From Points to Prints

Monthly Meeting (4)

Alexandre Bry

May 18, 2026

TU Delft

IGN

Context

Context 1
Roof edge points 4
Creation of the roofprints 8
Footprint points 24
Creation of the footprints 25

Roofprint vs. footprint

	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

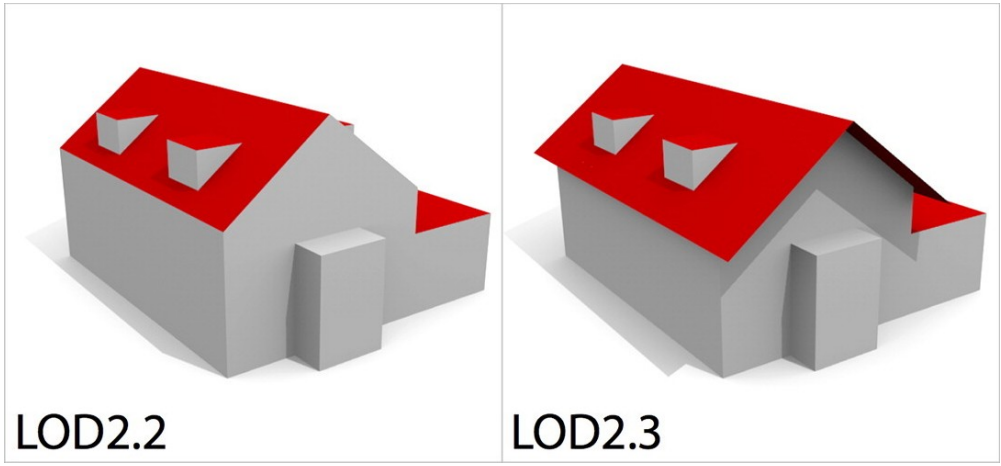


Figure 1: Part of the visual definition of buildings Level of Detail (LoD) containg LoD 2.2 and 2.3 (Biljecki et al., 2016).

Strengths and weaknesses of BD TOPO

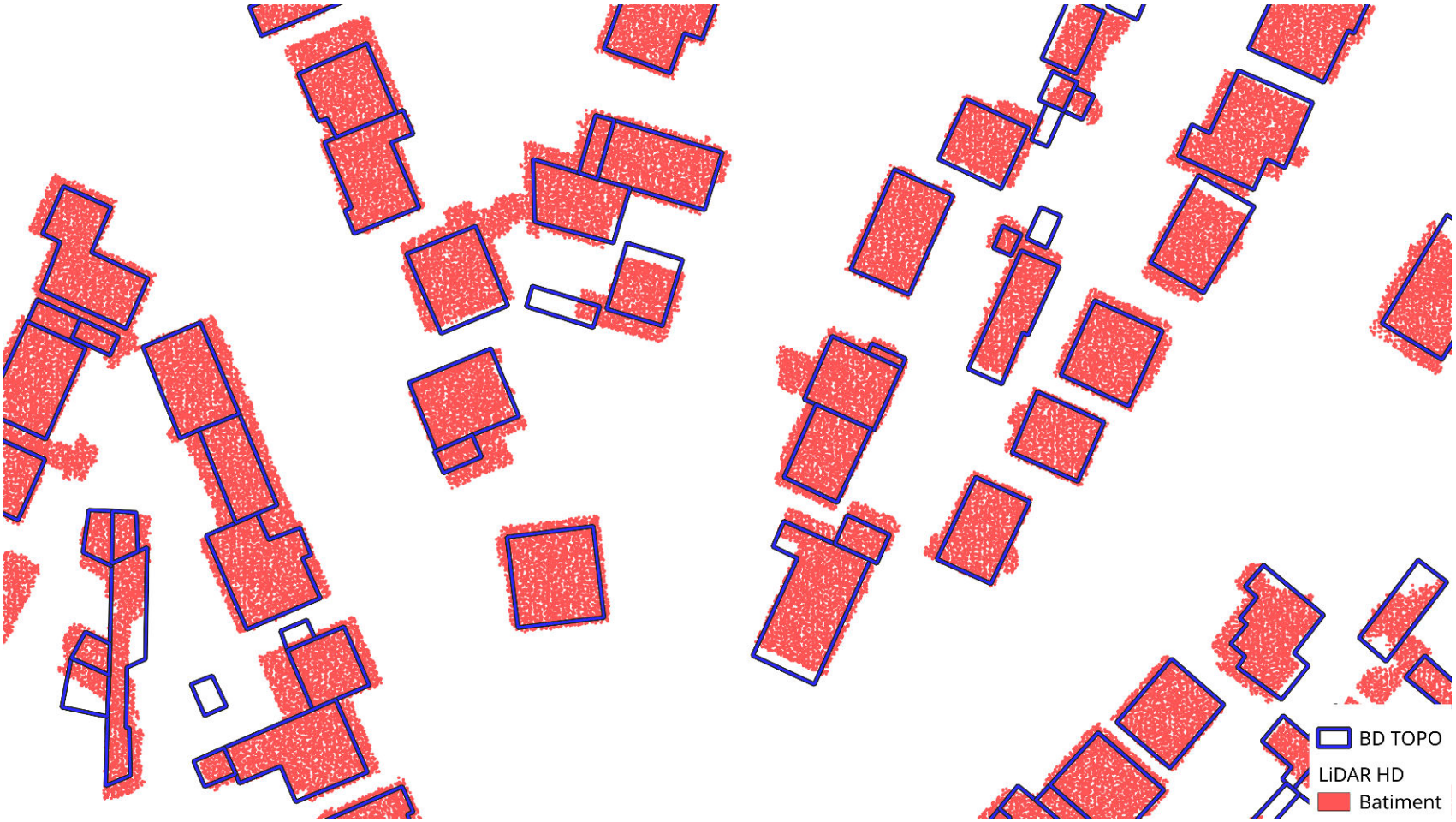


Figure 2: BD TOPO buildings are most of the time footprints, and often shifted.

Objective

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

Overview of the methodology

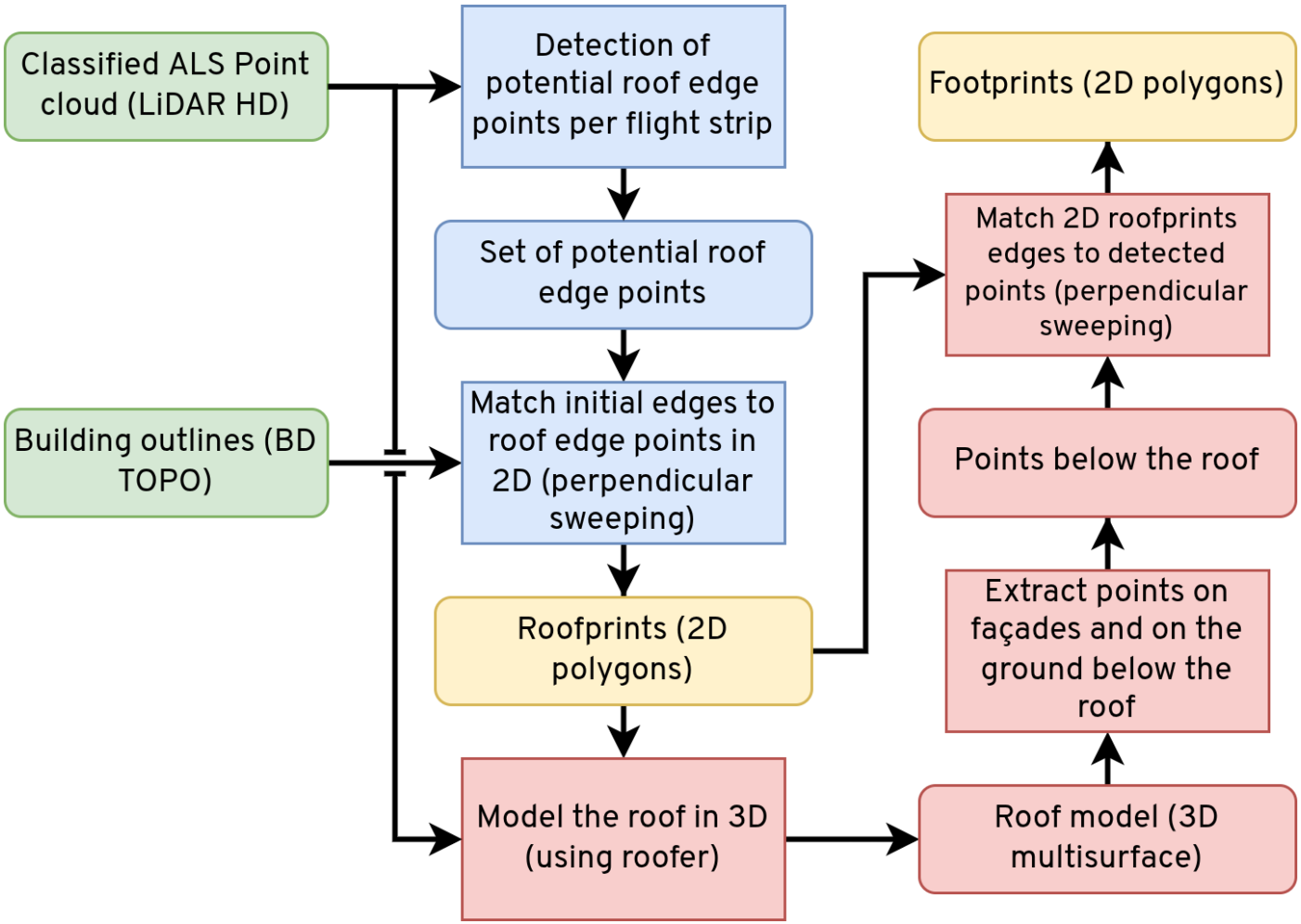


Figure 3: Overview of the pipeline.

