

From Points to Prints

Lastig Internal Seminar

Alexandre Bry

May 07, 2026

Context



Context	1
Roof edge points	4
Creation of the roofprints	8
Footprint points	20
Creation of the footprints	21

Context

—

	Roofprint	Footprint
Defined by	Roof	Façades
Comes from	Aerial data (images, ALS)	Terrain measurements (TLS, MLS)
Used for	3D modelling, solar potential	Tax estimation, energy consumption

Table 1: Comparison between roofprints and footprints

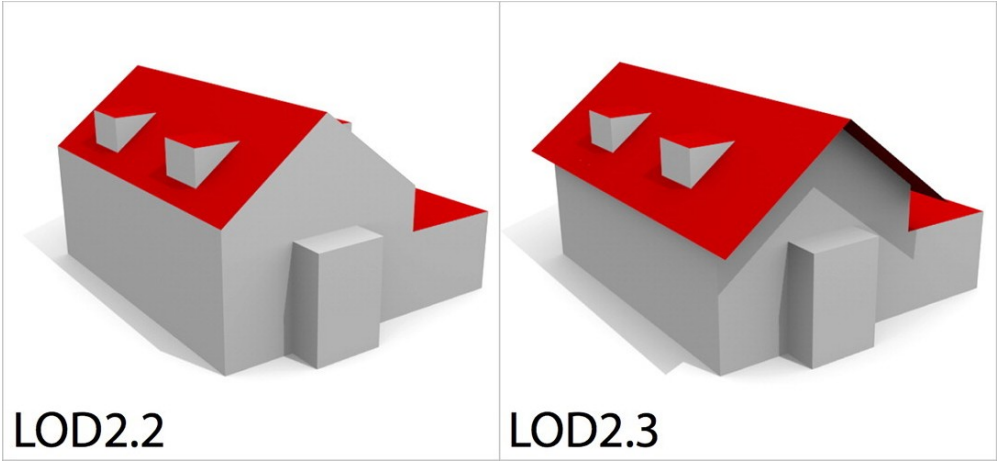


Figure 1: Part of the visual definition of buildings Level of Detail (LoD) containg LoD 2.2 and 2.3 (Biljecki et al., 2016) .

Strengths and weaknesses of BD TOPO

Context

– Strengths and weaknesses of BD TOPO

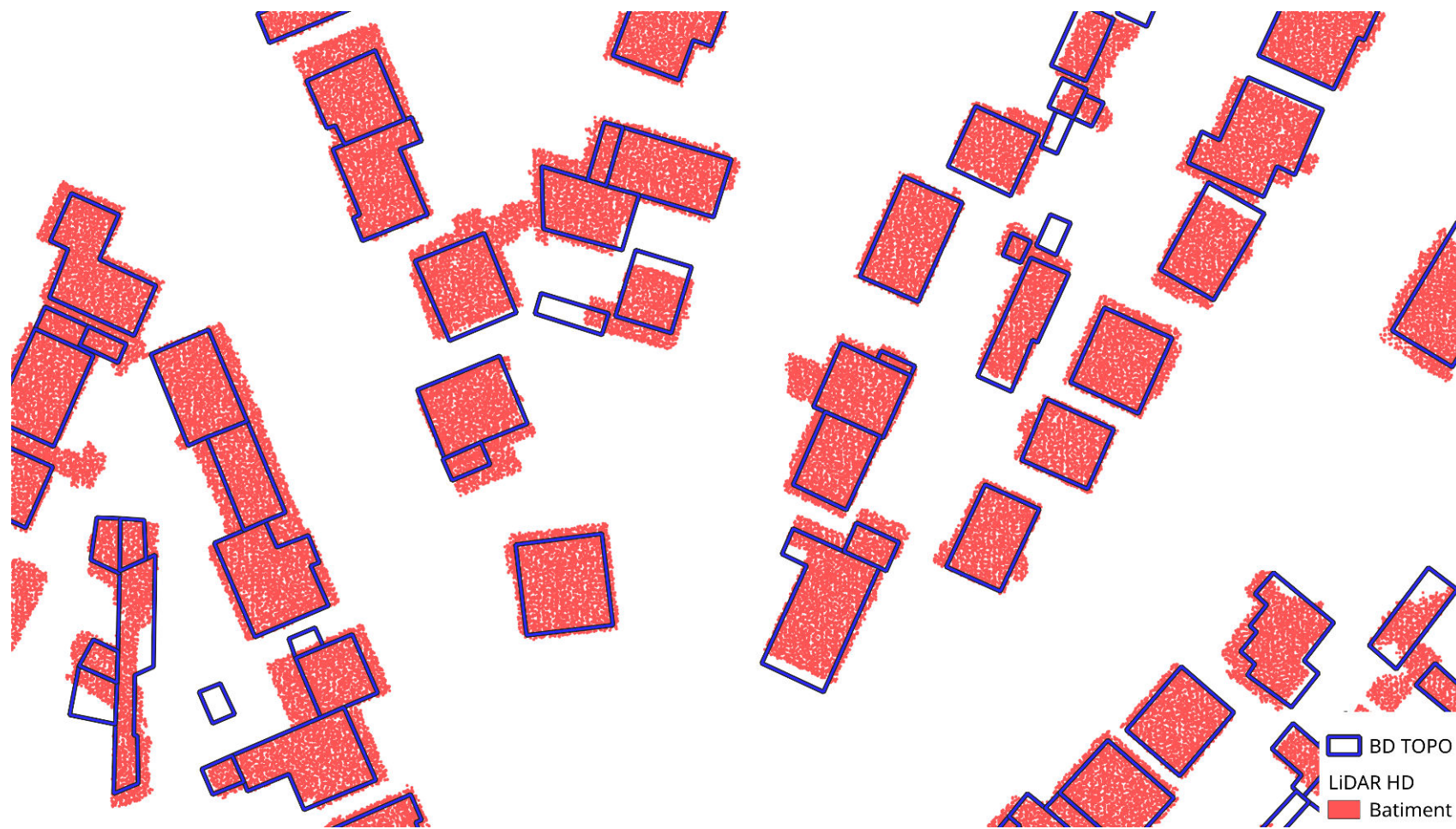


Figure 2: BD TOPO buildings are most of the time footprints, and often shifted.

*Generate for each building a **roofprint** and a **footprint** coherent with that roofprint, and by doing so estimate the **roof overhangs**.*

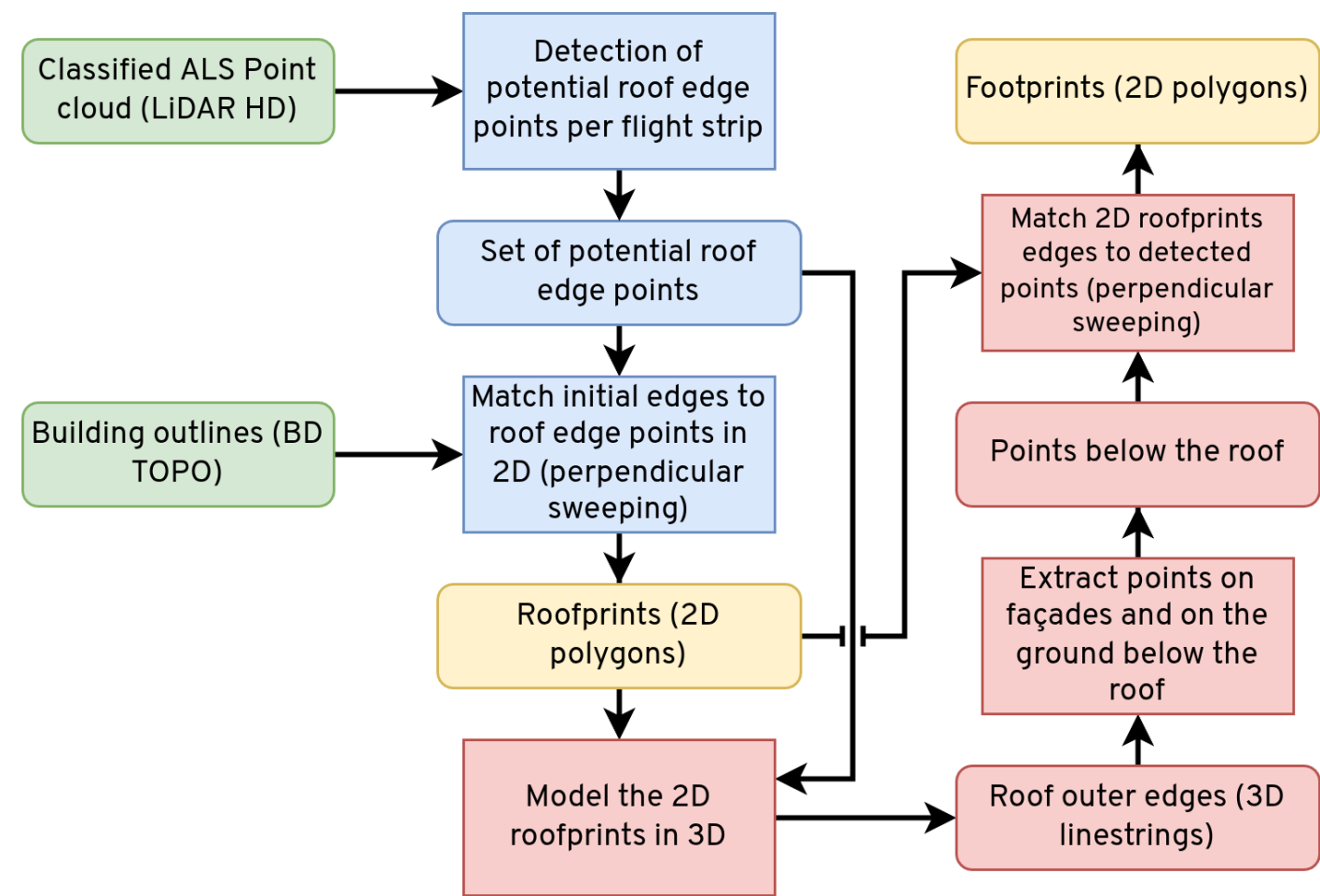


Figure 3: Overview of the pipeline.

