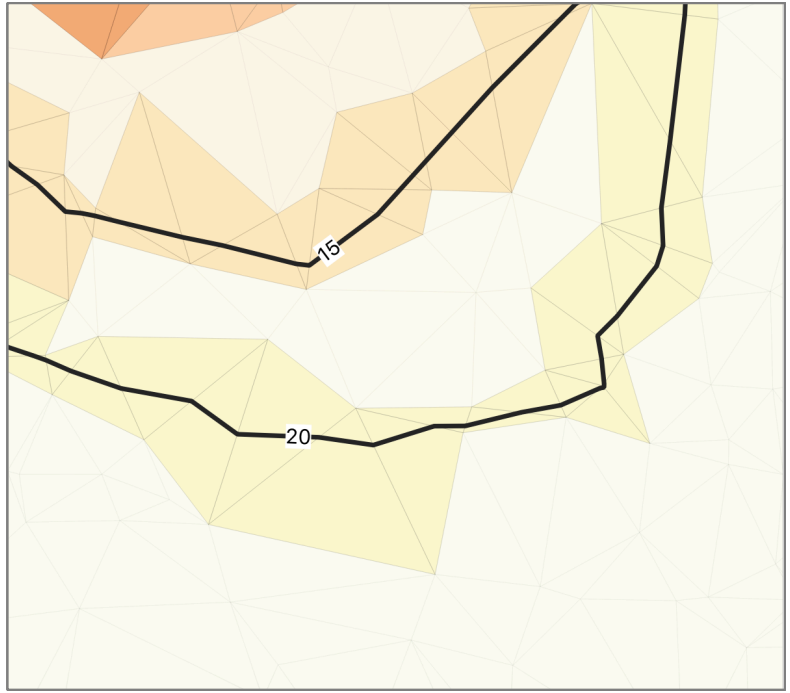


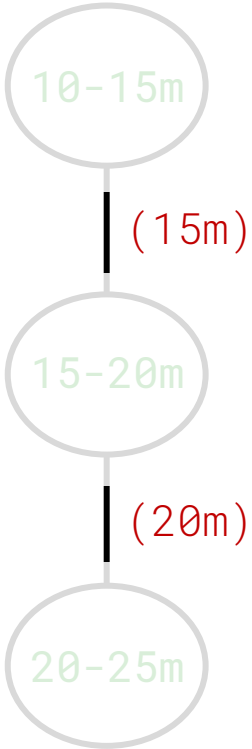
TRG Structure





TRG Structure





TRG Data

- `Nodes[NodeID]`
  - `[triangles]` `{tri, tri, tri, … }`
  - `[region]` `int (> [float, float])`
  - `[deeper]` `{nodeID, nodeID, … }`
  - `[shallower]` `{nodeID, nodeID, … }`
  - `[boundary]` `{edgeID}`
  - `[holes]` `{edgeID, edgeID, … }`
- `Edges[EdgeID]`
  - `[edge]` `[shallowNode, deepNode]`
  - `[closed]` `[Bool]`
  - `[value]` `[float]`

`after/during isobath generation:`

- `[ordered_tris]` `{1: tri, 2: tri, … }`
- `[tri_segs]` `{1: [[vId, vId], [vId, vId]] }`
- `[iso_geom]` `[[x,y], [x,y], … ]`
- `[iso_pointers]` `[[x,y]: {tri, tri, … }, …}`

TRG Operations

- Traverse from node and edges to:
  - Adjacent nodes (deeper or shallower)
  - Adjacent edges (deeper or shallower)
- Get triangles in node or edges
- Generate isobath geometries
- Generate depth areas (closed)

- Get triangles adjacent to an isobath-vertex
- Expand selected triangles by ‘rings’
- Expand selected triangles by nodes

Methodology

Methodology

- Locally steer generalisation operators, where needed
- Build TRG to link: survey data > triangulation > isobaths
- Repeating process
  - Compute metrics
  - Identify conflicting points/vertices, nodes, triangles
  - Decide on conflict region expansion
  - Apply smoothing or densification