Roof edge points

Context 1
Roof edge points 4
Creation of the roofprints 8
Footprint points 24
Creation of the footprints 25

Point cloud topology

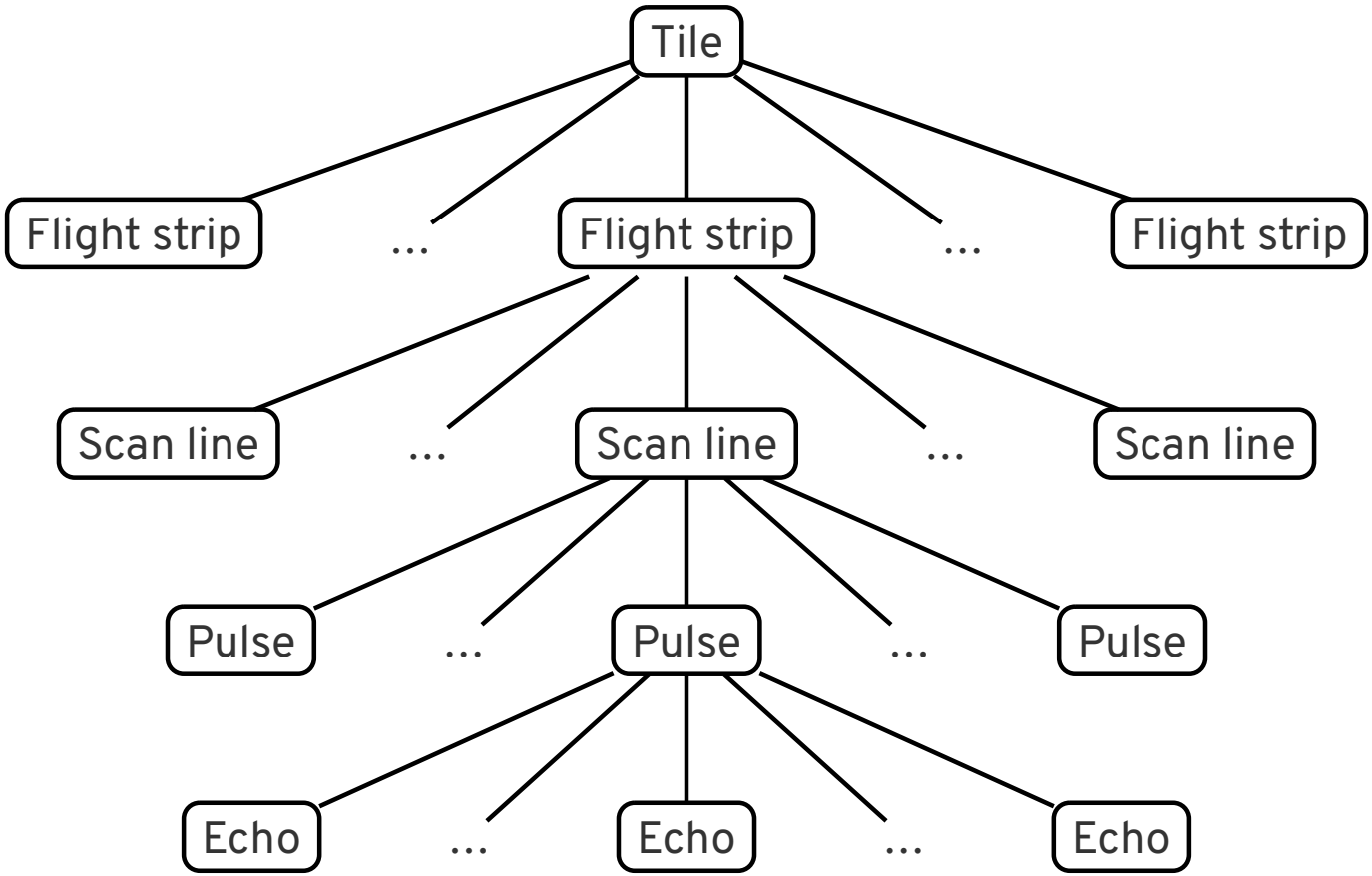
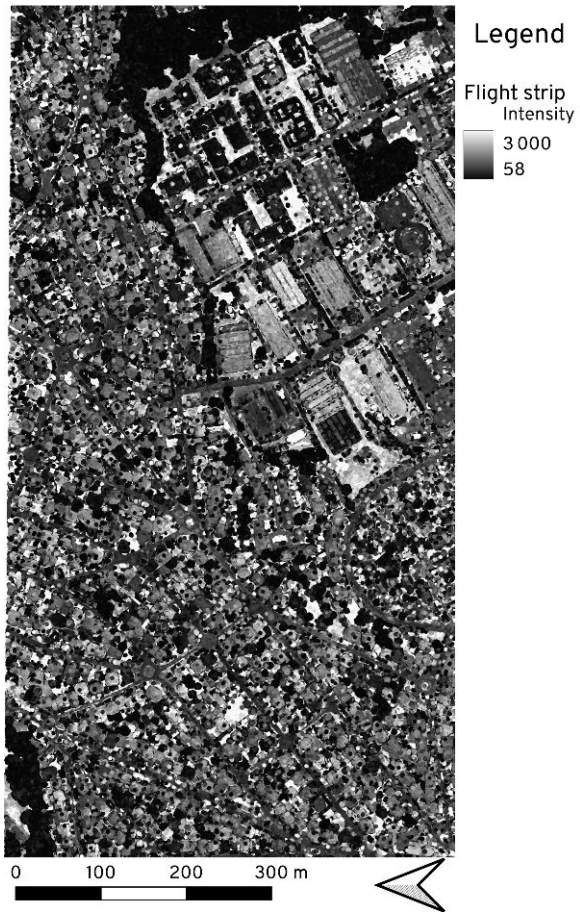
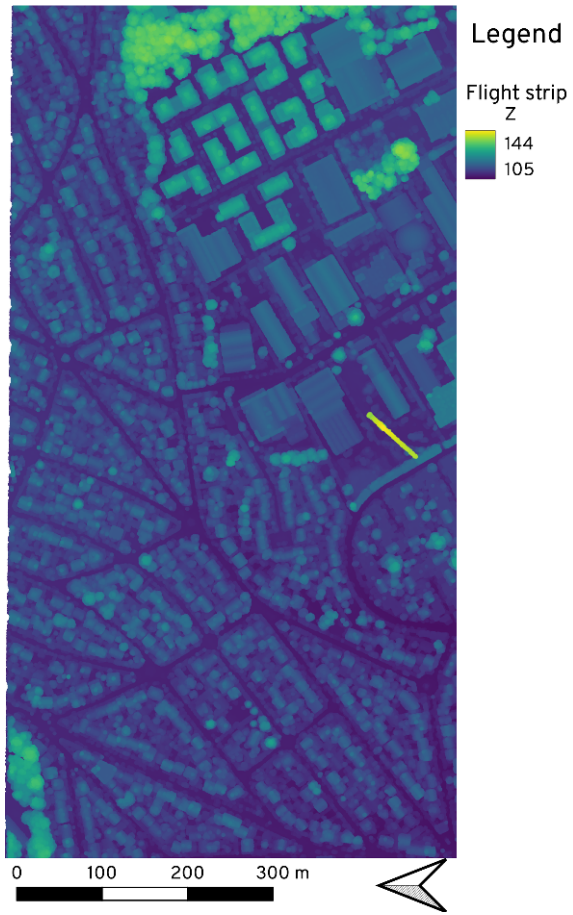


Figure 4: Illustration of the structure of an ALS point cloud.

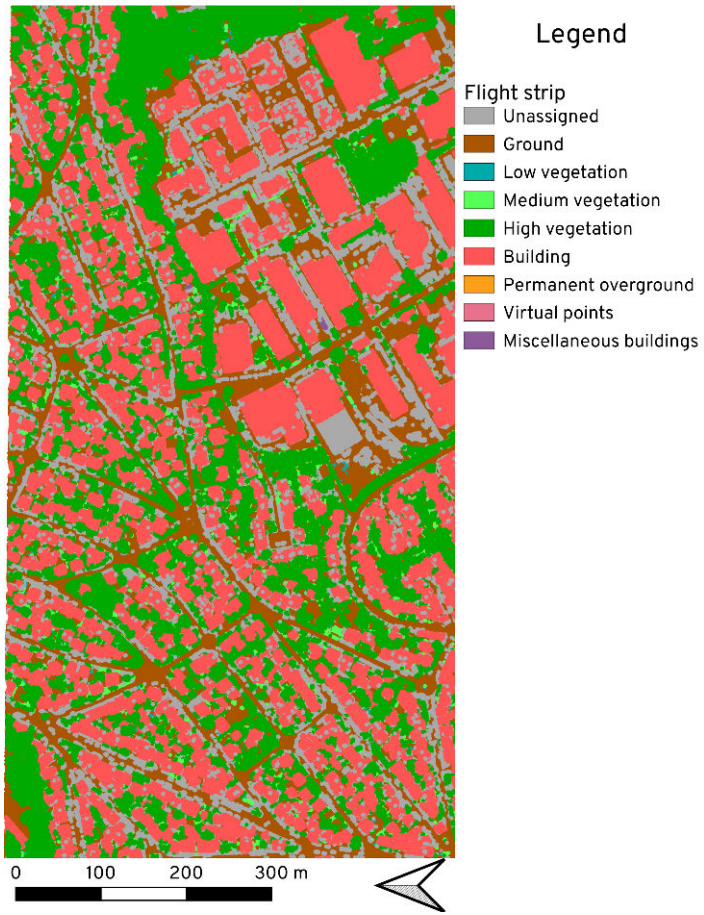
Point cloud topology



(a) Points coloured by intensity.