Roof edge points



- Context 1
- Roof edge points 4
- Creation of the roofprints 8
- Footprint points 20
- Creation of the footprints 21

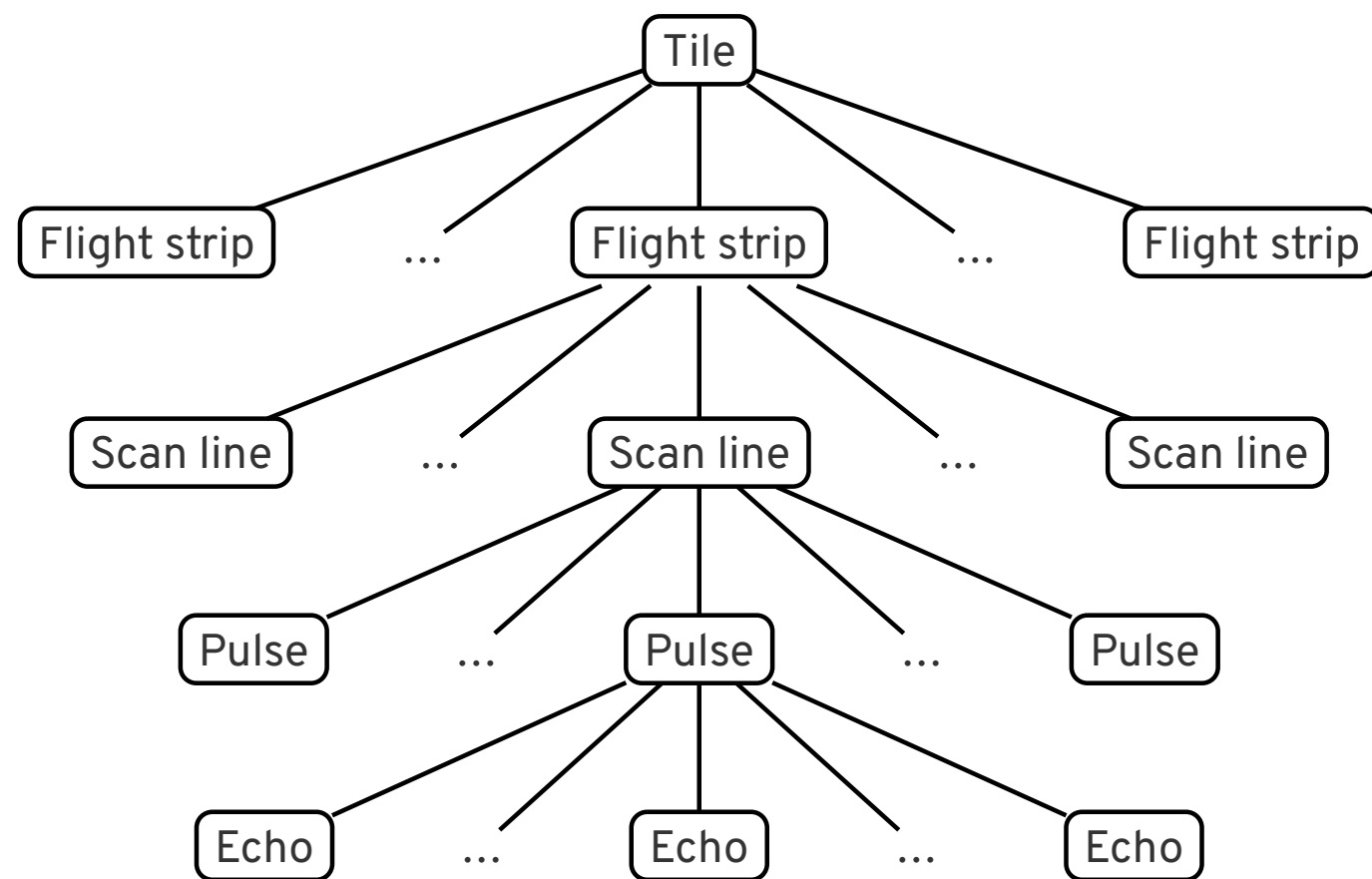


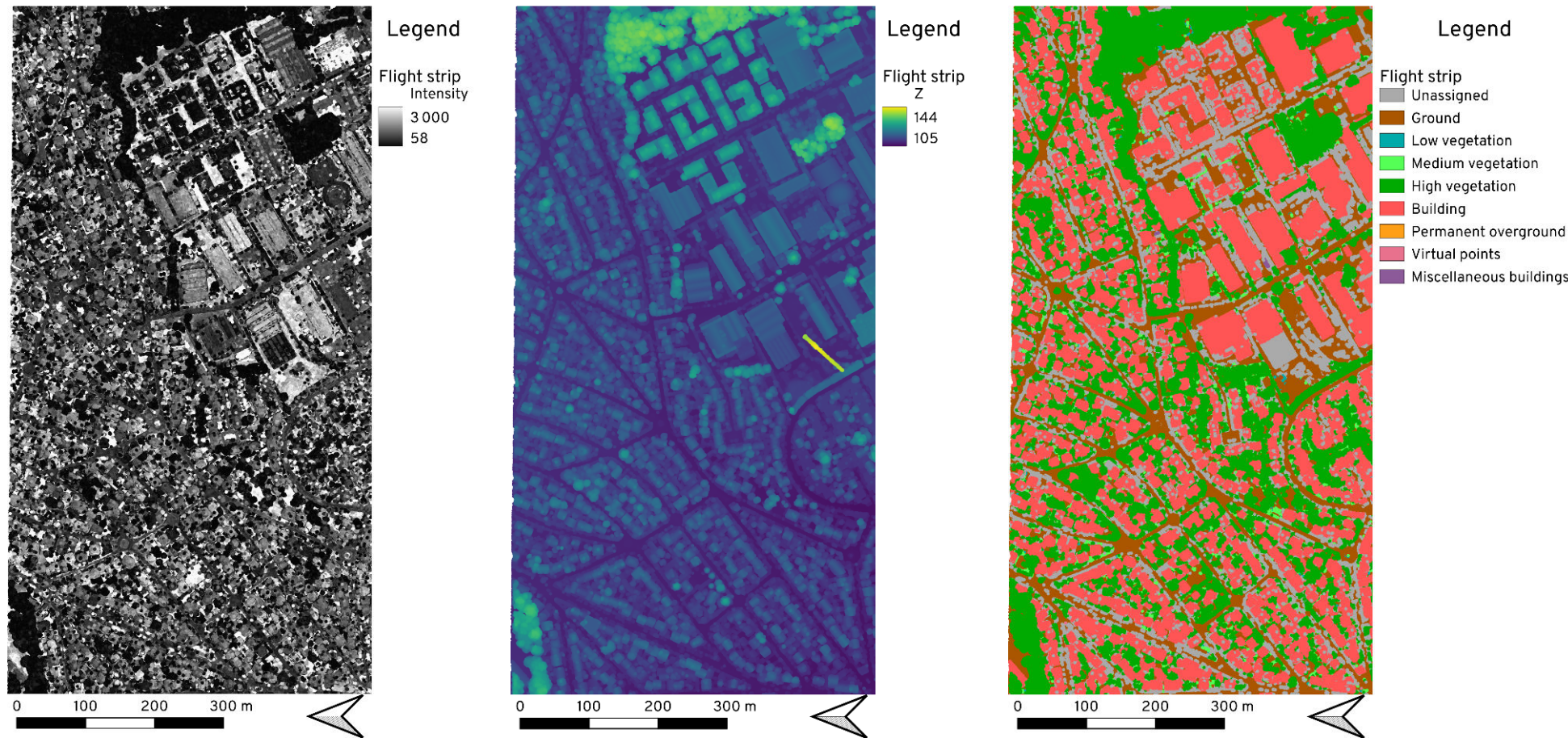
Figure 4: Illustration of the structure of an ALS point cloud.

Roof edge points – Point cloud topology

- This creates a **hierarchy** in the point cloud, with points belonging to pulses, pulses belonging to scan lines, and scan lines belonging to flight strips.
- With multiple flight strips in the same area, this creates a sort of **irregular 3D structure**:
 - 1st dimension: the flight strip
 - 2nd dimension: the scan line
 - 3rd dimension: the pulse

It is then possible to **navigate** in this structure along any of these dimensions.

Point cloud topology



(a) Points coloured by intensity.

(b) Points coloured by height.

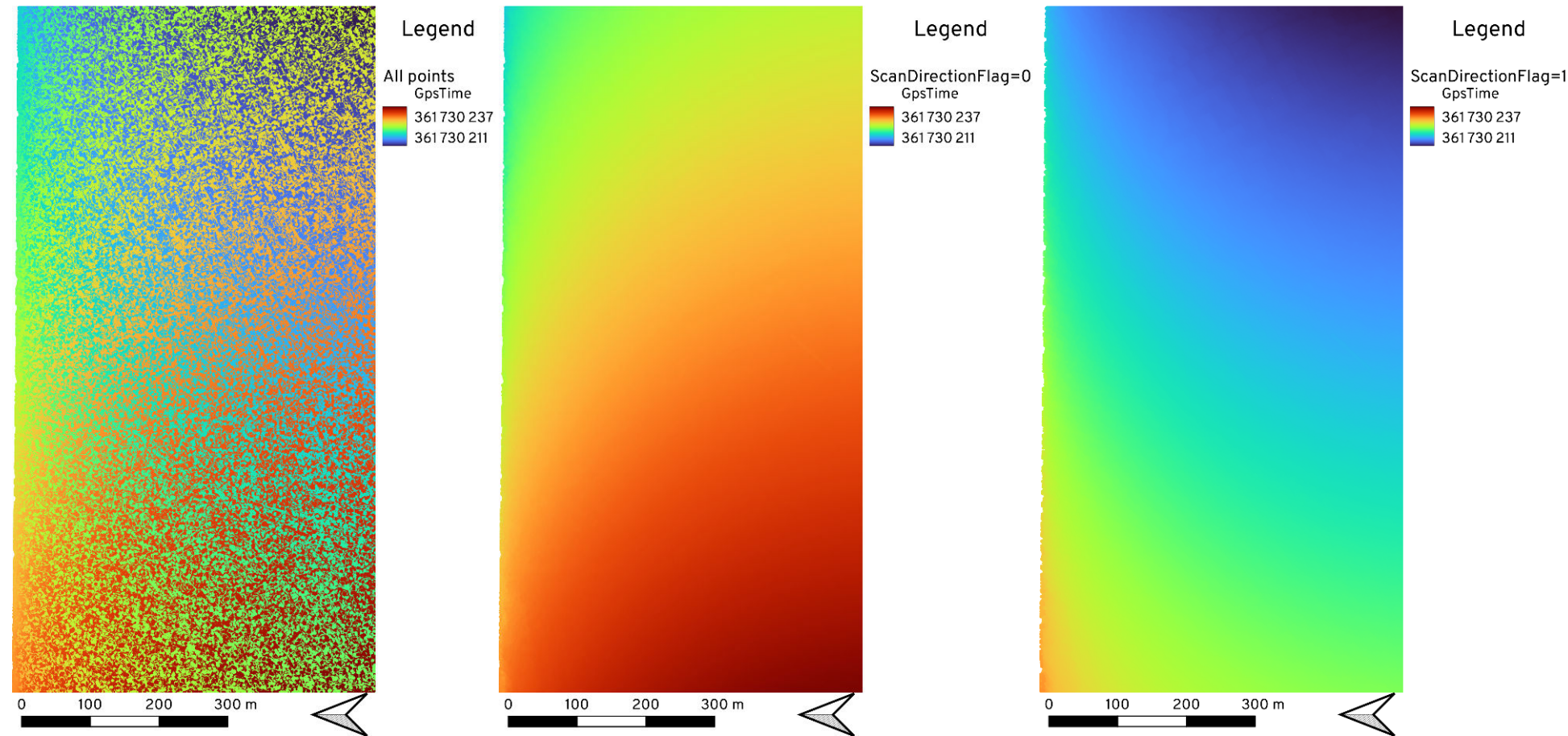
(c) Points coloured by classification.