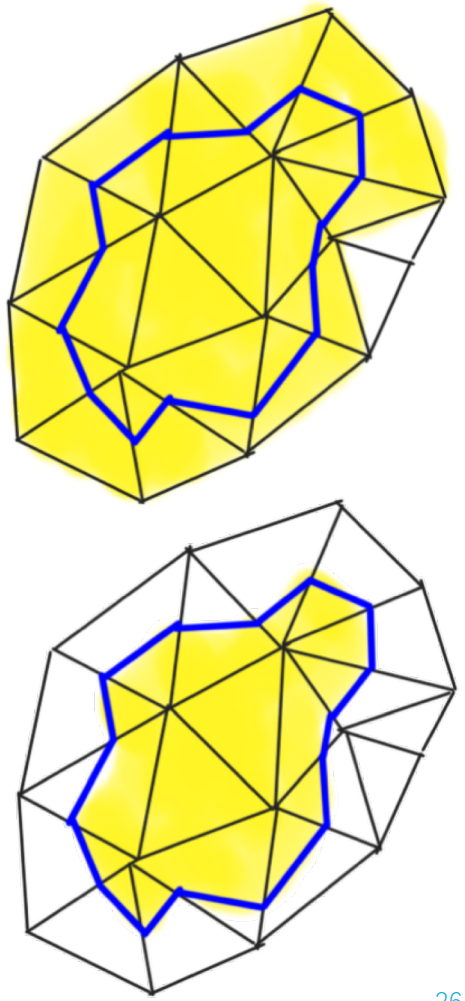
TU Delft

Metrics: Node area

- Full area > sum of triangles
- Isobath area > only if closed isobath

- Peaks need a minimum area

- Conflicts result in:
  - Nodes

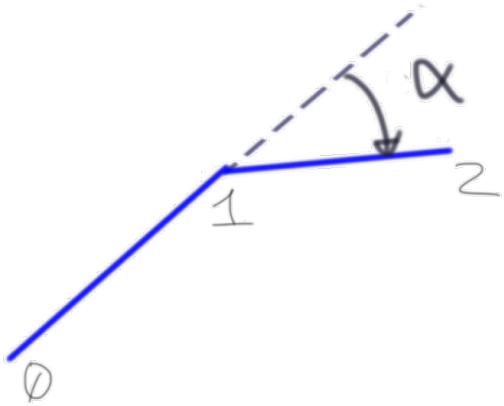


Metrics: Angularity

- Deviation from straight propagation
- Measure for smoothness of the line
- May be averaged for a full edge

- Maximum allowed angle

- Conflicts result in:
  - Isobath vertices

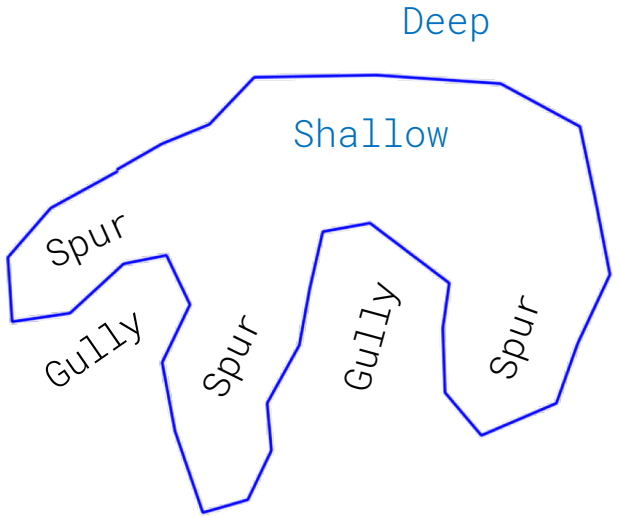


Metrics: Spurs and Gullies

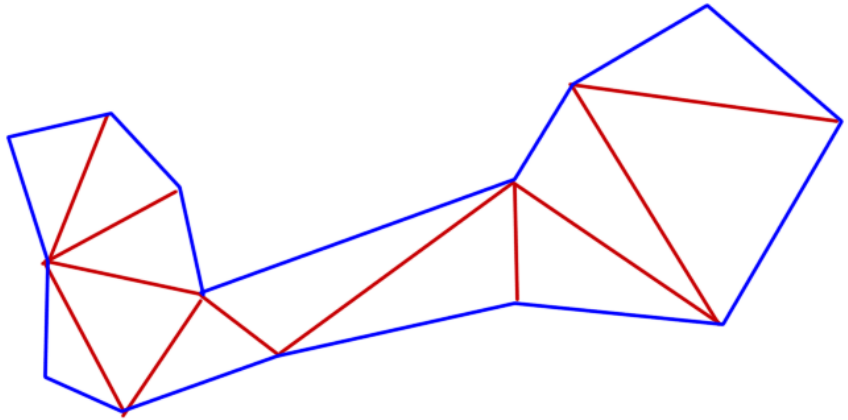
- Indents of isobaths
- Line separation
- Navigable gullies?
- Smoothness on detail level

- Minimum *width* of feature

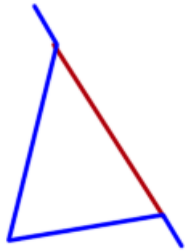
- Conflicts result in:
  - Isobath vertices