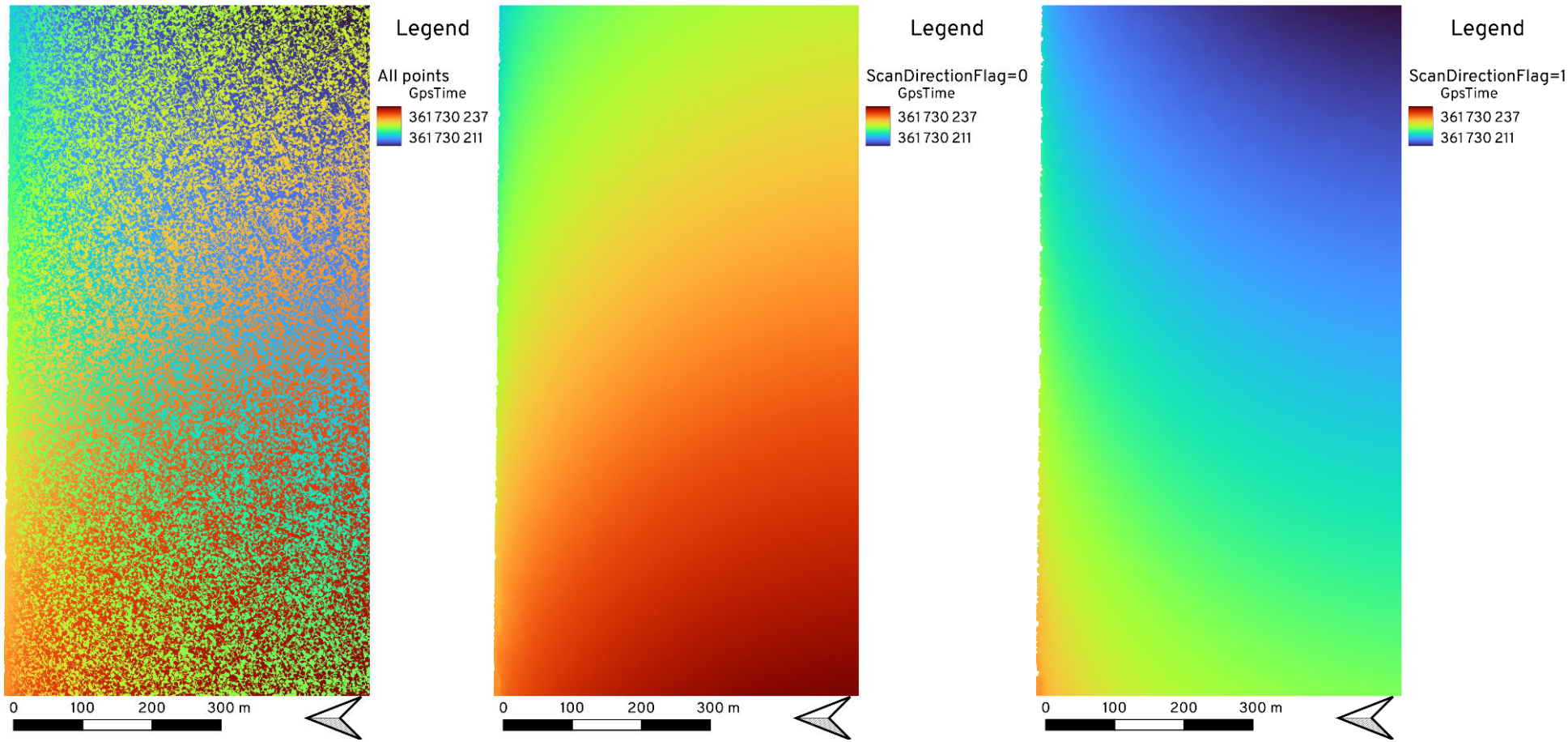
(b) Points coloured by height.



(c) Points coloured by classification.

Figure 5: Different visualizations of one flight strip.

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.

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.

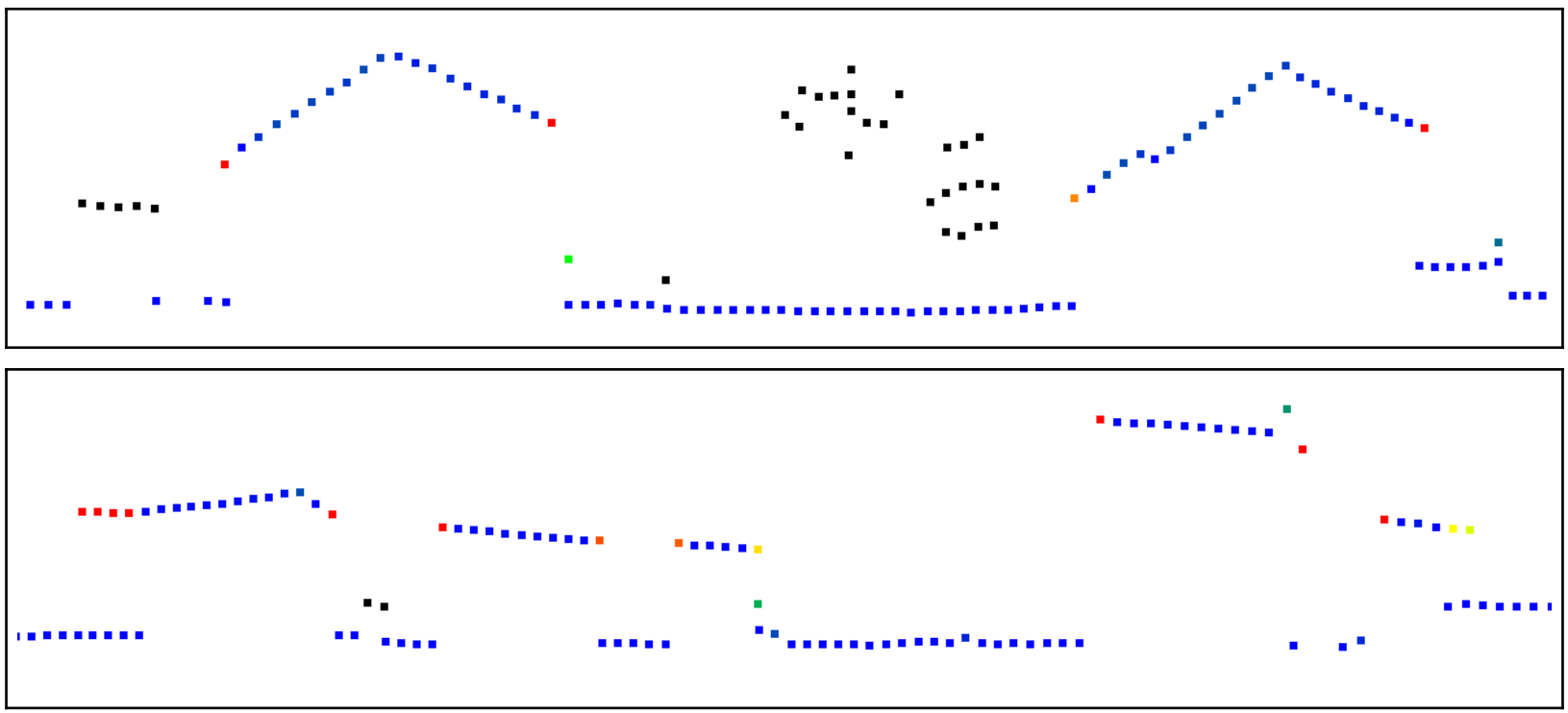


Figure 7: Two examples of the height differences obtained on a scan line. Black points are vegetation points, and for the other points the color represents the value of Δh, with blue-green-yellow-red from low to high values.

Creation of the roofprints

Context 1
Roof edge points 4
Creation of the roofprints 8
Footprint points 24
Creation of the footprints 25

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.**

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.

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.

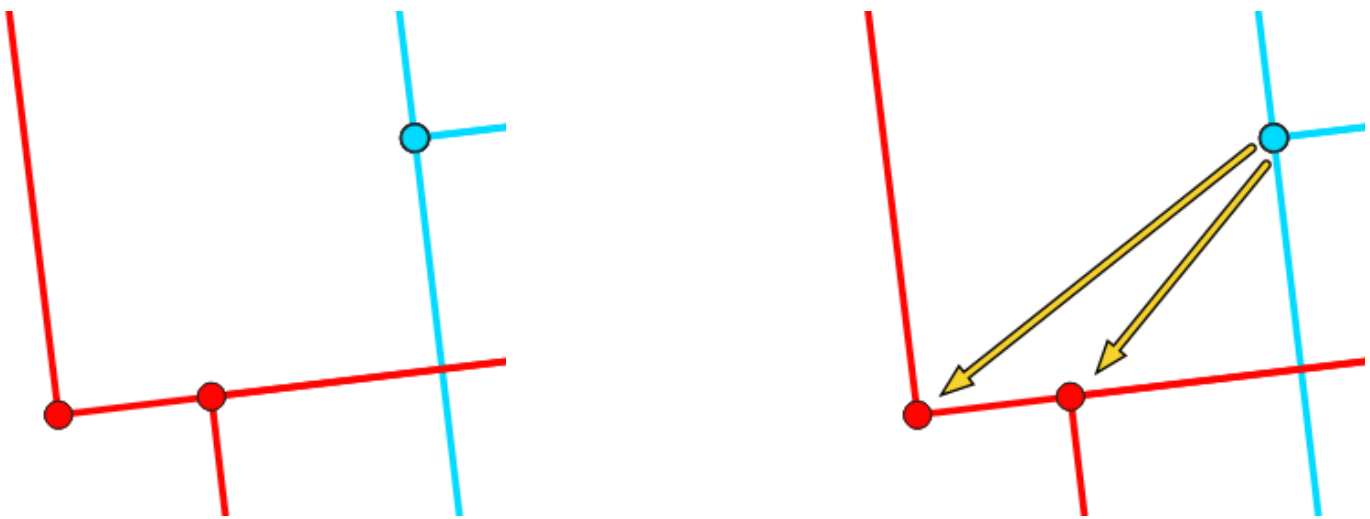


Figure 8: Two vertices at the exact same position become two distinct vertices at the boundary between two buildings.

General idea

We focus on **edges one by one**, except for overlapping edges which are treated together. Edges are treated as **directed lines**, which can be translated in any direction, but not rotated.

General idea

We focus on **edges one by one**, except for overlapping edges which are treated together. Edges are treated as **directed lines**, which can be translated in any direction, but not rotated.

Therefore, to satisfy the constraints described previously, there is only one property $\mathcal{P}(l)$ that we need to preserve for each line l, regarding its previous line l^- and next line l^+:

> ***The intersection between l and l^- should occur before the intersection between l and l^+.***

This property is **equivalent to not flipping the edge l.**

General idea

We focus on **edges one by one**, except for overlapping edges which are treated together. Edges are treated as **directed lines**, which can be translated in any direction, but not rotated.

Therefore, to satisfy the constraints described previously, there is only one property $\mathcal{P}(l)$ that we need to preserve for each line l, regarding its previous line l^- and next line l^+:

The intersection between l and l^- should occur before the intersection between l and l^+.

This property is **equivalent to not flipping the edge l**.

Moreover, we know that if we shift l, l^- and l^+ by the **same extent in the same direction**, this property will still hold for l.