Figure 5: Different visualizations of one flight strip.

Roof edge points – Point cloud topology

Different visualizations of a part of a flight strip, cropped to fit in a given LiDAR HD tile.

Point cloud topology



(a) All points of one flight strip.

(b) Front scan line.

(c) Back scan line.

Figure 6: Visualization of the topology of the point cloud, with points coloured by their GPS Time.

Roof edge points – Point cloud topology

Due to the ellipsoidal scanning pattern of the LiDAR scanner, the GPS Time values follow a **curved pattern**, and are mixed up when looking at the whole flight strip. However, using the **Scan Direction Flag** to separate front and back scan lines shows the distribution.

Edge points detection

We use the **height differences between consecutive points** on the same scan line to identify points that are likely to be on the edges of roofs.

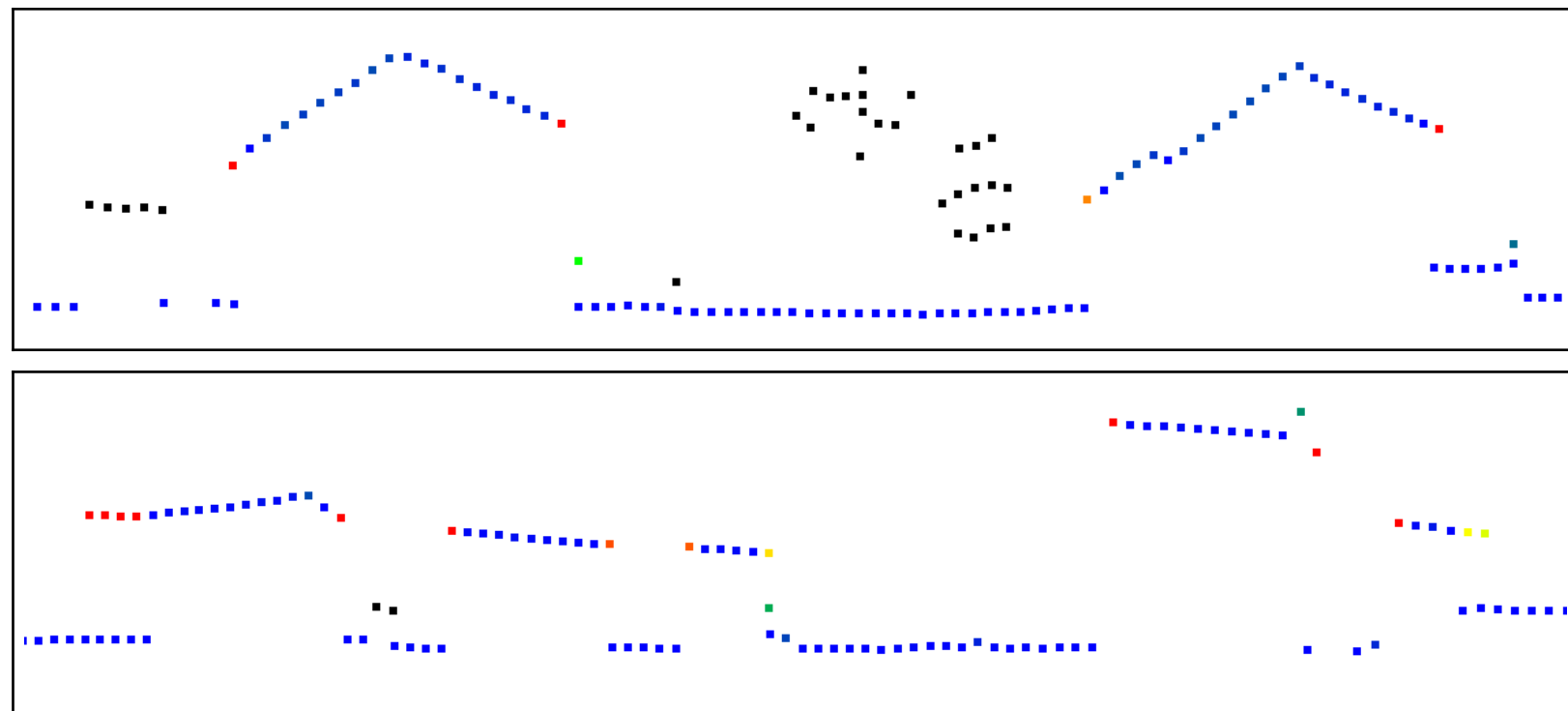


Figure 7: Two examples of the height differences obtained on a scan line. Black points are vegetation points, and for the other points the color represents the value of Δh , with blue-green-yellow-red from low to high values.

Roof edge points – Edge points detection

- Points at the edges of what looks like building roofs have **high values of Δh**
- Other points get **low values of Δh**
- Points classified as **vegetation** would have gotten high values if not excluded
- These scan lines are actually **curved** when looked at from above, but this is not a problem as the **angle is very small** between consecutive pulses

Creation of the roofprints



Context	1
Roof edge points	4
Creation of the roofprints	8
Footprint points	20
Creation of the footprints	21

Constraints over the movements

We modify the polygons by applying the following rules:

- **Never rotate an edge.**
- **Never flip an edge.**
- **Move overlapping edges together.**

Creation of the roofprints

– Constraints over the movements

Constraints over the movements

We modify the polygons by applying the following rules:

- **Never rotate an edge.**
- **Never flip an edge.**
- **Move overlapping edges together.**

This ensures that we **preserve some topology** while keeping a **lot of freedom** for the edges to move.

Creation of the roofprints

– Constraints over the movements

Constraints over the movements

We modify the polygons by applying the following rules:

- **Never rotate an edge.**
- **Never flip an edge.**
- **Move overlapping edges together.**

This ensures that we **preserve some topology** while keeping a **lot of freedom** for the edges to move.

However it is not perfect as it can **separate two points** that were initially at the same position.

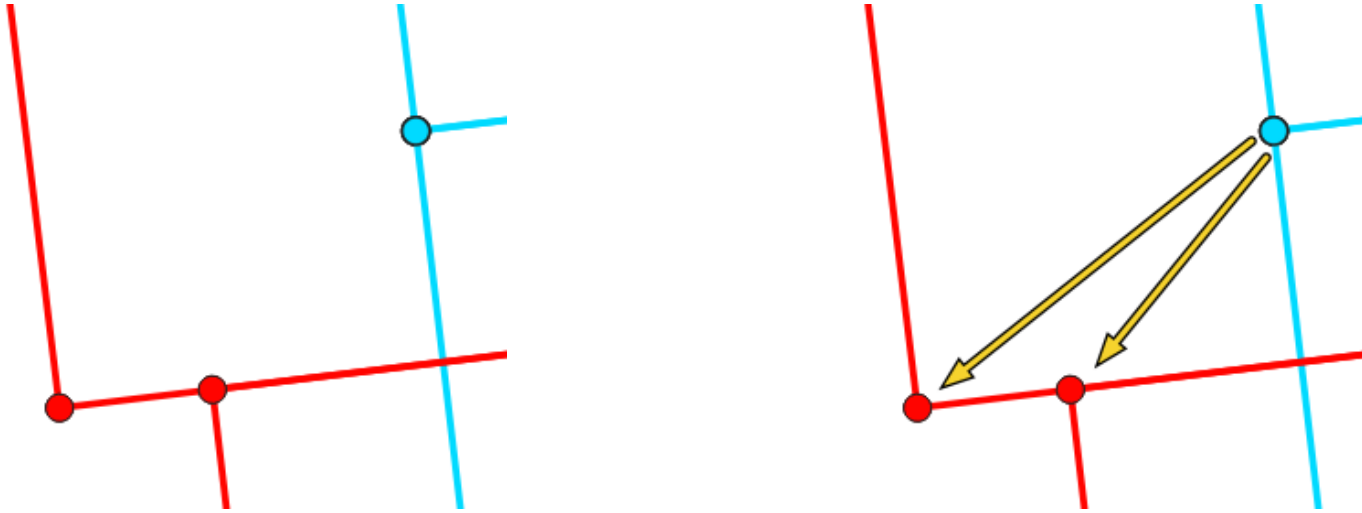


Figure 8: Two vertices at the exact same position become two distinct vertices at the boundary between two buildings.

Creation of the roofprints – Constraints over the movements

Ideas behind the polygon deformation algorithm

- Edges are considered as **infinite lines**
- Edges are **translated perpendicularly** to themselves
- As soon as an edge is **flipped**, its **neighbours are translated one by one** until there is no flip

Creation of the roofprints

– Ideas behind the polygon deformation algorithm

Illustrations of the polygon deformation algorithm

Illustrations can be viewed by following this link: https://alexandre-bry.github.io/MSc_Thesis-Report/slides_pages/2026_04_20-monthly.html

Creation of the roofprints

– Illustrations of the polygon deformation algorithm

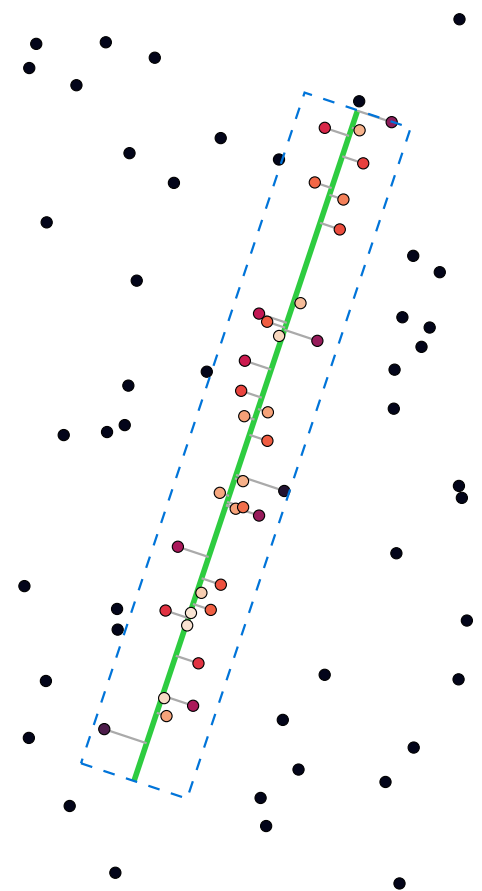


Figure 9: Illustration of the proximity score for an isolated edge.

