



HAL
open science

Parametric Surface Fitting on Airborne Lidar Point Clouds for Building Reconstruction

Guillaume Coiffier, Justine Basselin, Nicolas Ray, Dmitry Sokolov

► **To cite this version:**

Guillaume Coiffier, Justine Basselin, Nicolas Ray, Dmitry Sokolov. Parametric Surface Fitting on Airborne Lidar Point Clouds for Building Reconstruction. *Computer-Aided Design*, 2021, 140, pp.103090. 10.1016/j.cad.2021.103090 . hal-03281816

HAL Id: hal-03281816

<https://hal.science/hal-03281816>

Submitted on 8 Jul 2021

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Parametric surface fitting on airborne Lidar point clouds for building reconstruction

Guillaume Coiffier^{a,*}, Justine Basselin^{a,*}, Nicolas Ray^a, Dmitry Sokolov^a

^a *Université de Lorraine, CNRS, Inria, LORIA, F-54000, Nancy, France*

Abstract

Surface reconstruction is an essential step in most processing pipelines involving point clouds. By constructing a surfacic or volumetric model of the cloud, it is possible to infer large-scale semantic and geometric information required for most applications in computer graphics, simulation and virtual reality. Among the different types of point cloud and the variety of possible problems, we are interested in airborne Lidar data and the problem of building reconstruction for urban planning ranging from flood and light exposure simulation to virtual touristic visits.

While most existing reconstruction methods are based on characteristic features extraction in point clouds such as planes, ridges, contours and their combination into a more complex model, we instead adopt a template-based approach relying on a library of complex primitives developed in an industrial context.

This is formulated as a global fitting problem between a constrained triangulated mesh (our template) and the point cloud. More precisely, we design an energy function that takes into account the distance between both objects while integrating outliers rejection directly in our numerical optimization through the use of an M-estimator. This energy function being smooth everywhere, it can be efficiently minimized by quasi-Newtonian methods like the L-BFGS algorithm.

We demonstrate the reliability of our approach on a collection of diverse roof models and several publicly available Lidar datasets, as well as its robustness and limits in function of initialization, point cloud quality and presence of outliers. By only fitting onto relevant points, this method allows a precise fitting as well as a correct outlier segmentation in a unique step, providing a reasonable initialization close to the barycenter of the cloud.

Keywords: Surface fitting, Lidar point clouds, Roof & building reconstruction, Energy minimization, Voronoi diagrams

1. Introduction

Thanks to recent advances in acquisition technologies, point cloud data have become a precise and widespread digital representation of real-world objects, providing the opportunity to perform a large range of numerical treatments on such objects. While they allow an easy visualization by a human-being, these raw acquisitions, however, lack the geometrical semantic required by most applications in computer graphics. A pre-processing step, where the point cloud is segmented and reconstructed from geometrical primitives (planes, cylinders, spheres, *etc.*) or more complex templates, is needed in order to obtain a more suited, higher level representation.

In this work, we are interested in this reconstruction phase in the context of LIDAR point clouds of city districts. These data are acquired via an airplane and contain every object present in the city: cars, trees, building roofs, roads, ground features, *etc.*

Applications of these data, ranging from city visualization for tourism to flood and wind simulation, require the reconstruction of buildings as geometrical objects. As LIDAR point cloud are acquired from the sky, buildings are most often represented by their roofs and extruded down to

ground level. Hence, determining the polygonal surfaces of roofs enables the reconstruction of the whole city.

Most existing roof reconstruction methods are made of two steps. First, a feature detection pipeline, based on statistic tools (as RANSAC or regression), is applied on the point cloud: for instance, planes and ridges are computed from the point cloud to produce features of higher levels of abstractions, such as contour edges and graphs of adjacent planes. Then, features are combined to generate a complete roof template under specific constraints.

We propose an alternative that considers the roof reconstruction as a unique optimization problem. The advantage of this approach is that the determination of each feature directly takes into account that the object we are looking for is an instance of a given roof template, which we assume to be known. In our case, the roof template is a polygonal surface subject to some constraints like the planarity of roof panes, alignment of gutters, or horizontality of the roof's ridge (Figure 2). Our idea consists in minimizing a distance function that we have defined between the roof and the point cloud: we want the surface to be as close as possible to every point of the point cloud and vice versa. Such an energy function is presented in this article as well as its optimization, exploiting an efficient evaluation of its value and gradient.

This article is organized as follows: after a brief overview of related work (§2), we formulate the template fitting as an

*Corresponding author

Email addresses: guillaume.coiffier@inria.fr (Guillaume Coiffier), justine.bassel@inria.fr (Justine Basselin)