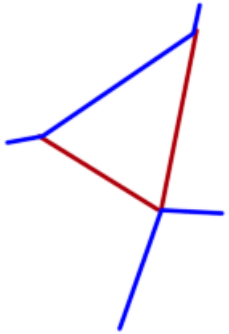
Metrics: Spurs and Gullies



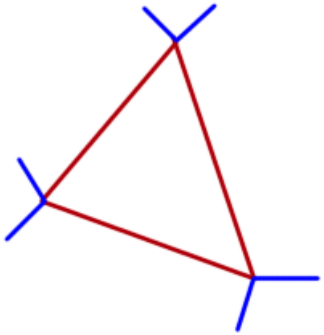
Metrics: Spurs and Gullies



1 segment

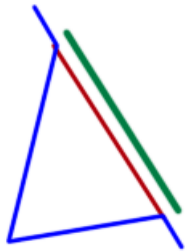


2 segments

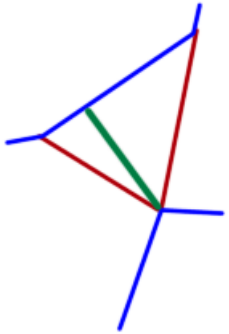


3 segments

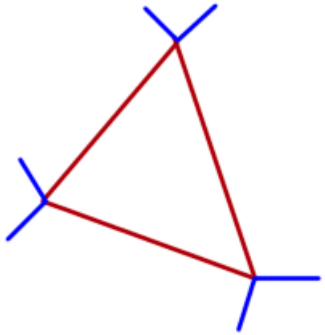
Metrics: Spurs and Gullies



1 segment



2 segments



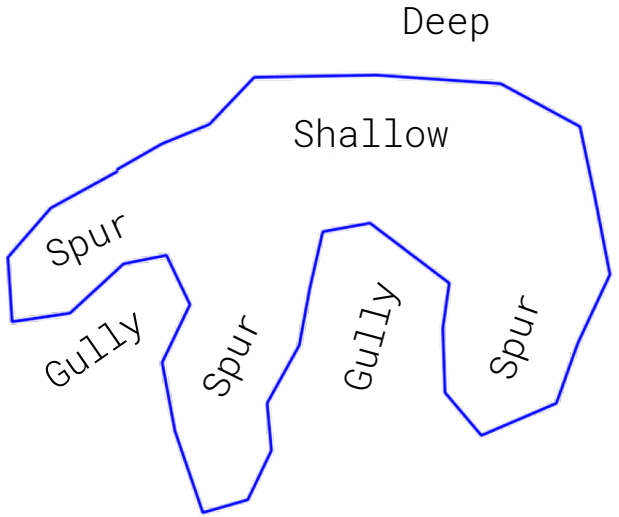
3 segments

Metrics: Spurs and Gullies

- Indents of isobaths
- Line separation
- Navigable gullies?
- Smoothness on detail level