Creation of the roofprints
– Criterion to score configurations

Criterion to score configurations

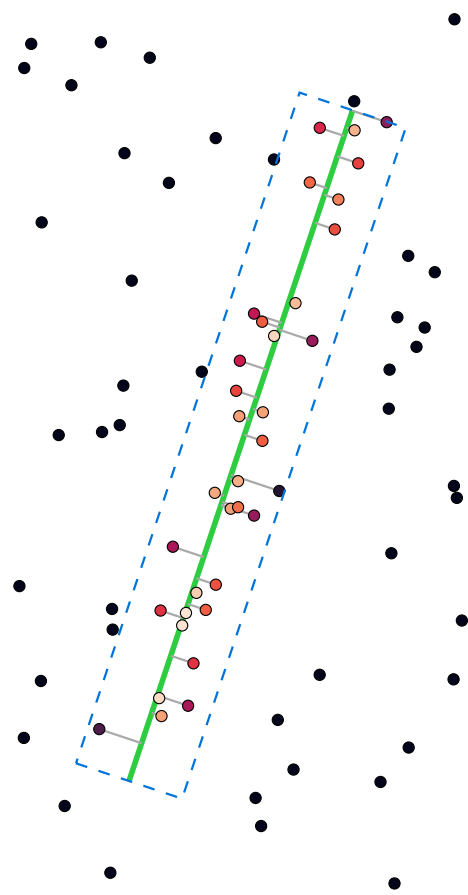


Figure 9: Illustration of the proximity score for an isolated edge.

We minimize the sum of an **energy** of all edges. The energy is defined with:

1. a **proximity term** → counts the number of points close to the edges
2. a **regularization term** → keeps edges close to their initial length

Creation of the roofprints
– Criterion to score configurations

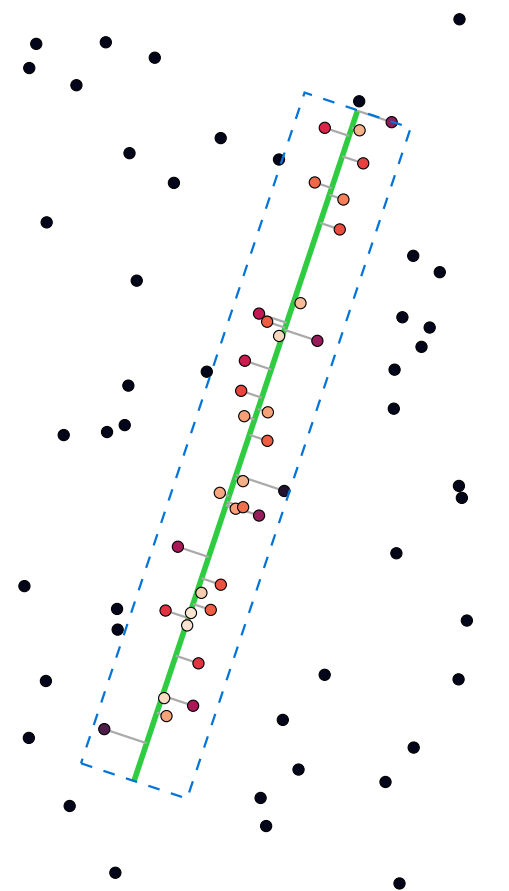


Figure 9: Illustration of the proximity score for an isolated edge.

We minimize the sum of an **energy** of all edges. The energy is defined with:

1. a **proximity term** → counts the number of points close to the edges
2. a **regularization term** → keeps edges close to their initial length

The proximity term contains:

1. a **weight** for each point,
2. a score based on the **distance** of the point to the edge
3. a score based on the **alignment** of the point with the edge

Illustrations of matching the edges to the points

Illustrations can be viewed by following this link: https://alexandre-bry.github.io/MSc_Thesis-Report/slides_pages/2026_04_20-monthly.html

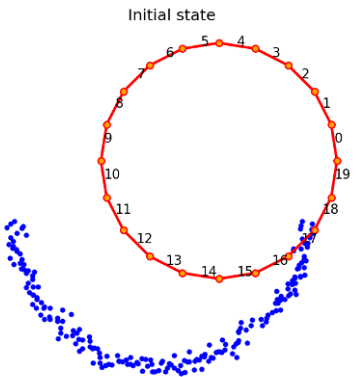
Creation of the roofprints

– Illustrations of matching the edges to the points

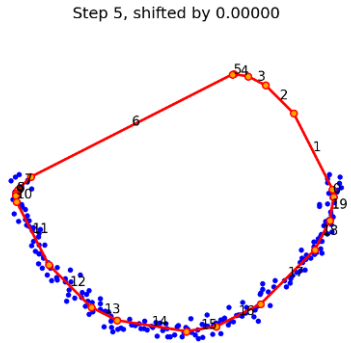
Illustrations of matching the edges to the points

Creation of the roofprints
– Illustrations of matching the edges to the points

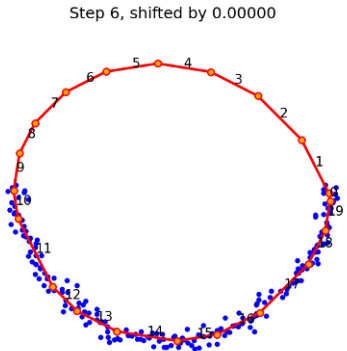
We can see how regularization helps to get a shape more similar to the target even with half of the points missing.



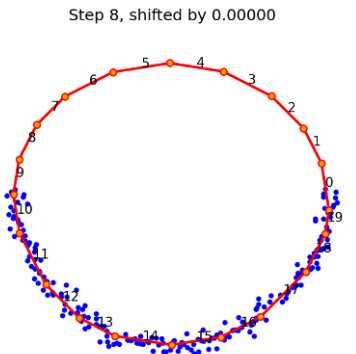
(a) Initial state



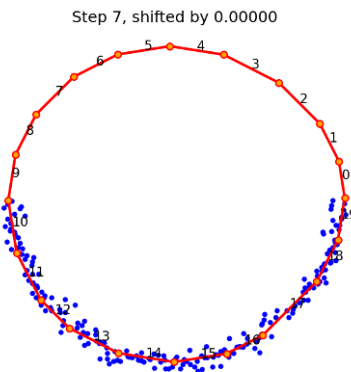
(b) $\alpha = 0.00$ (no regularization)



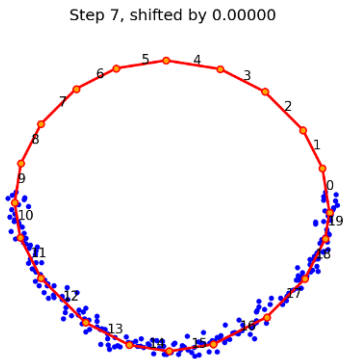
(c) $\alpha = 0.05$



(d) $\alpha = 0.20$



(e) $\alpha = 0.50$



(f) $\alpha = 1.00$

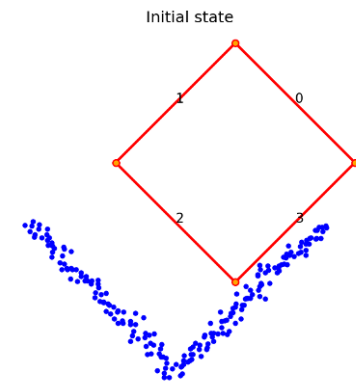
Figure 10: Matching a circle to a half-circle of points with different values of the regularization parameter α .

Illustrations of matching the edges to the points

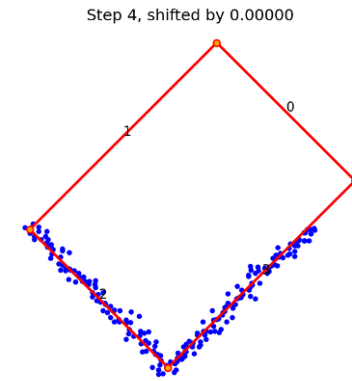
Creation of the roofprints

– Illustrations of matching the edges to the points

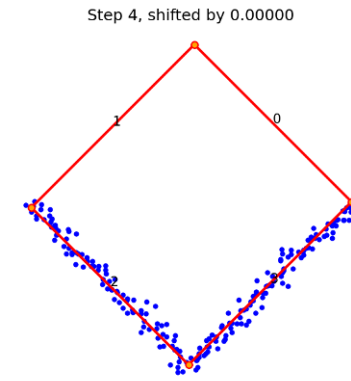
$\alpha > 0$ encourages the shape to be closer to the initial one, which is better than $\alpha = 0$, until a certain point where the regularization is too strong and prevents the shape from extending to capture all the points.



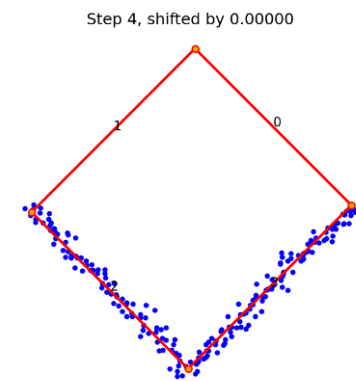
(a) Initial state



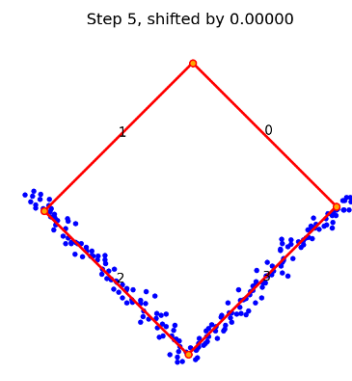
(b) $\alpha = 0.00$ (no regularization)



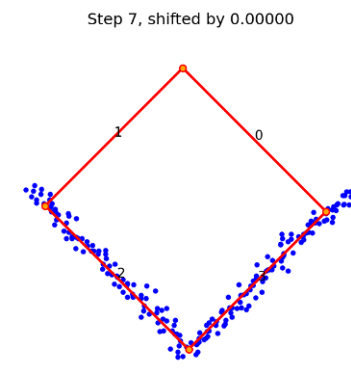
(c) $\alpha = 0.05$



(d) $\alpha = 0.20$



(e) $\alpha = 0.50$



(f) $\alpha = 1.00$

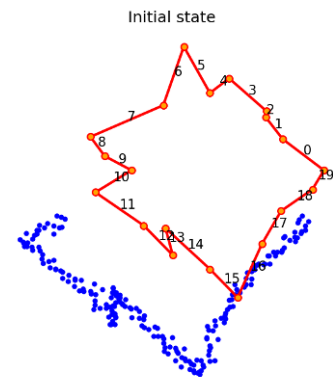
Figure 11: Matching a square to points along two sides of a larger square with different values of the regularization parameter α .

Illustrations of matching the edges to the points

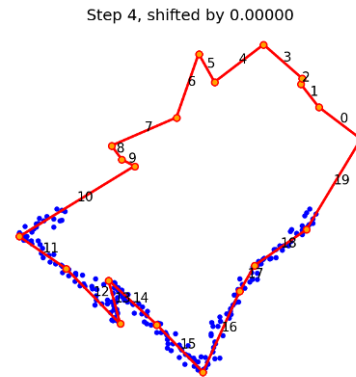
Creation of the roofprints

– Illustrations of matching the edges to the points

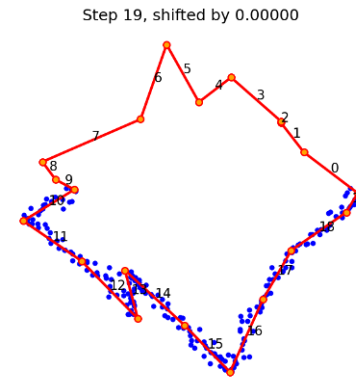
- $\alpha > 0$ forces the shape to be closer to the initial one, but it takes many more steps to converge, because the more edges that don't have points take longer to balance the regularization term between each other.
- If α is too high, it prevents the shape from matching the points, because the proximity score is not good enough to balance the regularization term, which is the case here for $\alpha \geq 0.50$.