Our simple approach

Simple algorithm to find a correct configuration given a line l_0 to move by an extent δ:

1. Compute the shift direction as the normal $\vec{n}$ of l_0.
2. Move l_0 by $\delta\vec{n}$.
3. If $\mathcal{P}(l_0)$ is not satisfied, move l_0^- and l_0^+ by $\delta\vec{n}$ as well. This ensures that $\mathcal{P}(l_0)$ is satisfied.
4. Set $l_p = l_0^-$. While either of $\mathcal{P}(l_p)$, $\mathcal{P}(l_p^-)$ or $\mathcal{P}(l_p^+)$ is not satisfied, move l_p by $\delta\vec{n}$ (if not already moved) and set $l_p = l_p^-$.
5. Set $l_n = l_0^+$. While either of $\mathcal{P}(l_n)$, $\mathcal{P}(l_n^-)$ or $\mathcal{P}(l_n^+)$ is not satisfied, move l_n by $\delta\vec{n}$ (if not already moved) and set $l_n = l_n^+$.

Our simple approach

Simple algorithm to find a correct configuration given a line l_0 to move by an extent δ:

1. Compute the shift direction as the normal $\vec{n}$ of l_0.
2. Move l_0 by $\delta\vec{n}$.
3. If $\mathcal{P}(l_0)$ is not satisfied, move l_0^- and l_0^+ by $\delta\vec{n}$ as well. This ensures that $\mathcal{P}(l_0)$ is satisfied.
4. Set $l_p = l_0^-$. While either of $\mathcal{P}(l_p)$, $\mathcal{P}(l_p^-)$ or $\mathcal{P}(l_p^+)$ is not satisfied, move l_p by $\delta\vec{n}$ (if not already moved) and set $l_p = l_p^-$.
5. Set $l_n = l_0^+$. While either of $\mathcal{P}(l_n)$, $\mathcal{P}(l_n^-)$ or $\mathcal{P}(l_n^+)$ is not satisfied, move l_n by $\delta\vec{n}$ (if not already moved) and set $l_n = l_n^+$.

The only guarantees of this algorithm is that it will **terminate** and that the resulting configuration will **satisfy the constraints**.

However, it will **not find an optimal solution**, and in particular, it is **not continuous** with respect to δ.

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_05_18-monthly.html**

Total energy to minimize

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

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

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.

Total energy to minimize

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 supporting line 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.

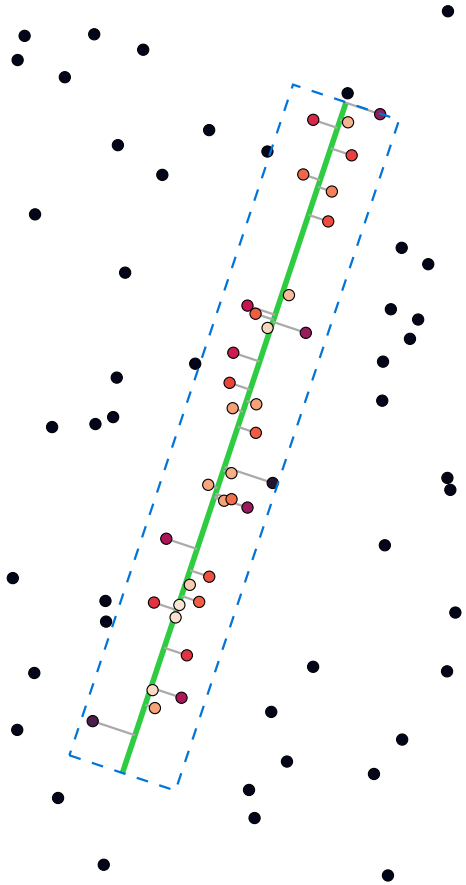


Figure 9: Illustration of the proximity score for an isolated edge.

Total energy to minimize

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 supporting line 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:

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

We use $\varepsilon = 30$ centimetres.

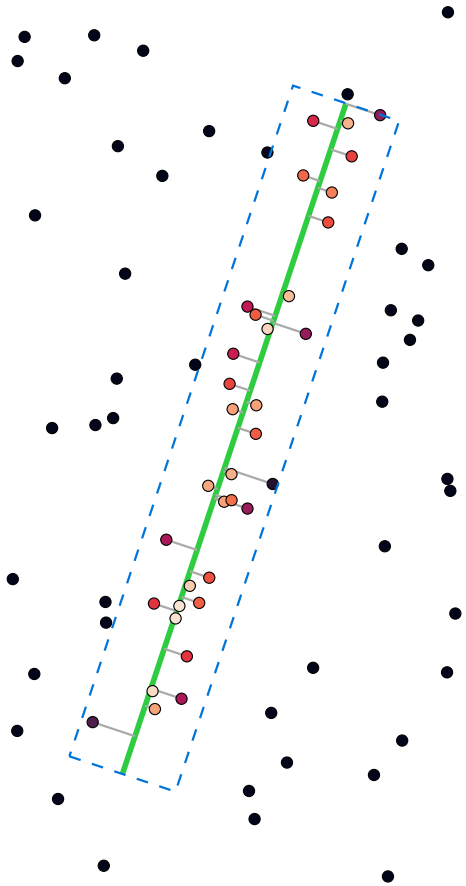


Figure 11: Illustration of the proximity score for an isolated edge.

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.

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.

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 $u_{i \to j}$.

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 $u_{i \to j}$.
3. Compute the inward vector v_i as the average of the vectors $u_{i \to j}$:

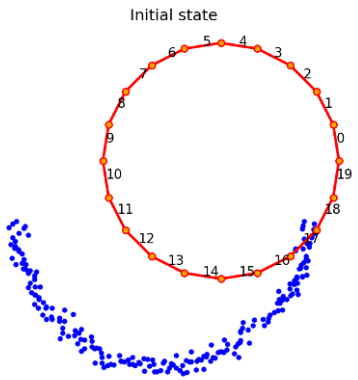
$$v_i = \frac{1}{|\mathcal{B}(p_i, \delta)|} \sum_{p_j \in \mathcal{B}(p_i, \delta)} \frac{p_j - p_i}{|p_j - p_i|}$$

We use $\delta = 2$ metres.

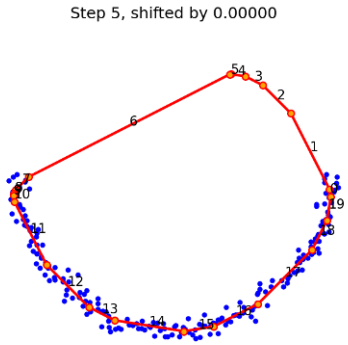
Illustrations of matching the edges to the points

Illustrations can be viewed by following this link: **https://alexandre-bry.github.io/MSc_Thesis-Report/slides_pages/2026_05_18-monthly.html**

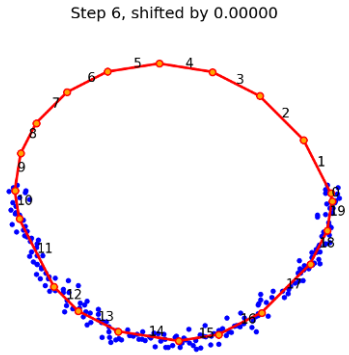
Illustrations of matching the edges to the points



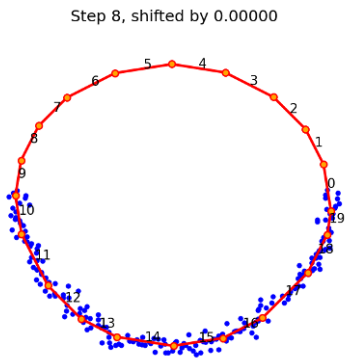
(a) Initial state



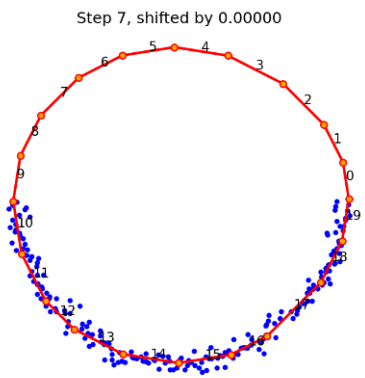
(b) $\alpha = 0.00$ (no regularization)



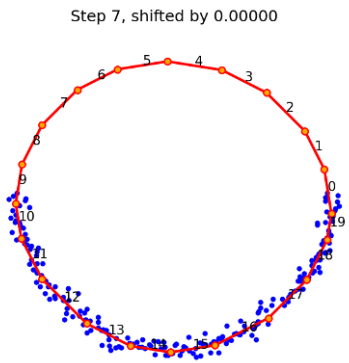
(c) $\alpha = 0.05$



(d) $\alpha = 0.20$



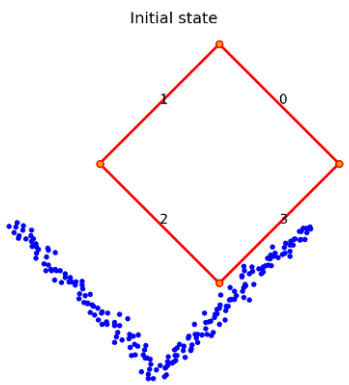
(e) $\alpha = 0.50$



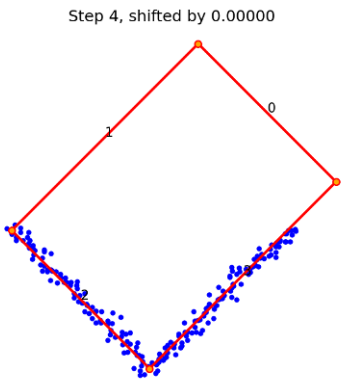
(f) $\alpha = 1.00$

Figure 12: Matching a circle to a half-circle of points with different values of the regularization parameter α.