- Minimum *width* of feature

- Conflicts result in:
  - Isobath vertices



Metrics: Triangle area

- Simple triangle area
- Where to densify

- Maximum value

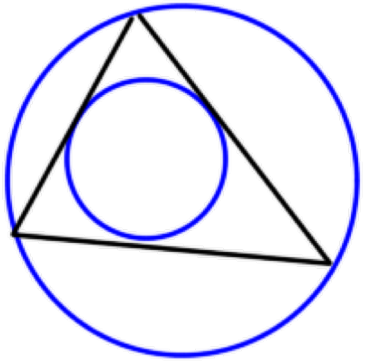
- Conflicts result in:
  - Triangle

Metrics: Triangle aspect ratio

- Simple triangle area
- Where to densify

- Maximum value

- Conflicts result in:
  - Triangle



Process and Implementation

https://www.github.com/willemvanopstal/hydropolator

Process and implementation

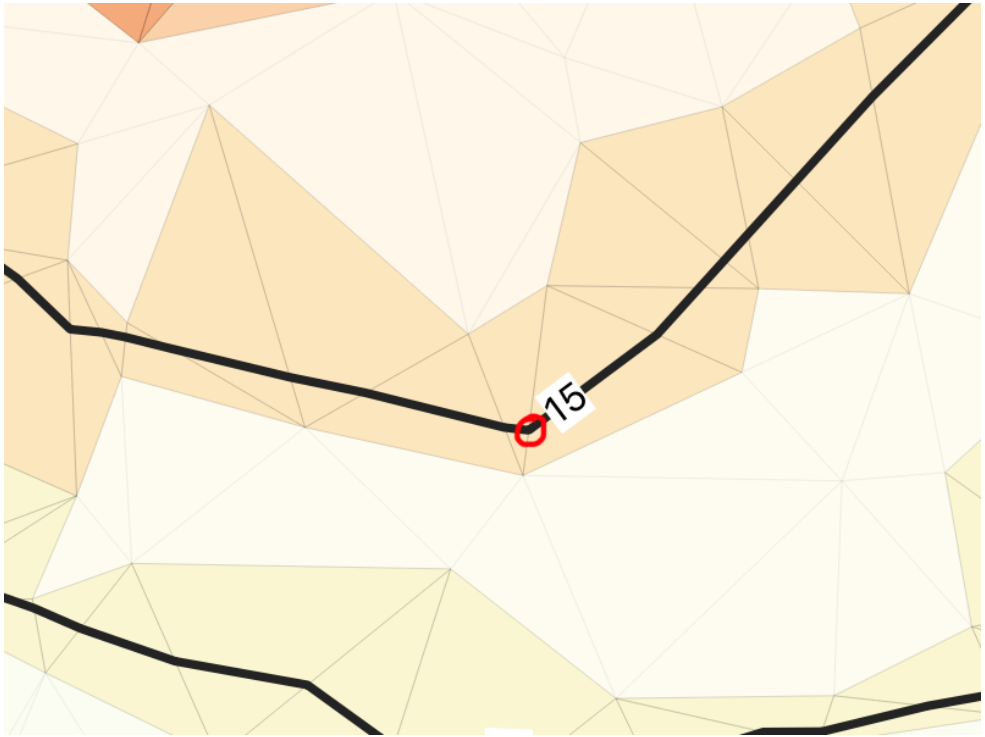
- Generate TRG > Isobaths > Metrics

- Metrics with threshold results in:
  - conflicting nodes — get all node triangles
  - conflicting triangles
  - conflicting isobath vertices — get adjacent triangles

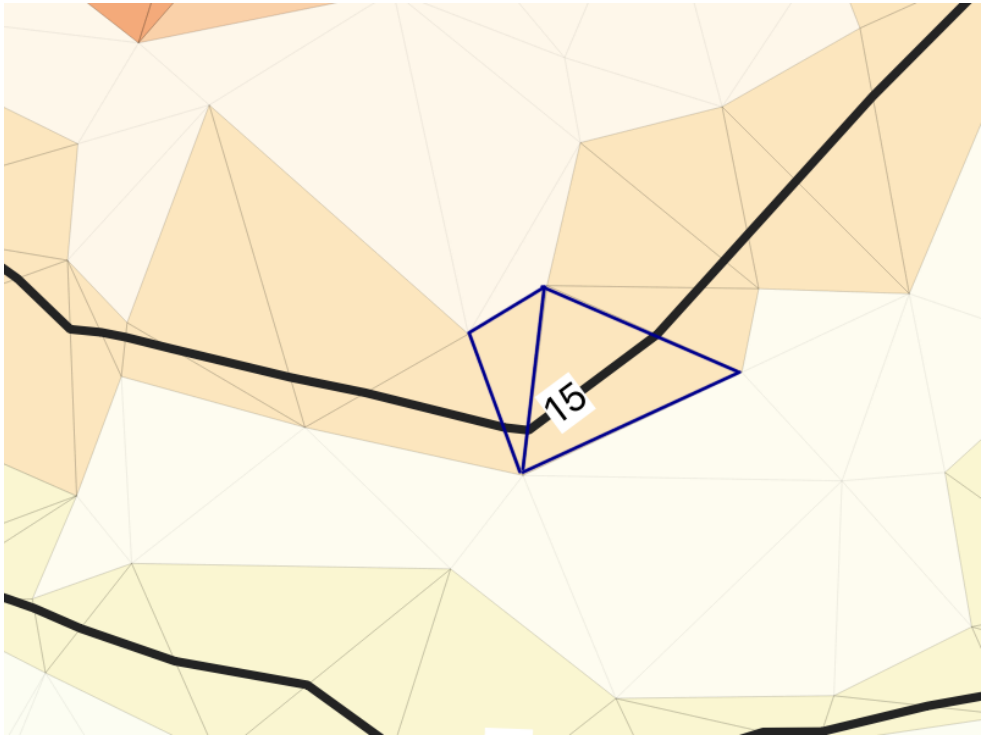
- Possibly extend these triangles
- Extract vertices of extended triangles
- Add vertices to smoothing queue
- Apply operator