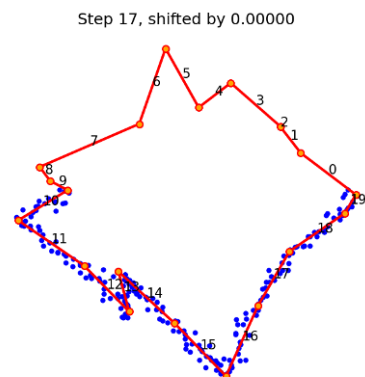
(a) Initial state



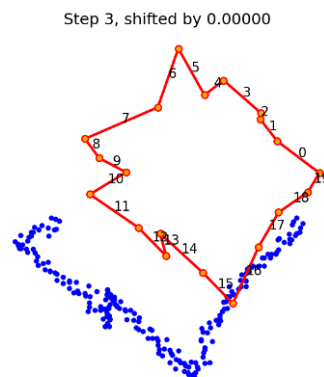
(b) $\alpha = 0.00$ (no regularization)



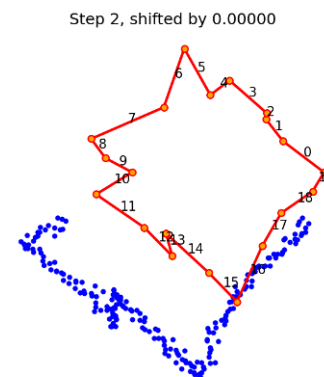
(c) $\alpha = 0.05$



(d) $\alpha = 0.20$



(e) $\alpha = 0.50$



(f) $\alpha = 1.00$

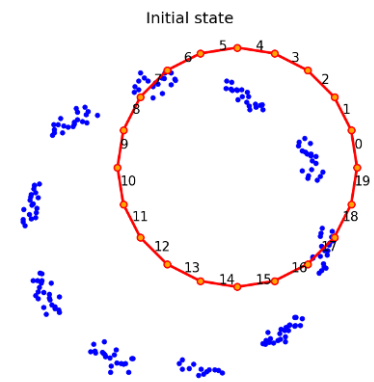
Figure 12: Matching a polygon to half of its shape with points with different values of the regularization parameter α .

Illustrations of matching the edges to the points

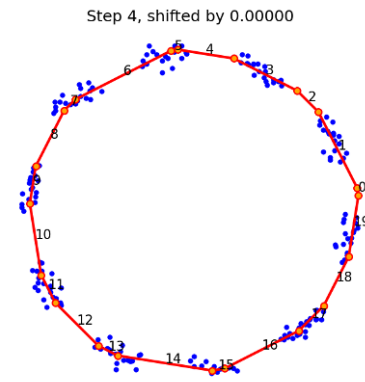
Creation of the roofprints

– Illustrations of matching the edges to the points

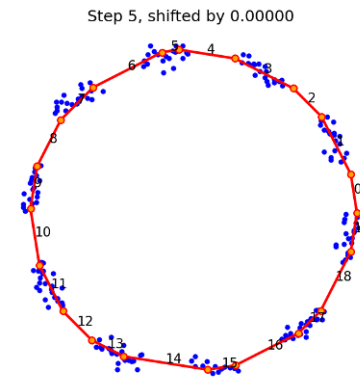
At the cost of more iterations, a larger α outputs a more regular shape, closer in proportions to the initial one.



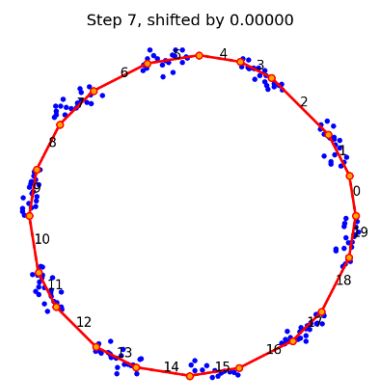
(a) Initial state



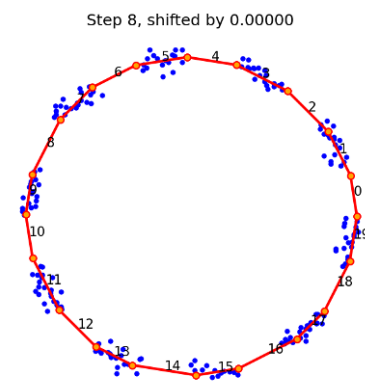
(b) $\alpha = 0.00$ (no regularization)



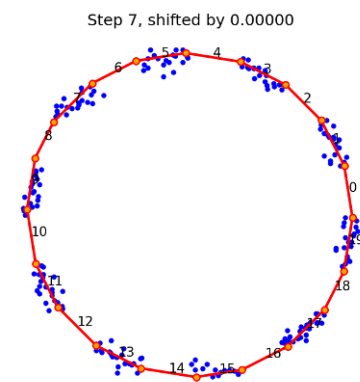
(c) $\alpha = 0.05$



(d) $\alpha = 0.20$



(e) $\alpha = 0.50$



(f) $\alpha = 1.00$

Figure 13: Matching a circle to points along half of its boundary with different values of the regularization parameter α .

Illustration with a real building

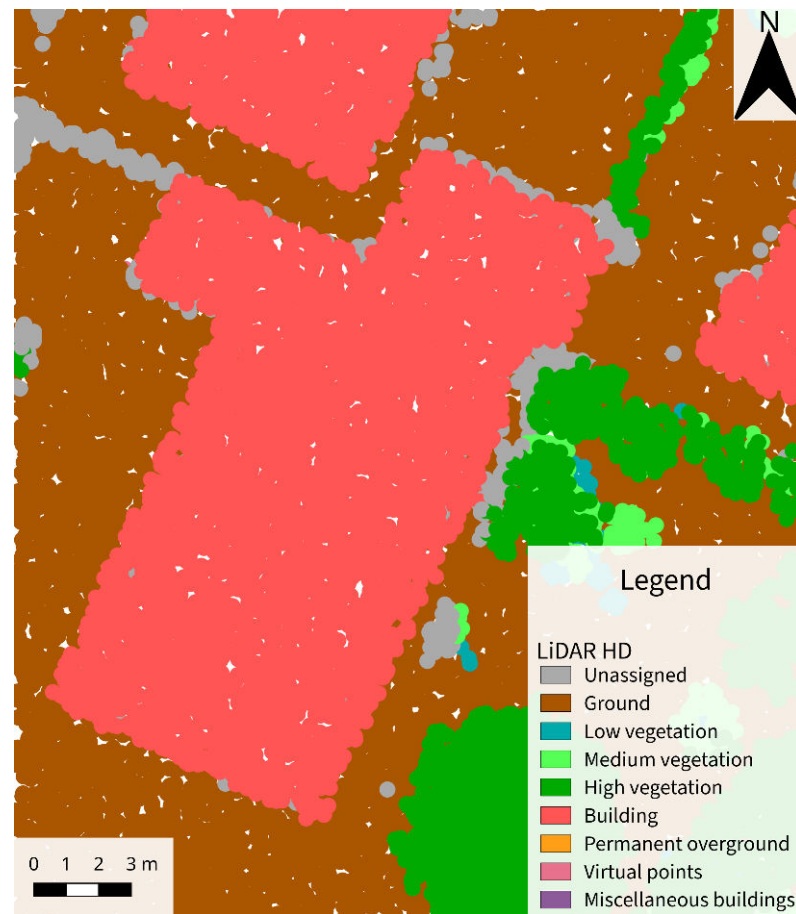
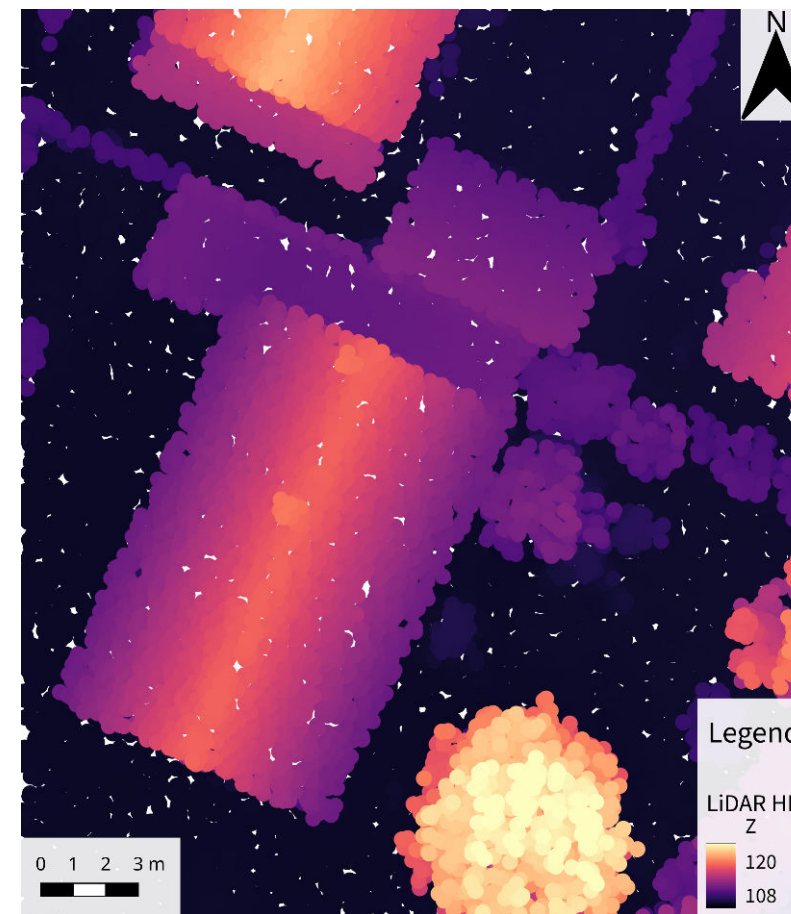


Figure 14: The LiDAR HD data.



Creation of the roofprints

– Illustration with a real building

- We can see **three different building parts** that are connected, one with a pitched roof and the two others with flat roofs.
- The points are well classified overall, but there are a number of points **classified as Unassigned (grey)**, especially at the boundaries between the building and the vegetation.