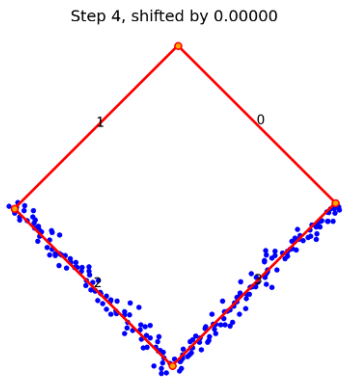
Illustrations of matching the edges to the points



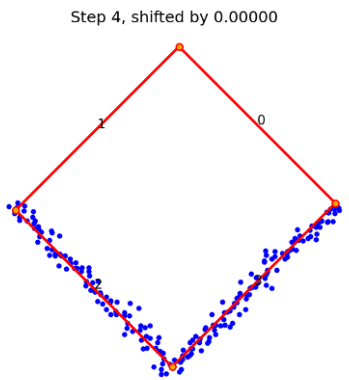
(a) Initial state



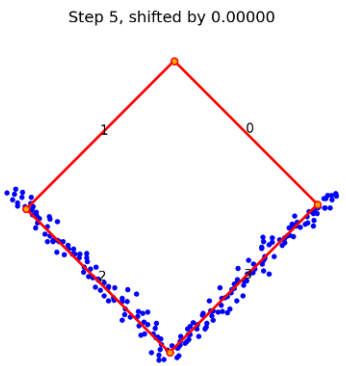
(b) $\alpha = 0.00$ (no regularization)



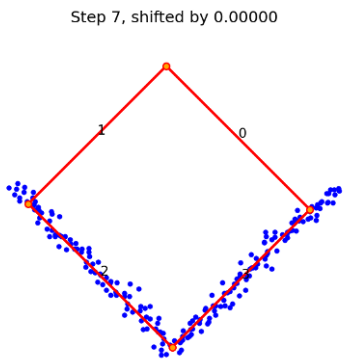
(c) $\alpha = 0.05$



(d) $\alpha = 0.20$



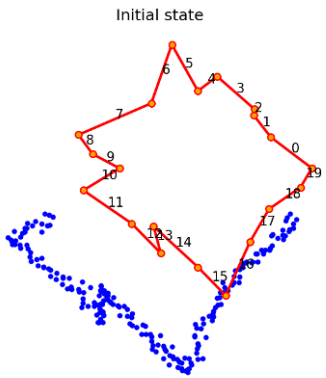
(e) $\alpha = 0.50$



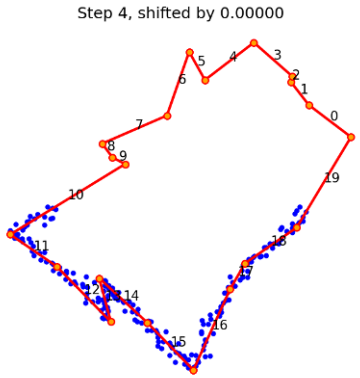
(f) $\alpha = 1.00$

Figure 13: Matching a square to points along two sides of a larger square with different values of the regularization parameter α.

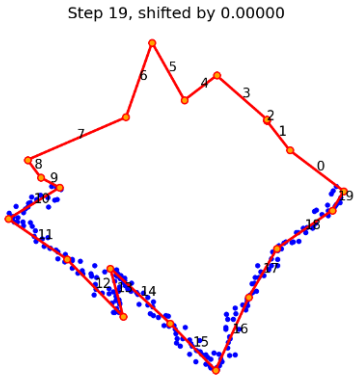
Illustrations of matching the edges to the points



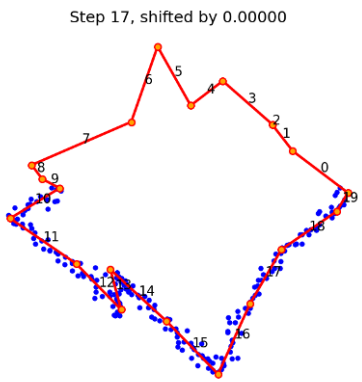
(a) Initial state



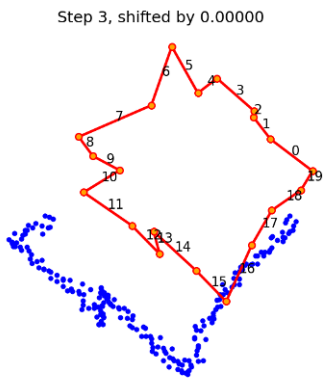
(b) $\alpha = 0.00$ (no regularization)



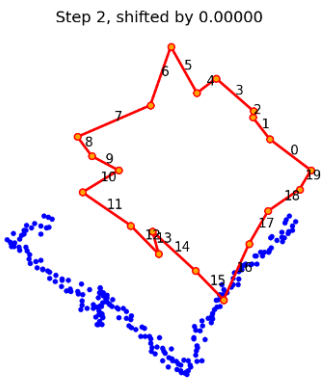
(c) $\alpha = 0.05$



(d) $\alpha = 0.20$



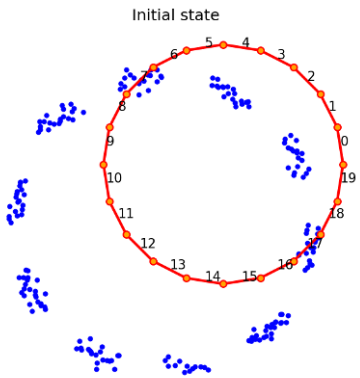
(e) $\alpha = 0.50$



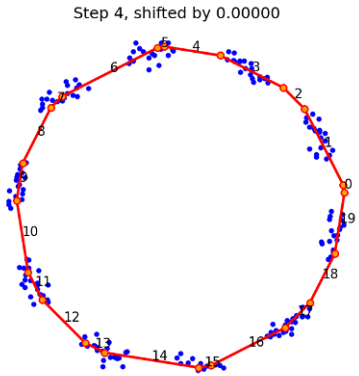
(f) $\alpha = 1.00$

Figure 14: Matching a polygon to half of its shape with points with different values of the regularization parameter α.

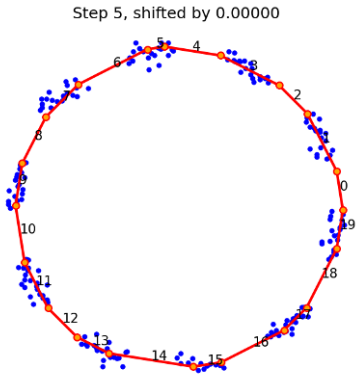
Illustrations of matching the edges to the points



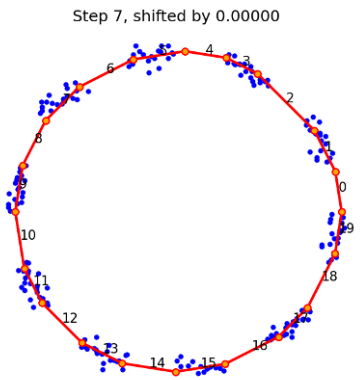
(a) Initial state



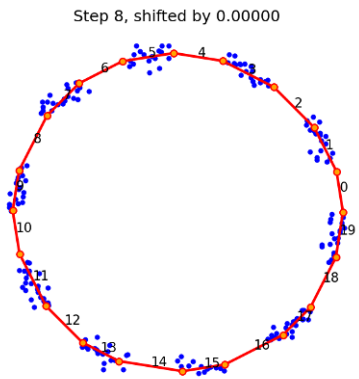
(b) $\alpha = 0.00$ (no regularization)



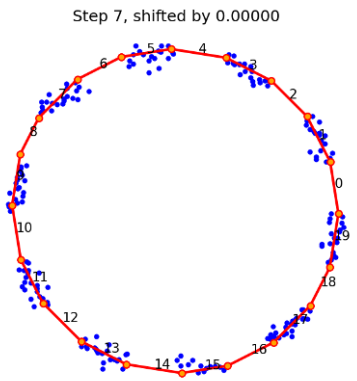
(c) $\alpha = 0.05$



(d) $\alpha = 0.20$



(e) $\alpha = 0.50$



(f) $\alpha = 1.00$

Figure 15: Matching a circle to points along half of its boundary with different values of the regularization parameter α.

Illustration with a real building

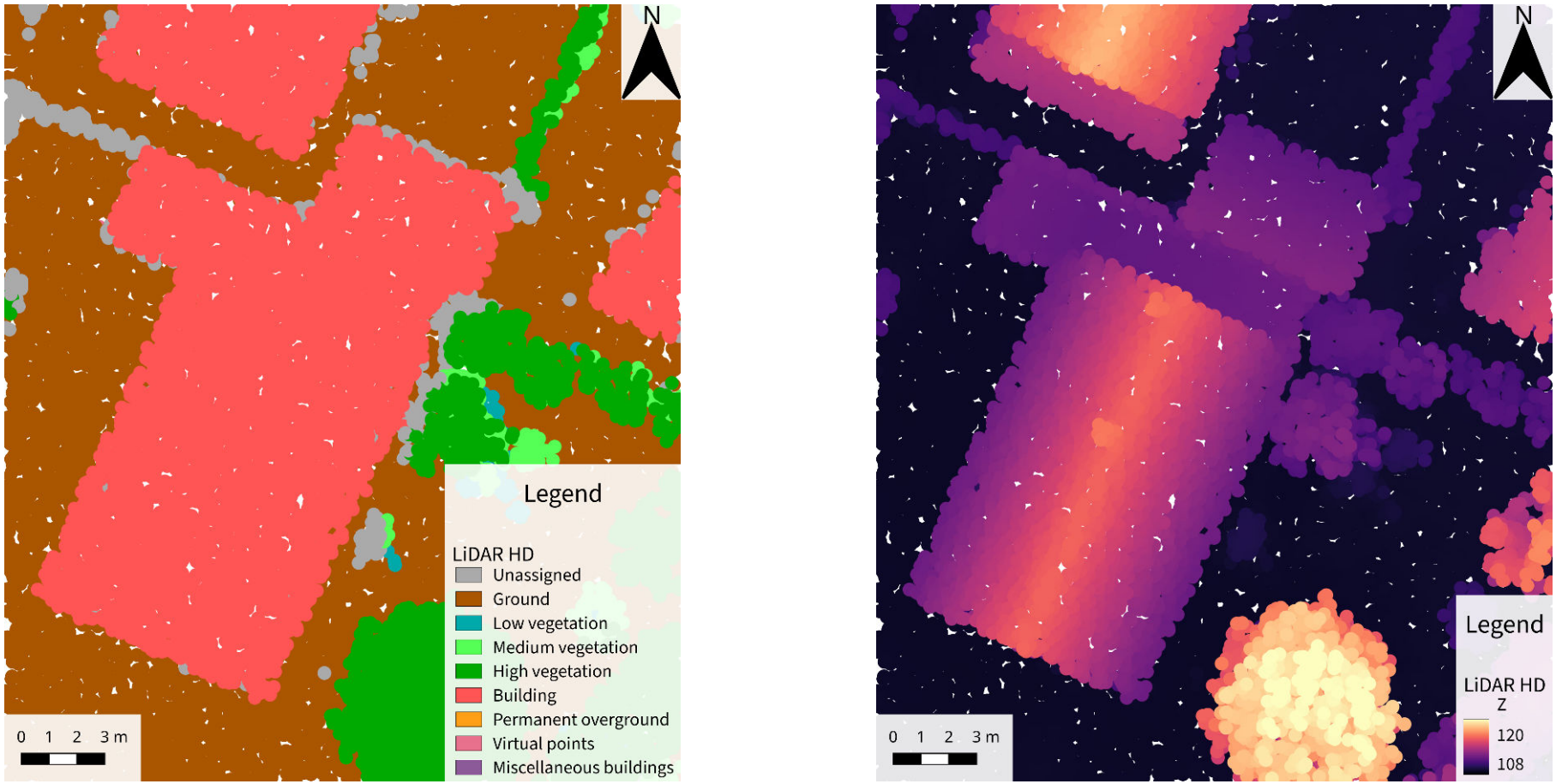


Figure 16: The LiDAR HD data.