TU Delft

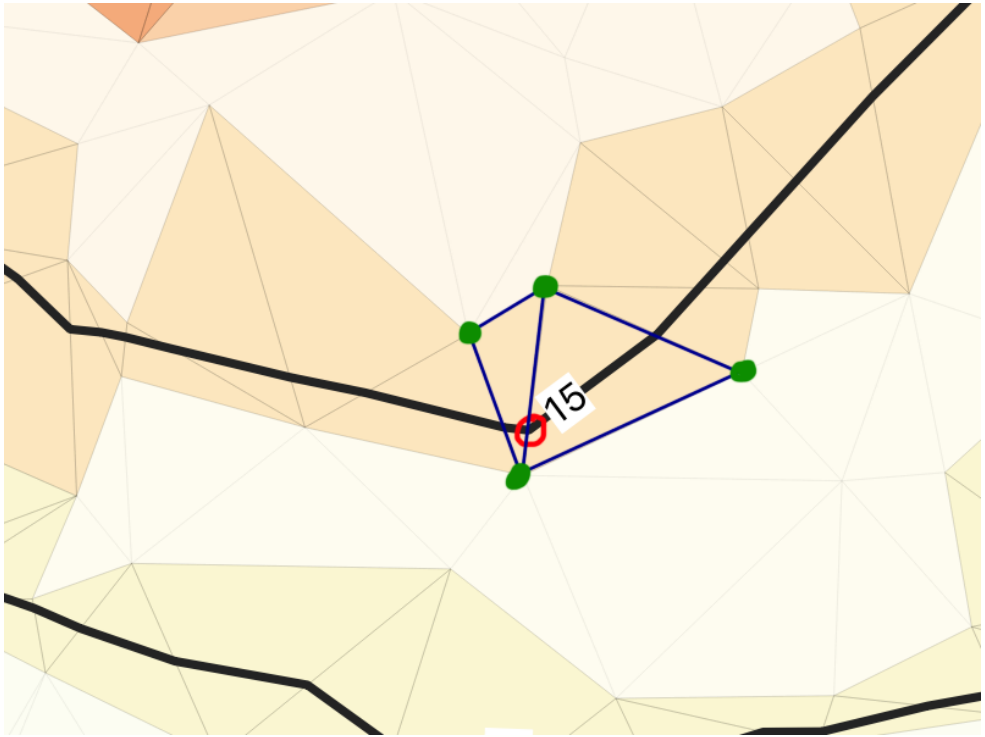
Conflicting isobath-vertex



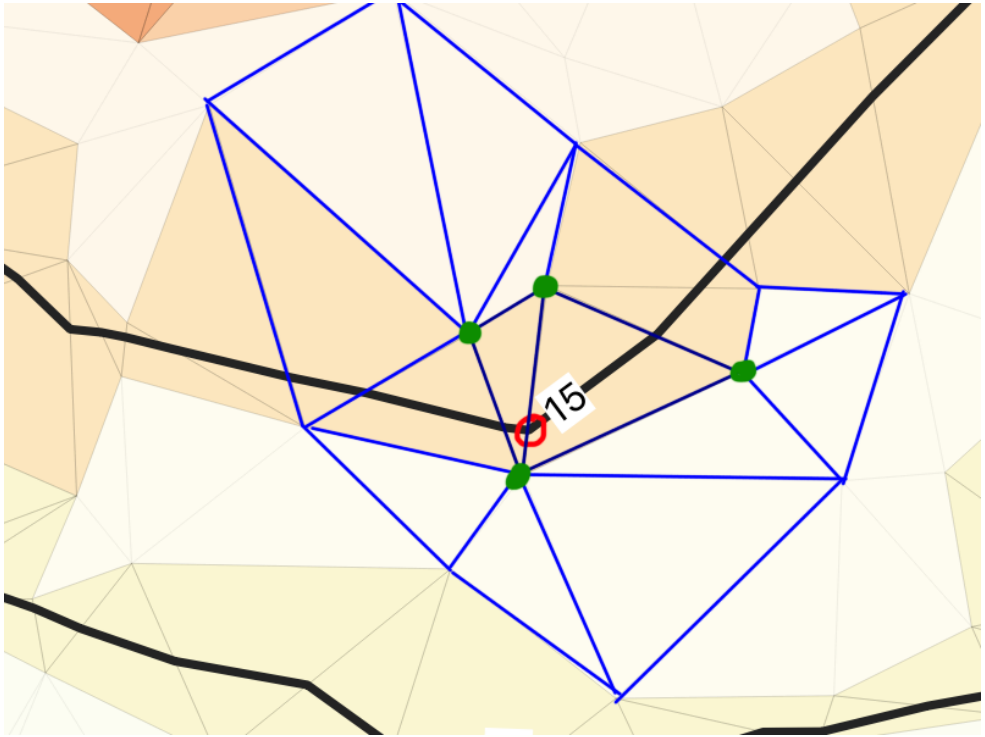
Adjacent triangles



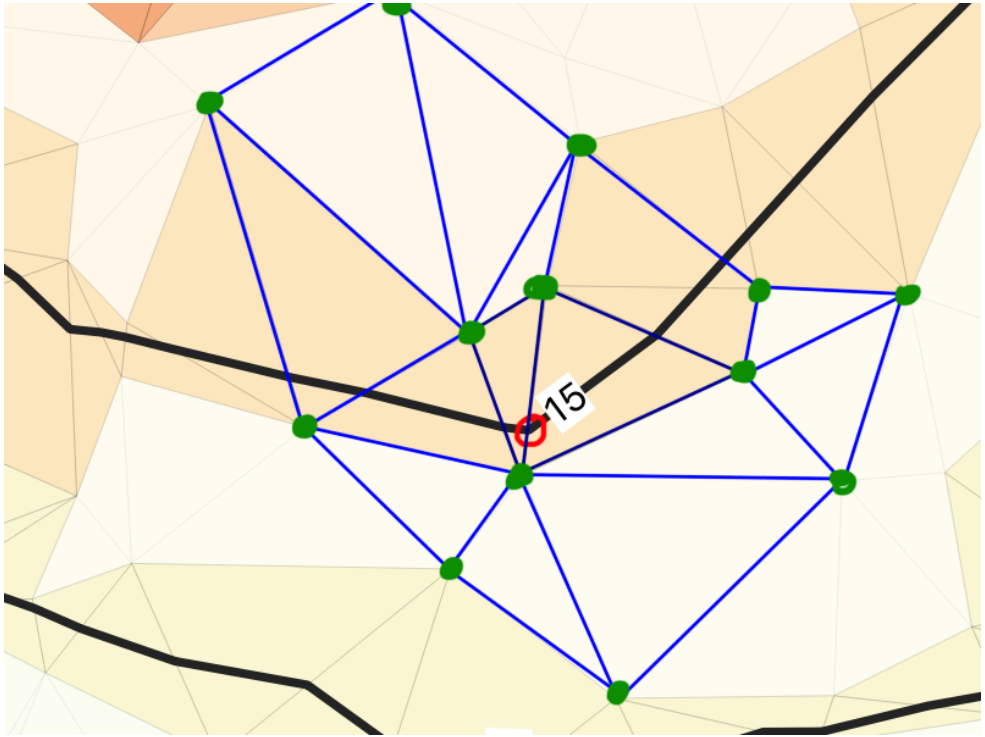
Vertices to smooth



Possibly extend region (rings)



Vertices to smooth



Process and implementation

```
paramDict = {'prepass':                 1,
             'densification':           3,
             'maxiter':                 100,

             'angularity_threshold':    1.0,
             'spurgully_threshold':     None,
             'spur_threshold':          15,
             'gully_threshold':         25,
             'aspect_threshold':        4,
             'size_threshold':          60
             'seg_threshold':           100,
             }
```