Illustration with a real building

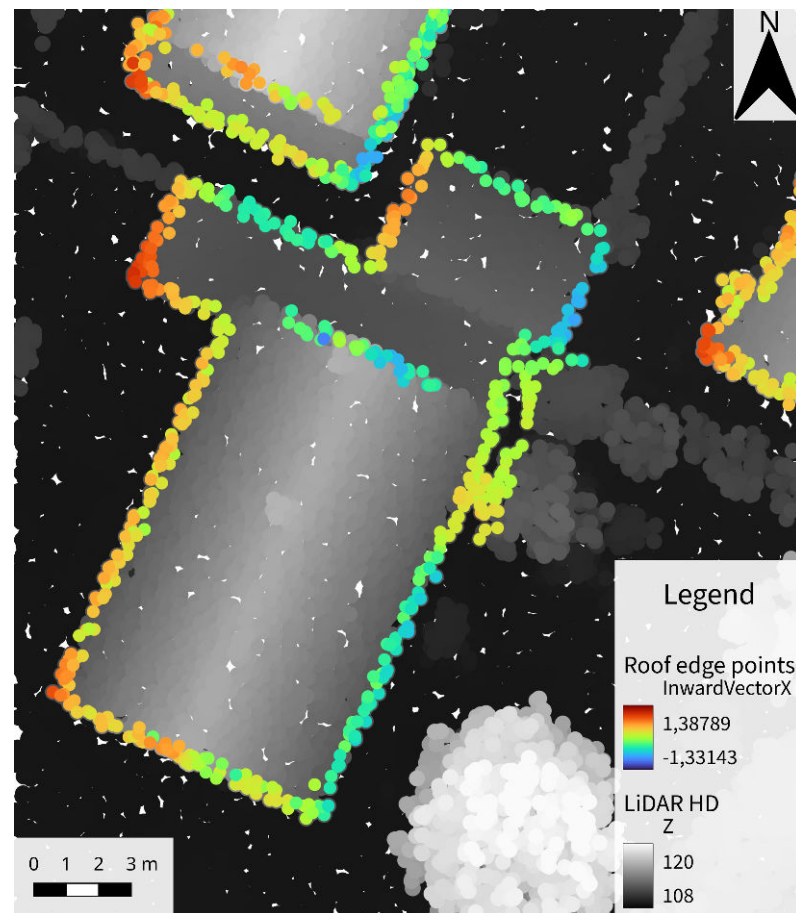
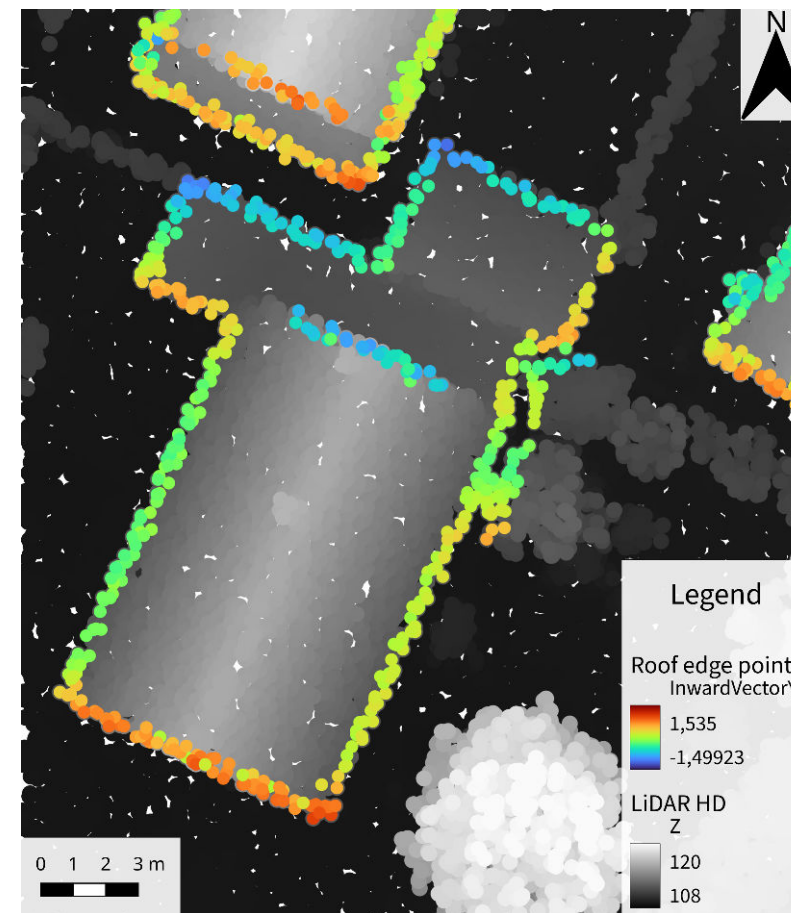


Figure 15: The detected roof edge points coloured by their inward vector.



Creation of the roofprints

– Illustration with a real building

- The detected roof edge points are shown here, coloured by their **inward vector** (the direction in which the building is located from the edge).
- We can see that the inward vectors are **generally well oriented** towards the building, except in areas where the building is **very close to vegetation** at the same height.

Illustration with a real building

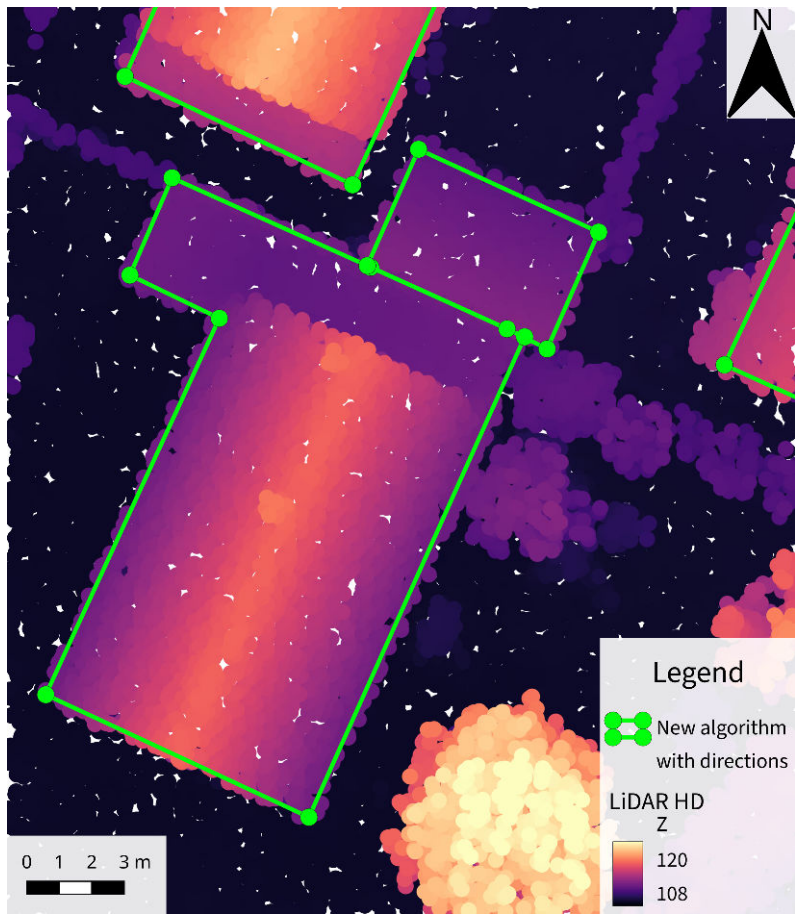
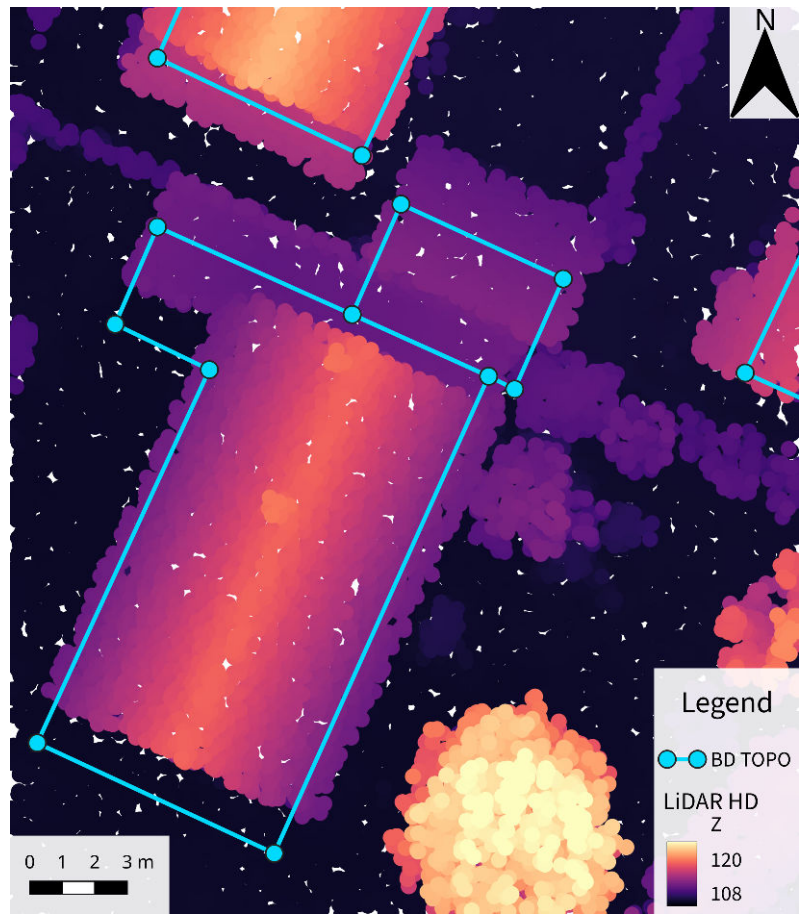


Figure 16: Comparison of BD TOPO outlines and roofprints computed with my method.

Creation of the roofprints
– Illustration with a real building

Footprint points



Context	1
Roof edge points	4
Creation of the roofprints	8
Footprint points	20
Creation of the footprints	21

Footprint points

1. Move the 2D roofprints to 3D creating a **3D roof outline**
2. Look for points **under** the 3D roof outline and **towards the inside** of the building

Footprint points – Footprint points

Footprint points

1. Move the 2D roofprints to 3D creating a **3D roof outline**
2. Look for points **under** the 3D roof outline and **towards the inside** of the building

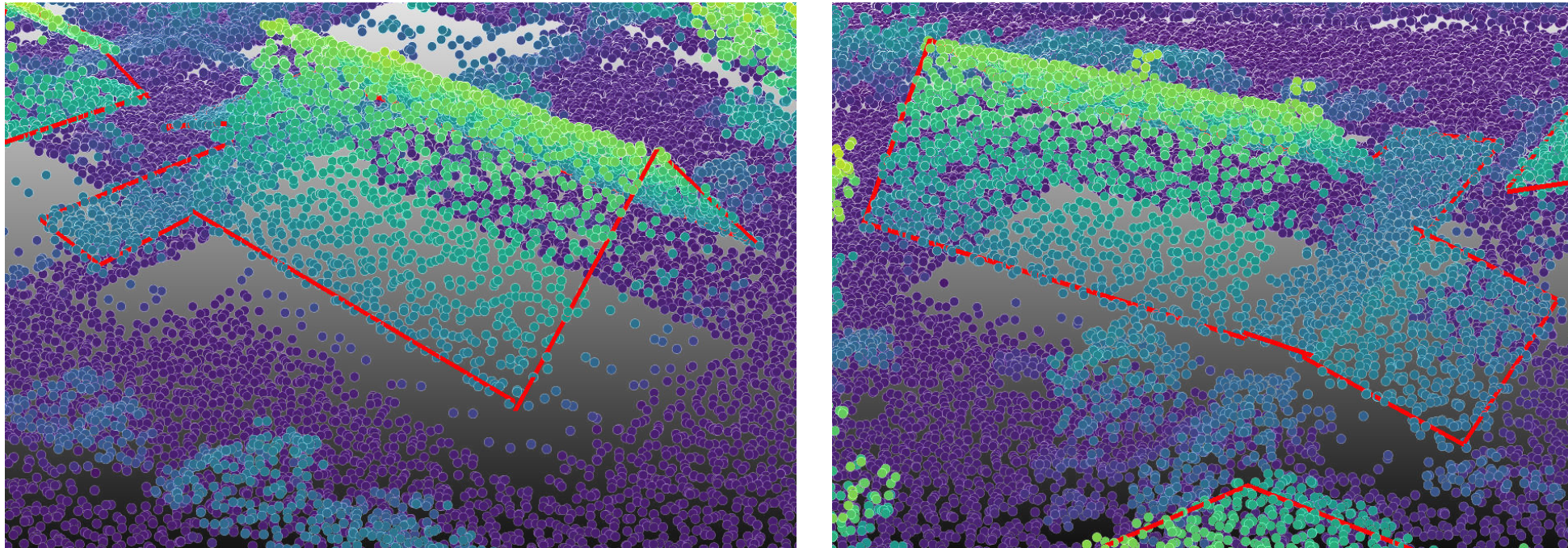


Figure 23: Illustration of the 3D roof outline.