Illustration with a real building

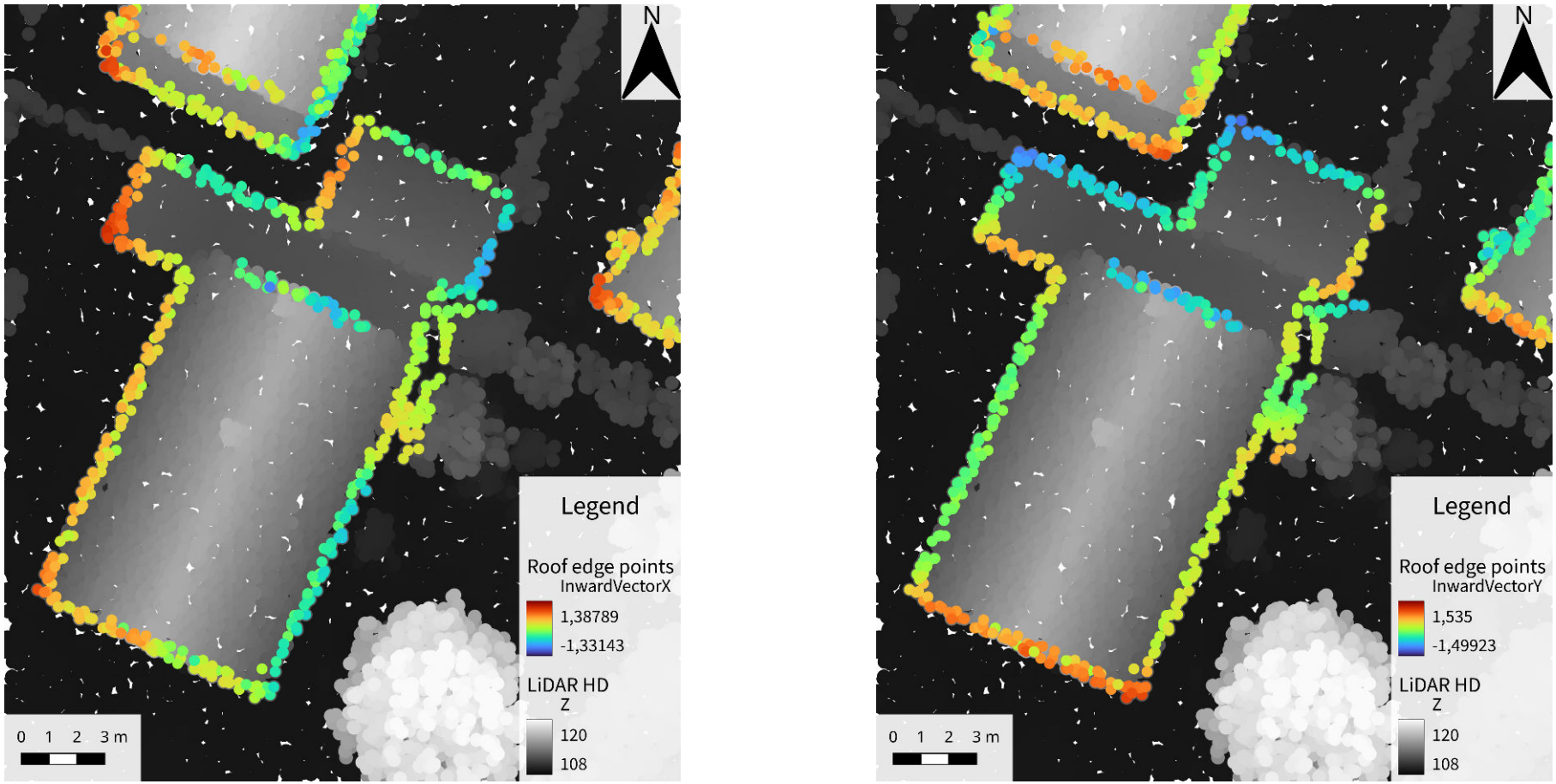


Figure 17: The detected roof edge points coloured by their inward vector.

Illustration with a real building

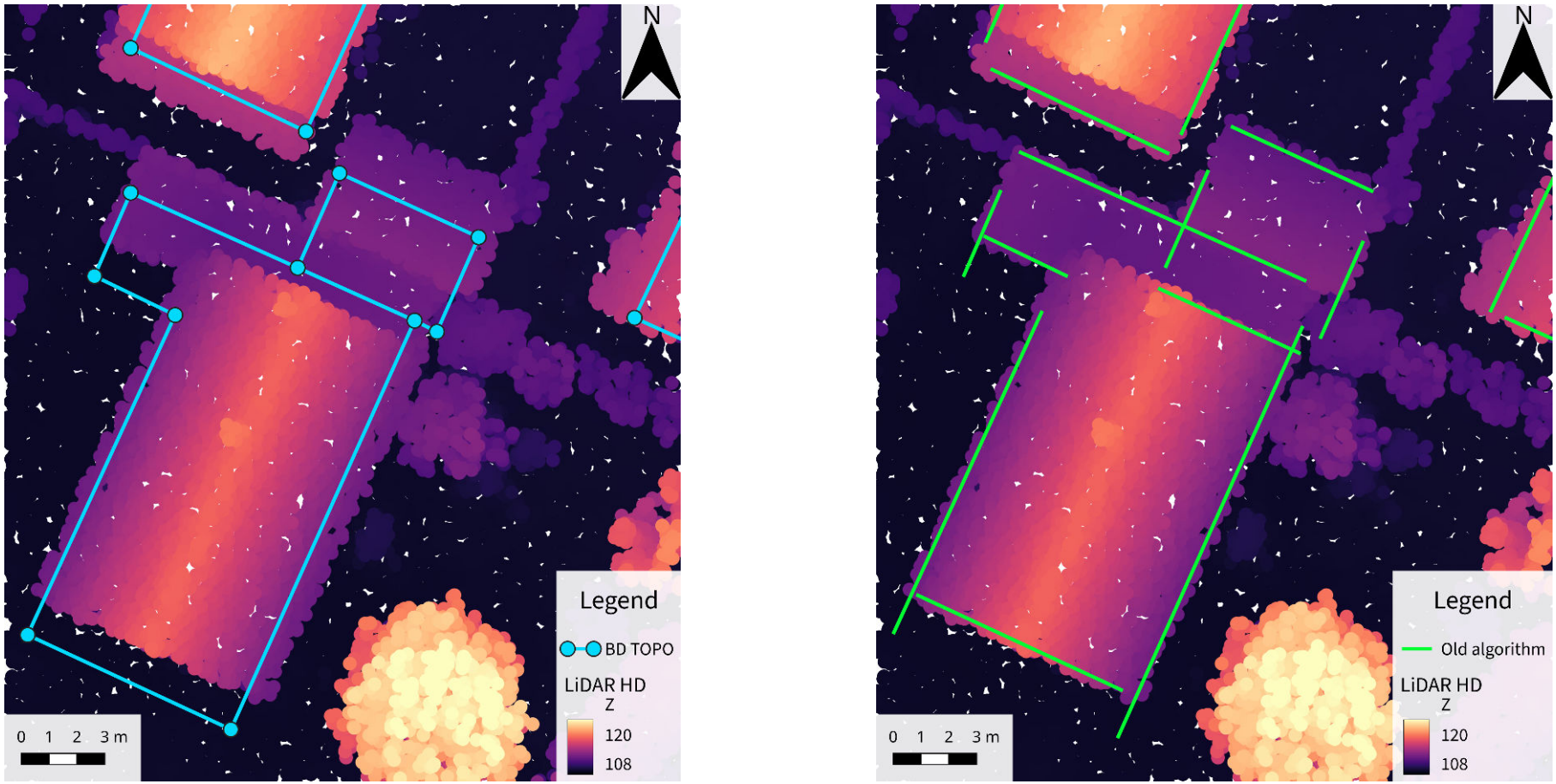


Figure 18: Comparison of BD TOPO outlines and roofprints computed with the old algorithm.