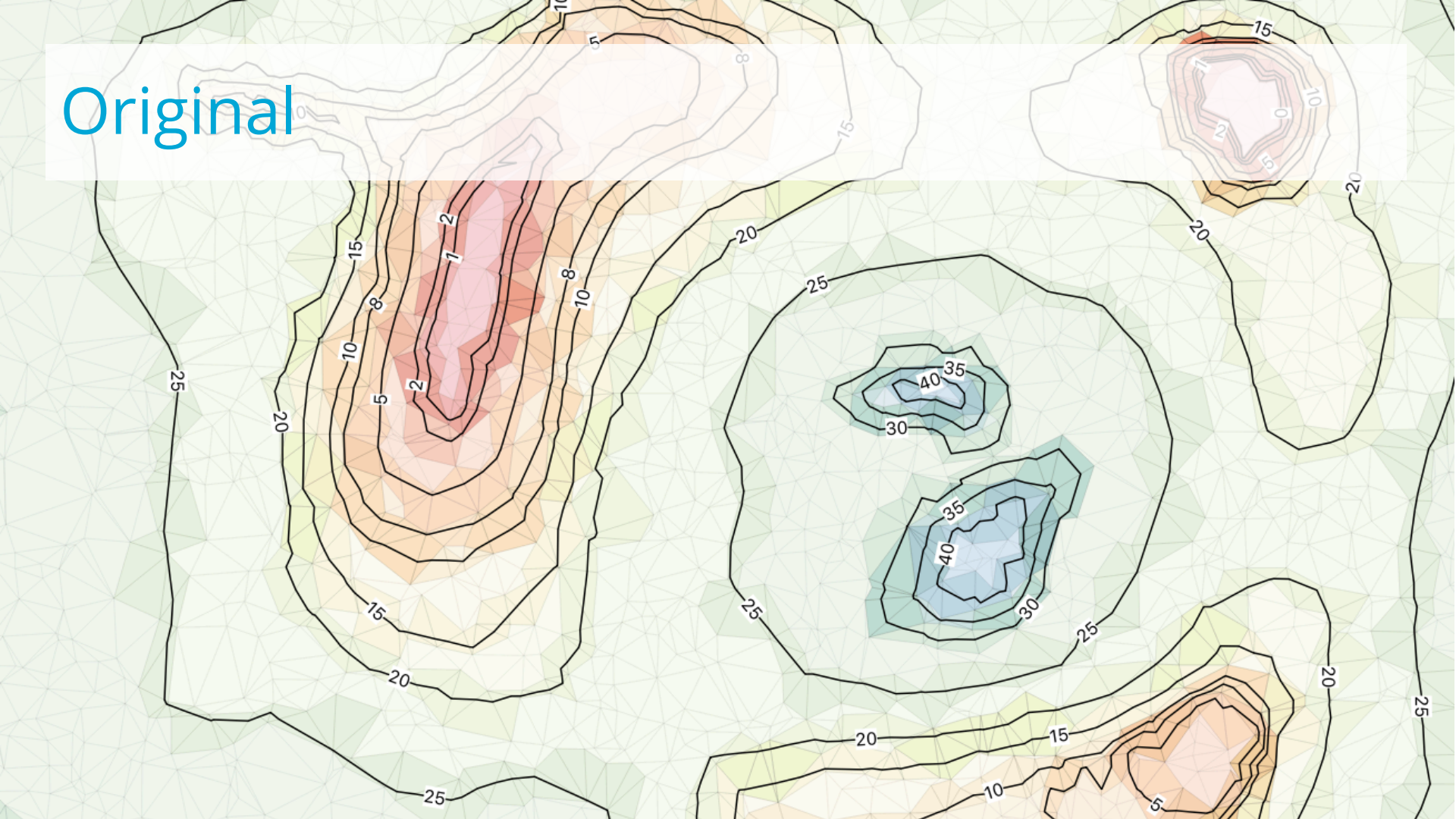
Process and implementation

- Modular approach, makes it possible to adapt the process

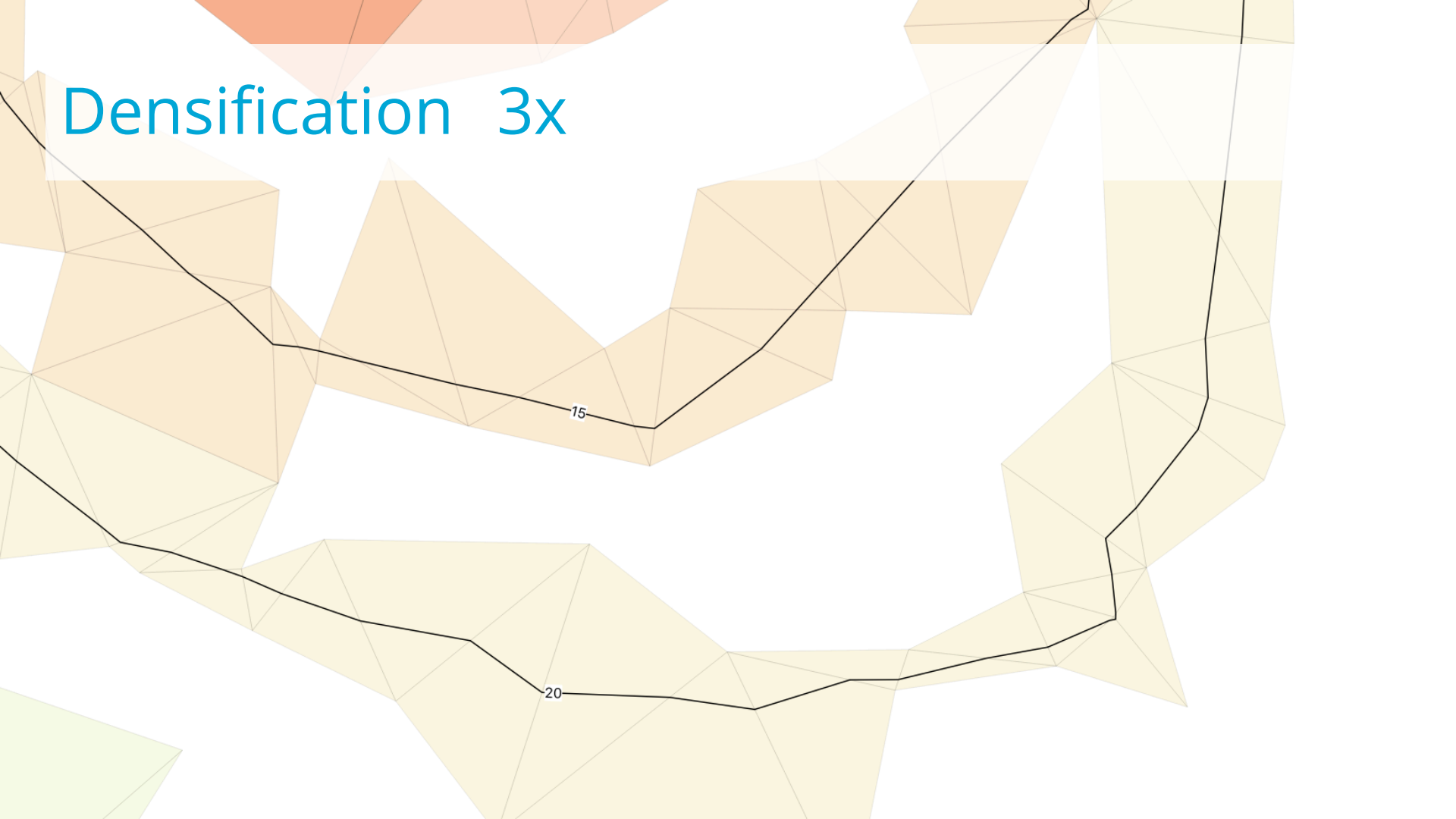
```
paramDict['process'] = [

[['angularity', 'r', 1], ['spurs', 'r', 1], ['gullys', 'r', 1],  0],
[['angularity', 'r', 2], ['spurs', 'r', 2], ['gullys', 'r', 2],  0],
[['angularity', 'r', 4],                                         3],
                        ]

paramDict['densification_process'] = [

['angularity',   'r', 1],
['aspect-edges', 'r', 0],
['size-edges',   'r', 0]
                                    ]
```