Footprint points – Footprint points

Footprint points

1. Move the 2D roofprints to 3D creating a **3D roof outline**
2. Look for points **under** the 3D roof outline and **towards the inside** of the building

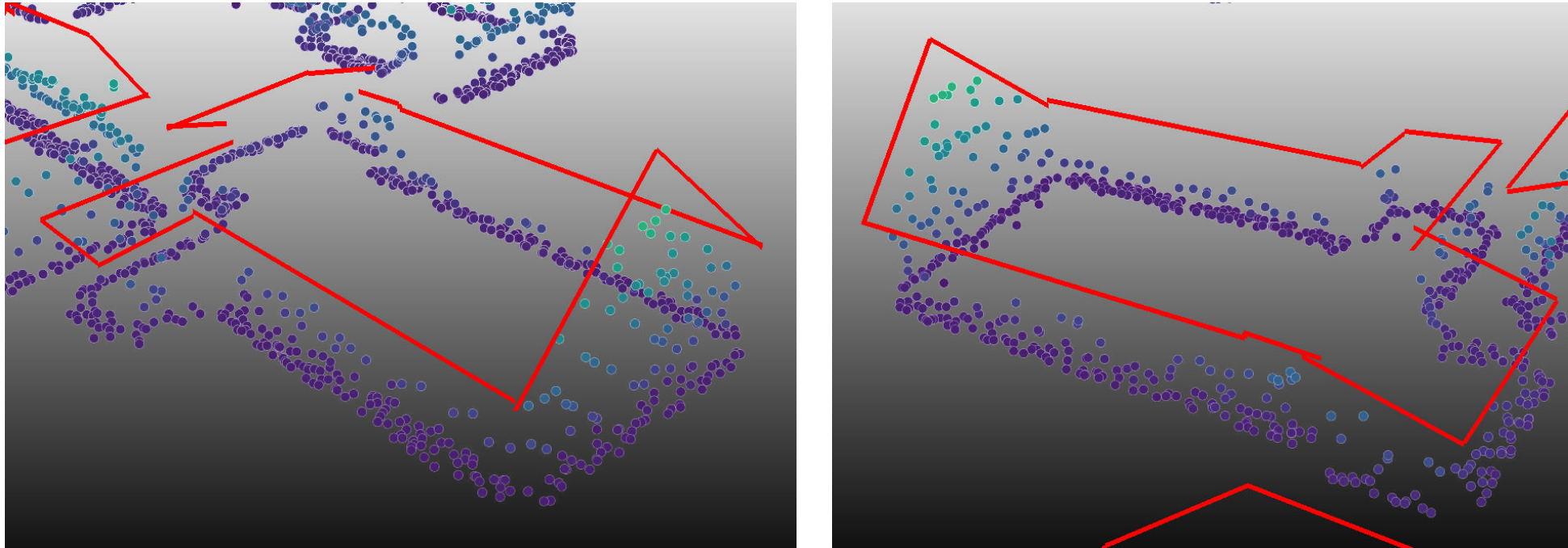


Figure 23: Illustration of the footprint points.

Footprint points – Footprint points

Creation of the footprints



Context	1
Roof edge points	4
Creation of the roofprints	8
Footprint points	20
Creation of the footprints	21

Creation of the footprints

- Sweep each roof edge towards the inside up to 1.5 metres and select the best scoring position

Creation of the footprints – Creation of the footprints

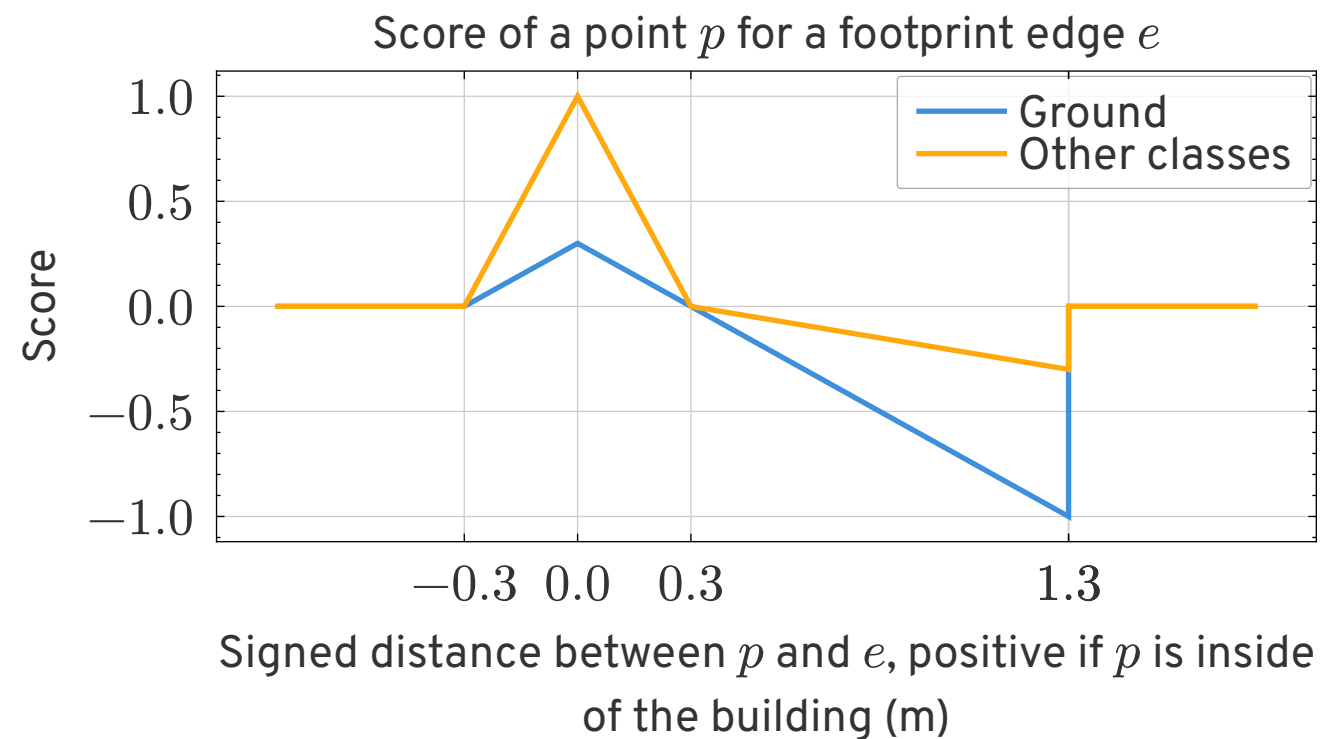
Creation of the footprints

- Sweep each roof edge towards the inside up to 1.5 metres and select the best scoring position
- The score has two components:
 1. A **proximity term** that counts the number of points close to the edge
 2. A **penalty term** that penalizes points that are behind the edge (inside the building)

Creation of the footprints – Creation of the footprints

Creation of the footprints

- Sweep each roof edge towards the inside up to 1.5 metres and select the best scoring position
- The score has two components:
 1. A **proximity term** that counts the number of points close to the edge
 2. A **penalty term** that penalizes points that are behind the edge (inside the building)



Creation of the footprints – Creation of the footprints

Creation of the footprints

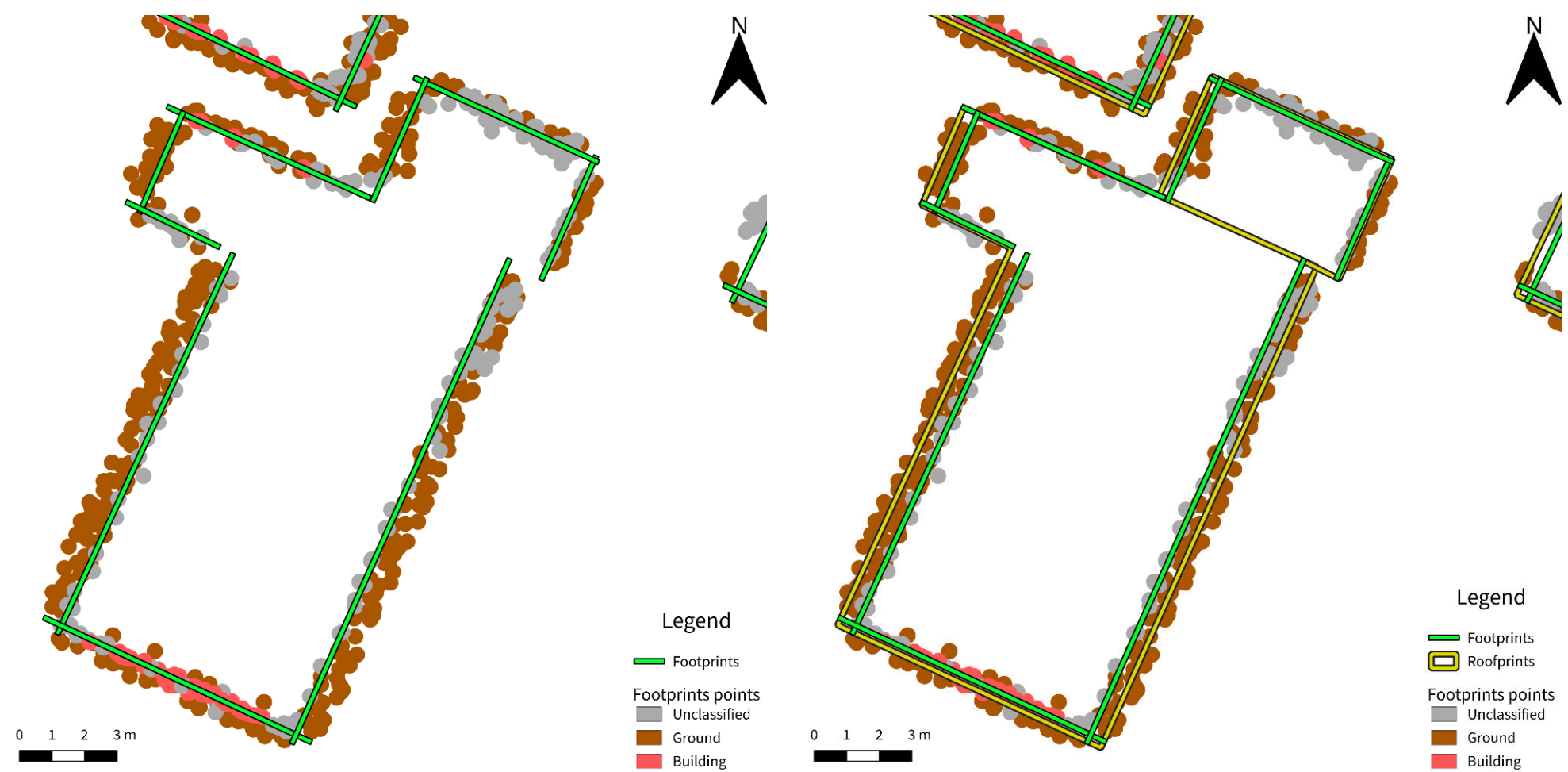


Figure 23: Illustration of the computed footprints.