Illustration with a real building

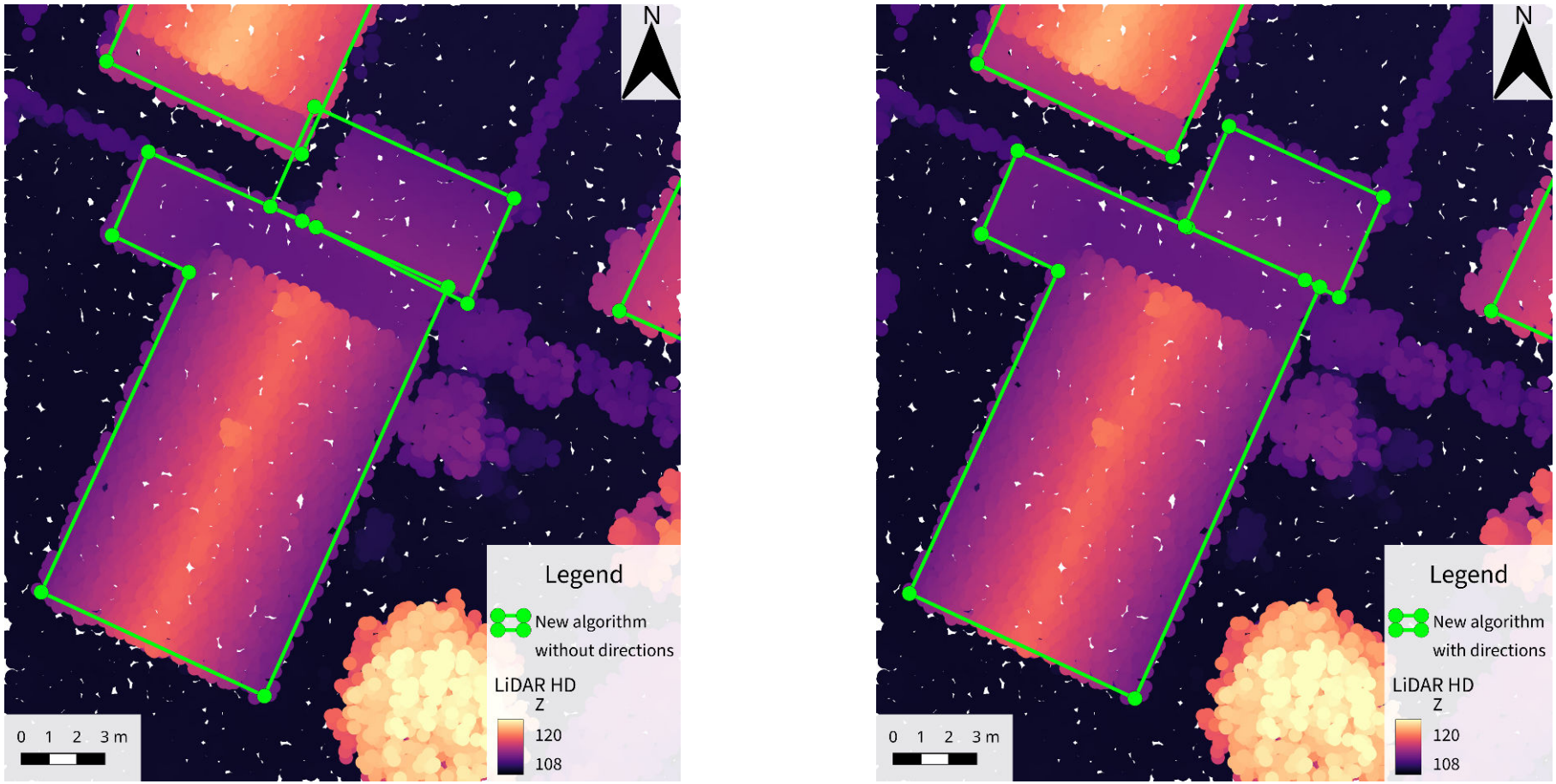


Figure 19: Comparison of the roofprints computed with the new algorithm without and with inward vectors.

Footprint points

Context 1
Roof edge points 4
Creation of the roofprints 8
Footprint points 24
Creation of the footprints 25

Footprint points

1. Build the roof in 3D using a 3D roof constructor (such as **roofer**) with the roofprints as input
2. Select all the points **under** the 3D roof with a small horizontal buffer (for the scoring function) and a small vertical buffer (for roof points slightly below the roof)

Footprint points

1. Build the roof in 3D using a 3D roof constructor (such as **roofer**) with the roofprints as input
2. Select all the points **under** the 3D roof with a small horizontal buffer (for the scoring function) and a small vertical buffer (for roof points slightly below the roof)

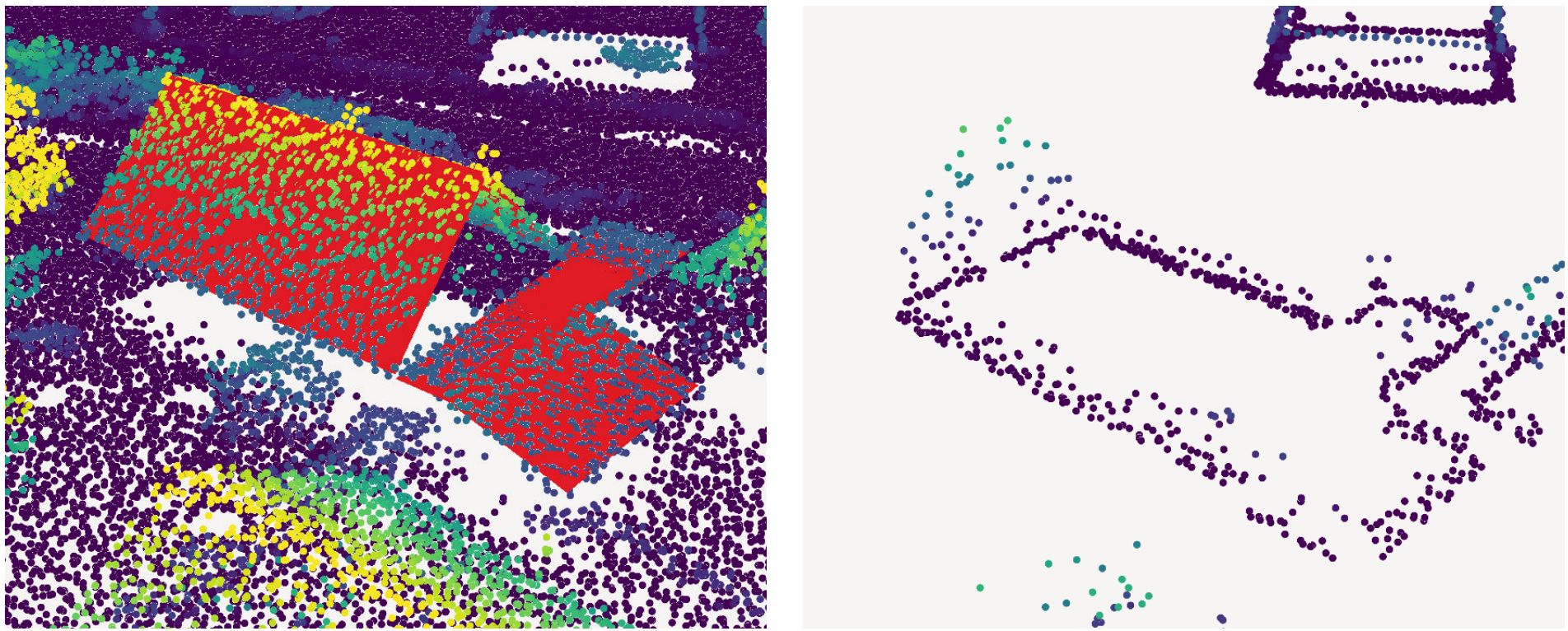


Figure 24: Illustration of the 3D roof structure and the selected points for the footprint computation.

Footprint points

1. Build the roof in 3D using a 3D roof constructor (such as **roofer**) with the roofprints as input
2. Select all the points **under** the 3D roof with a small horizontal buffer (for the scoring function) and a small vertical buffer (for roof points slightly below the roof)

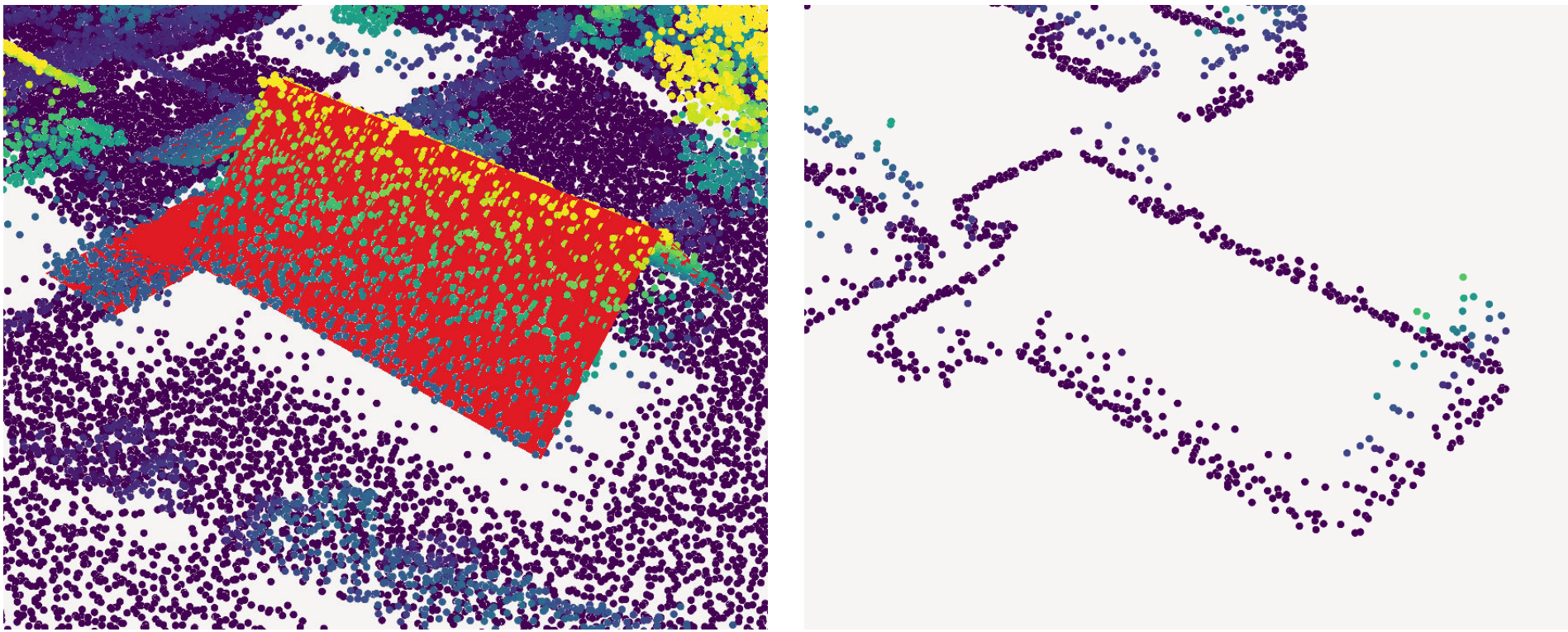


Figure 24: Illustration of the 3D roof structure and the selected points for the footprint computation.