Results

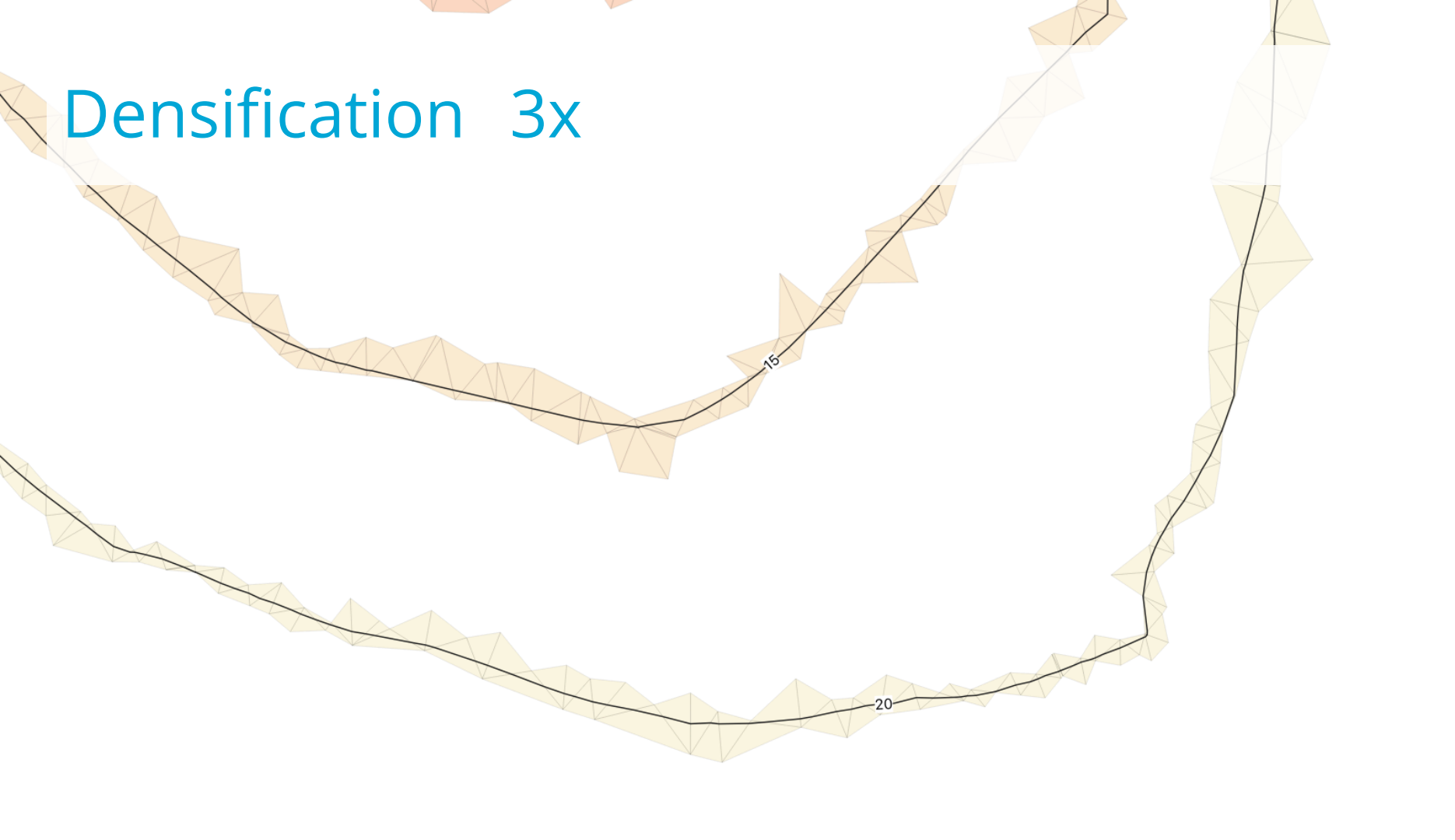
Original



Densification 3x



Densification 3x



Densification 3x