Creation of the footprints

– Creation of the footprints

Creation of the footprints

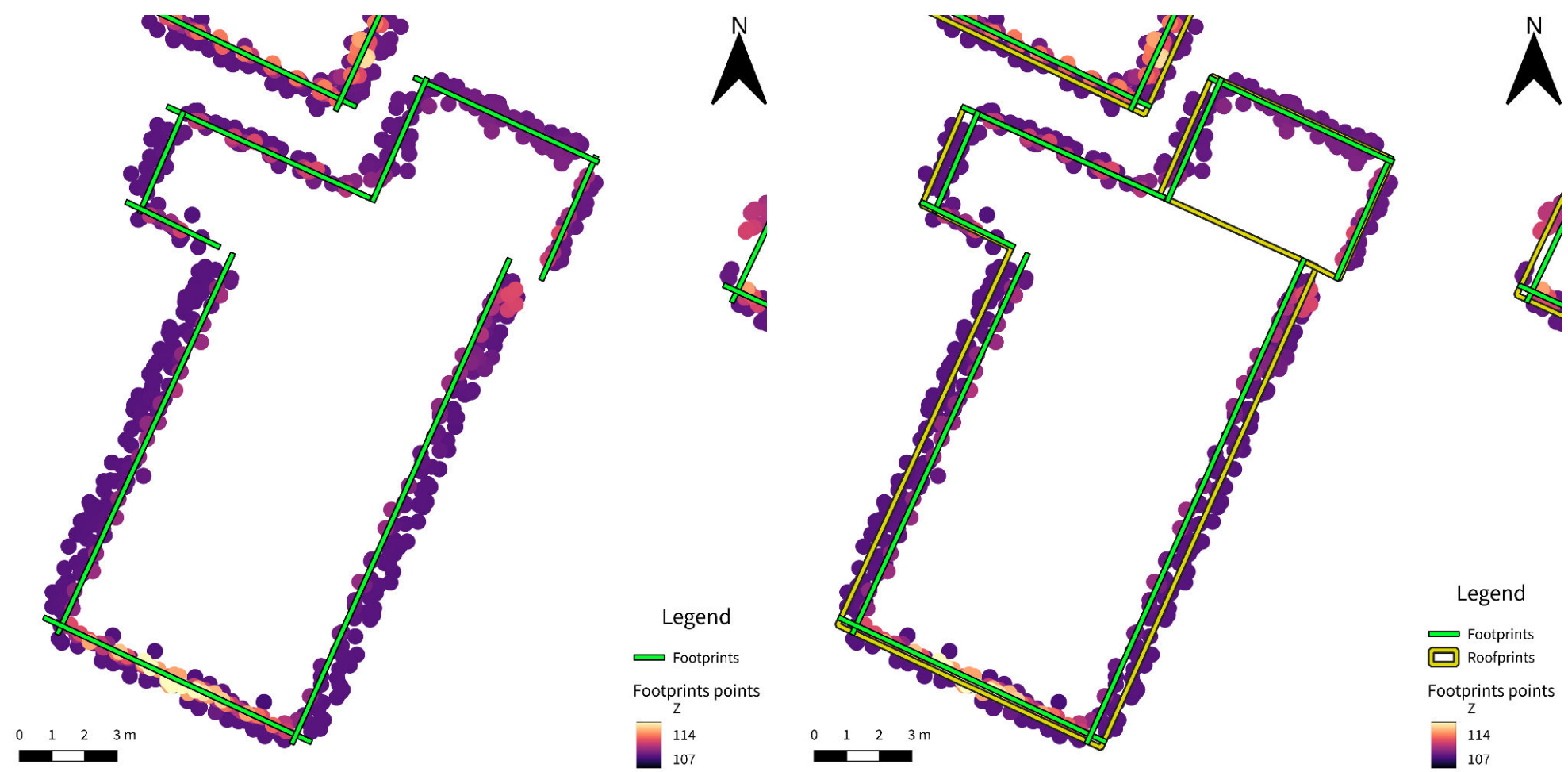


Figure 23: Illustration of the computed footprints.

Creation of the footprints

– Creation of the footprints

Thank you for your attention!

Any questions?

alexandre.bry@ign.fr

The end

—

References

Biljecki, F., Ledoux, H., & Stoter, J. (2016). An improved LOD specification for 3D building models. *Computers, Environment and Urban Systems*, 59, 25–37. <https://doi.org/10.1016/j.compenvurbsys.2016.04.005>

The end
—

Appendix



Context	1
Roof edge points	4
Creation of the roofprints	8
Footprint points	20
Creation of the footprints	21

Total energy to minimize

The energy that we try to minimize for each group of roofprints is the following:

$$E = \underbrace{- \sum_{i \in \mathcal{P}} w_i \max_{j \in \mathcal{L}} \{\text{score}(p_i, l_j)\}}_{\text{proximity to the points}} + \alpha \underbrace{\sum_{j \in \mathcal{L}} (|l_j| - |l_j^0|)^2}_{\text{similarity to the initial edges}}$$

Where:

- $\mathcal{P}$ is the set of points, and $\mathcal{L}$ is the set of edges (lines). $\mathcal{L}$ contains all the edges moved in any configuration and their neighbours.
- w_i is the weight of point p_i , which is independent of the lines.
- $\text{score}(p_i, l_j)$ is the score of point p_i for edge l_j , which is defined in the next slide.
- $|l_j|$ is the length of edge l_j in the current configuration, and $|l_j^0|$ is its initial length.
- α is a parameter to adjust between the two terms.

Appendix

– Total energy to minimize

- The energy is divided in two terms:
 - The first term is a **proximity term** that both counts the number of points close to the edges while prioritizing points with higher weights.
 - The second term is a **regularization term** that entices the edges to keep their initial length, to prioritize a shape closer to the initial one.
- Many regularization terms are possible, but we chose this one because it makes sure that by default, the edges will **keep their initial length** and otherwise **share the change equally** between them. This is desired if we assume that the scaling that we need to perform is the same on both sides for a given direction, no matter the size of the edges, because the roof overhang is the same.

Definition of the proximity score

The score of a point p_i for an edge l_j is defined as follows:

- The score is 0 if any of the following conditions is true:
 - The orthogonal projection $p_{i\perp j}$ of p_i on the line supporting of l_j is outside of the segment l_j .
 - The distance from p_i to $p_{i\perp j}$ is greater than a certain threshold ε .
 - The dot product of the inward vector v_i of p_i with the normal n_j of l_j oriented towards the inside of the building is negative.
- Otherwise, the score is:

$$\text{score}(p_i, l_j) = \underbrace{(v_i \cdot n_j)}_{\text{alignment of point and edge 'normals'}} \times \underbrace{\left(1 - \frac{|p_i - p_{i\perp j}|}{\varepsilon}\right)}_{\text{proximity to the edge}} \geq 0$$

We use $\varepsilon = 30$ centimetres.

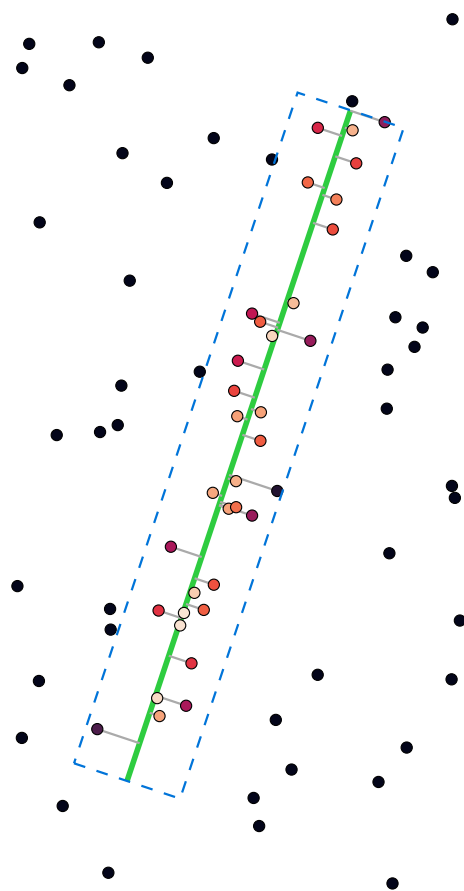


Figure 23: Illustration of the proximity score for an isolated edge.

Appendix

– Definition of the proximity score

- Any point **outside the rectangle** defined by extruding the edge along its normal by ε has a score of 0. This geometric ensures that only points that are close enough to the edge will count.
- Then, the dot product between the inward vector and the normal of the edge ensures that points that are part of a parallel but opposite edge will not count, **preventing matching to the neighbour building.**

Definition of the inward direction

The goal of the **inward direction** is to be as close as possible to the 2D normal of the roof edge, pointing towards the inside of the building. To compute it for a given point p_i , we use the following method:

1. Identify all the points p_j that are less than a certain distance δ from p_i .
2. For each point p_j , compute the vector from p_i to p_j , and normalize it into a unit vector $u_{i \rightarrow j}$.
3. Compute the inward vector v_i as the average of the vectors $u_{i \rightarrow j}$:

$$v_i = \frac{1}{|\mathcal{B}(p_i, \delta)|} \sum_{p_j \in \mathcal{B}(p_i, \delta)} \frac{p_j - p_i}{|p_j - p_i|}$$

We use $\delta = 2$ metres.

Appendix

– Definition of the inward direction

- The idea behind this method is that points on the edge of the roof should have **many points towards the inside** of the building, and **few points towards the outside**.
- Using unit vectors and averaging them allows for all points in the neighbourhood to contribute equally, meaning that we are somewhat counting the **density of points** in each direction. Therefore, the **magnitude** of the inward vector is an indication of the confidence of the direction.