Creation of the footprints

Context 1
Roof edge points 4
Creation of the roofprints 8
Footprint points 24
Creation of the footprints 25

Creation of the footprints

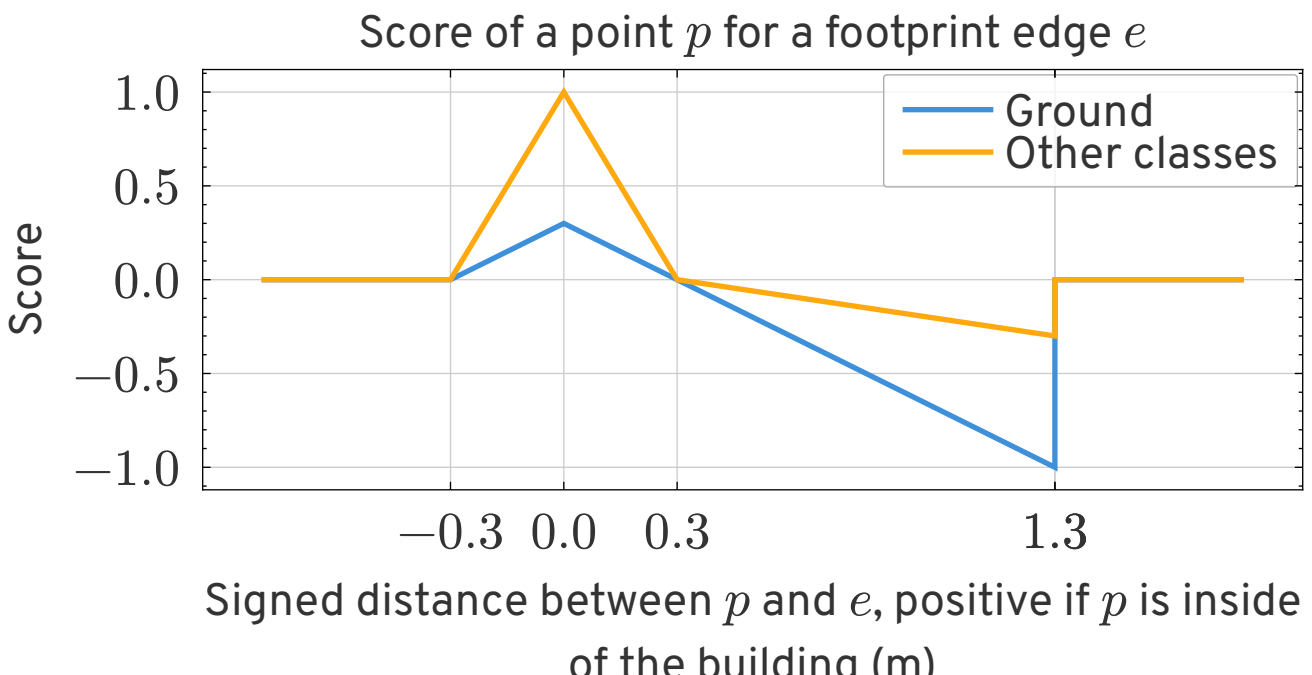
- **Sweep each roof edge horizontally towards the inside** up to 1.5 metres and select the best scoring position

Creation of the footprints

- **Sweep each roof edge horizontally 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

- **Sweep each roof edge horizontally 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

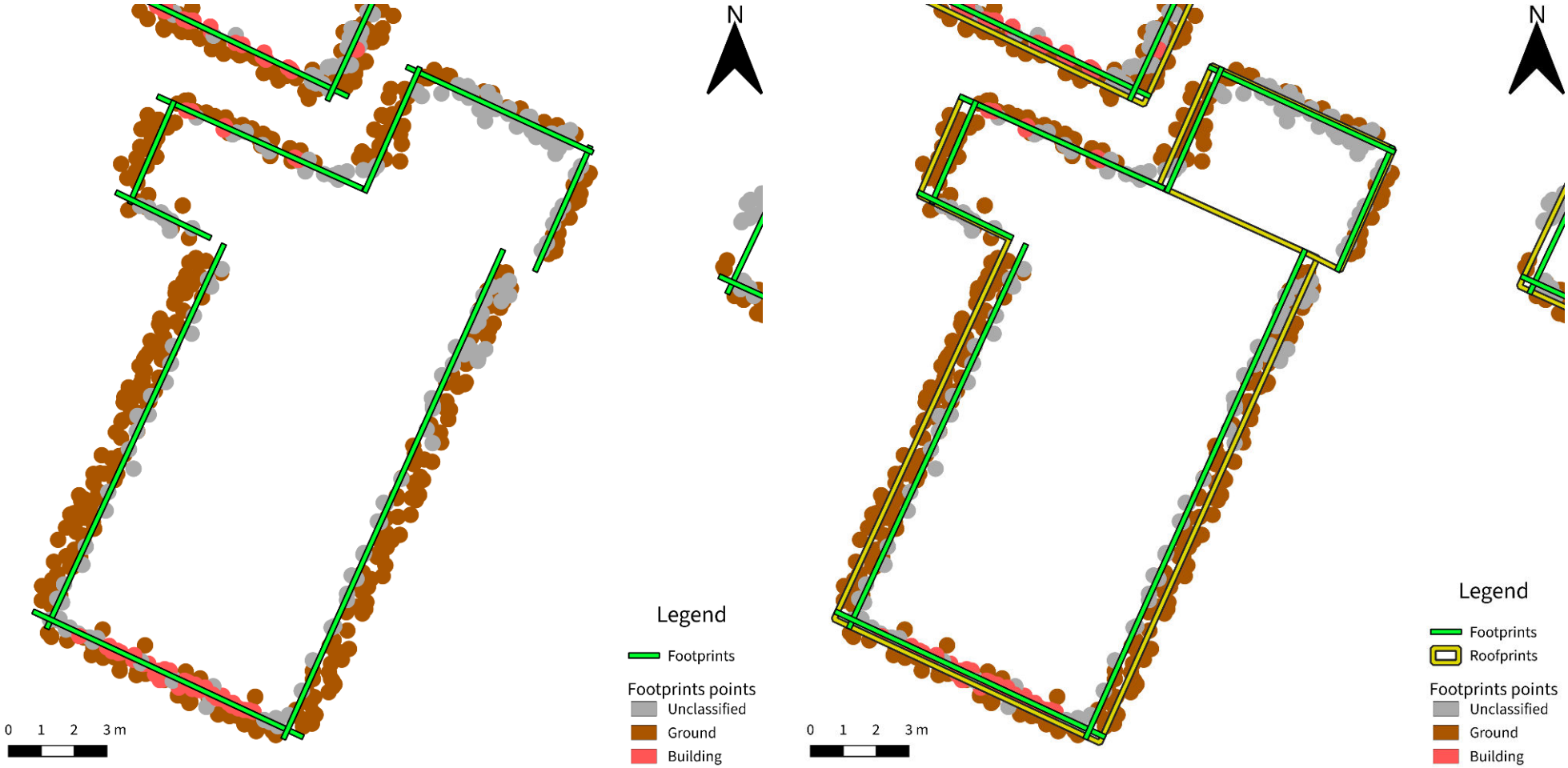


Figure 23: Illustration of the computed footprints.

Creation of the footprints

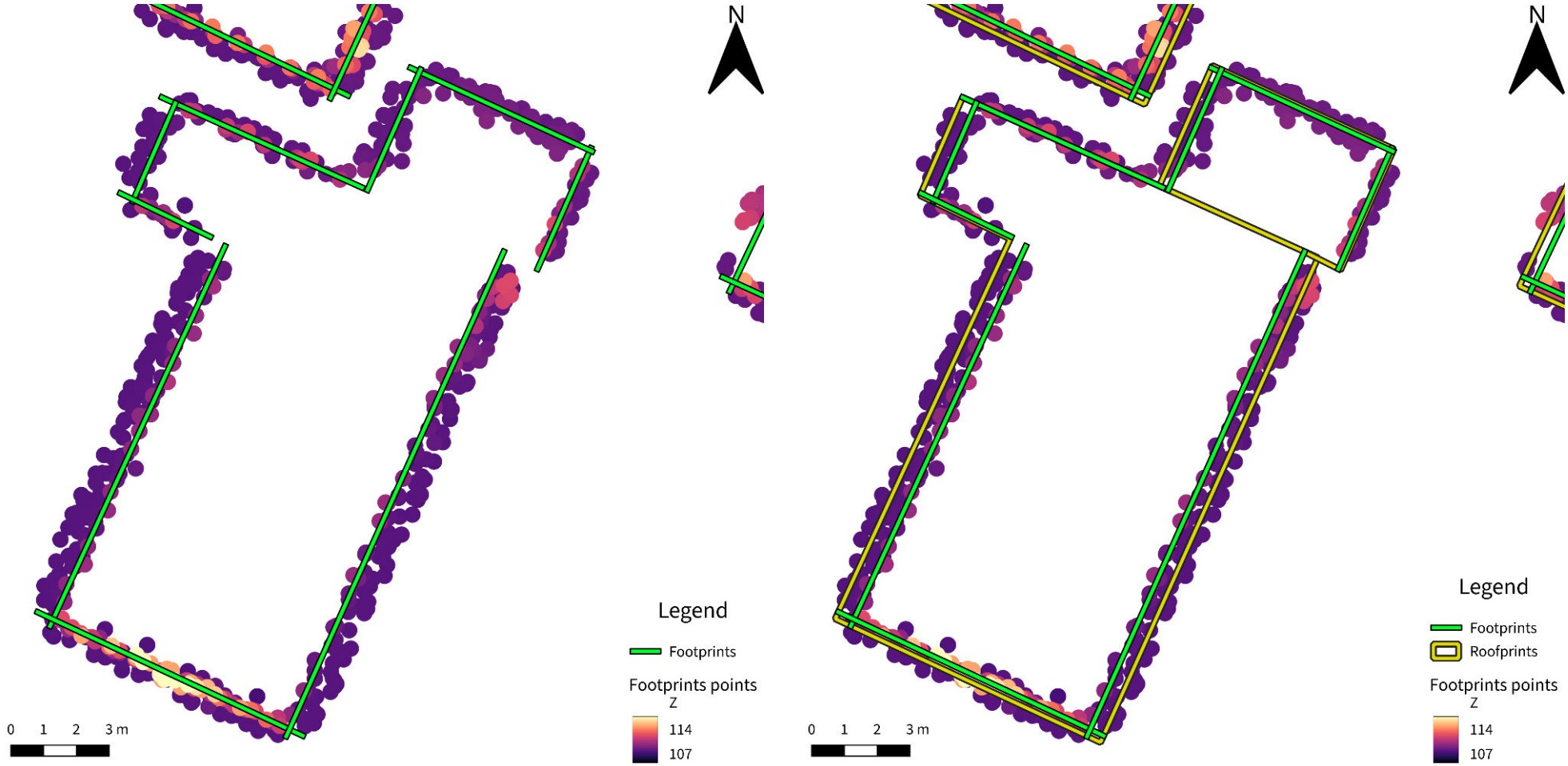


Figure 24: Illustration of the computed footprints.

Thank you for your attention!

Link to the slides:

https://alexandre-bry.github.io/MSc_Thesis-Report/slides_pages/2026_05_18-monthly.html

alexandre.bry@ign.fr, abry@tudelft.nl

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