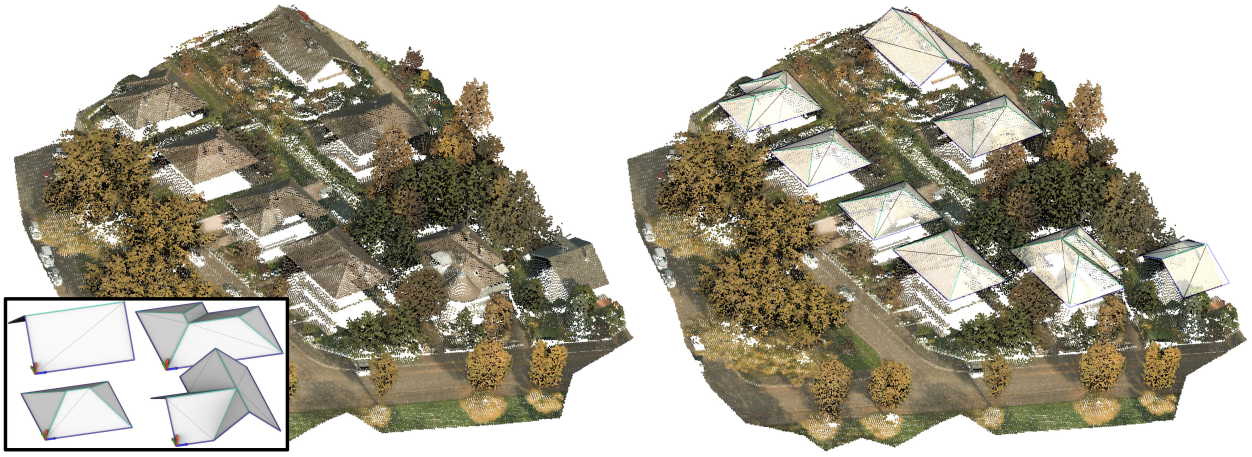


Figure 1: **Left:** example of roof templates (bottom frame) and a typical airborne LIDAR point cloud from the Strasbourg dataset [1]. **Right:** Our methods fit the parametric roof templates onto corresponding locations in the point cloud, effectively detecting them. Each template was selected and roughly initialized by a human operator near a roof of the point cloud and finally fitted on the whole cloud. Points belonging to non-roof objects like ground areas or trees are considered as outliers by our method.

optimization problem (§3) and then, develop our resolution method (§4). Finally, the behavior of the method is extensively evaluated (§5) to properly estimate its limitations.

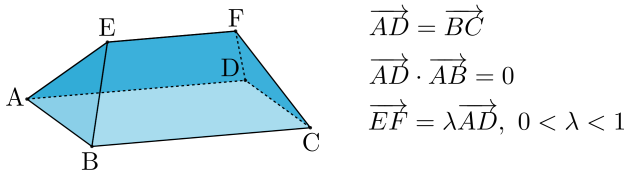


Figure 2: A roof template is a polygonal surface (in this example it has triangular faces ABE and CDF, and quad faces BCFE and DAEF), that can fit different roofs by changing the geometry of its vertices, subject to a set of constraints.

2. Related work

We focus on a *surface reconstruction* problem, where we have strong prior on the roof geometry, making it closer to *surface fitting*. We also need to consider the particularities of *roof reconstruction* problems. An overview of *surface reconstruction* is presented in [2]. For objects with large smooth regions, detecting them provides important information [3].

For surfaces with analytic expression (planes, spheres, cylinders, cones and tori) a robust reconstruction is possible with RANSAC algorithm [4]. RANSAC is not robust to noise in general [5] but works quite well in our case where the primitives are large planar regions. It is then possible to discover global constraints [6], that are introduced in a second step of optimization. The same idea was applied to interactive architecture modeling [7], with vertex, edge and face snapping as global constraints. Our problem is quite different because we exploit constraints that are defined in advance to find the best possible match. In our case, we expect the final roof surface to correspond to a roof template, making it relevant to explore the literature of the *surface fitting* community. To find the position

and deformation of a model, most work [8] rely on the Iterative Closest Point (ICP) optimization algorithm. The basic problem is to minimize the distance between a surface and a point cloud, by finding the optimal rigid transformation of the surface [9, 10]. It is also possible to consider deformations of the surface [11], but it often requires correspondence landmarks to be placed by a user or automatically detected [12]. A non-rigid transformation has to preserve the overall shape of the surface; so the deformation should be locally close to an isometry, i.e. it can be locally considered as a rigid transformation of the original surface [13, 14, 15]. This approach works pretty well for dense meshes (e.g. issued from scans), but minimizing the deformation of the surface is not a valid objective for every application (including roof reconstruction). For articulated models, a rigid per-bone transformation is instead considered [16] and, for CAD models, [17] works with a shape described by a set of parameters (hole radius, edge length, *etc.*). This is very close to our problem, our templates and constraints are very similar, but our method is working well in presence of noise and outliers, and is designed for using efficient solvers. Another way to fit a surface is presented in [18]: it minimizes a symmetric distance between two surfaces that is defined as an integral over all points of the model w.r.t a restricted Voronoi diagram. It gives a distance function that is smooth and can be directly optimized by L-BFGS [19], instead of alternating between projection and minimization as it is done in ICP. Our method is quite similar, but supports roof model constraints as well as rejection of outliers thanks to a robust M-estimator approach. The problem of *roof reconstruction* is also well studied. Most works [20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30] use different pipelines where each step detects and/or combines features [31, 32] extracted from the point cloud. The final mesh is then given by combining faces, edges, ridges, *etc.*. In [33], the final roof geometry is optimized under the constraints discovered when combining the different elements e.g. edge alignment or vertex at the same height. Instead of start-

ing from planes with RANSAC, [34] is similar to ours in the sense that they directly fit a roof model with a robust method, albeit it requires a tight 2D bounding box of the roof as input. Another approach [35, 36] directly matches roof templates from a fair footprint approximation. Their objective function minimizes distances in the Z axis and relies on information entropy to tune the size of the roof. Their optimization is based on Markov chain Monte Carlo, which allows to change the roof model during the optimization. Our solution exploits less application-specific prior (footprint and using only the z coordinate for the distance) and our smooth objective function strongly improves the converge rate near local minima. They also propose a solution to combine basic roofs into more complex structures, that is an interesting way to improve our results in the future.

3. Problem formulation

We consider the problem of fitting a parametric template onto a point cloud. In this article, we call template a user-defined triangulated mesh subject to various geometrical constraints. While this framework is broad and can fit a variety of problems, we will restrain ourselves to the study of building reconstruction in airborne data. Our templates will therefore be roof templates and their associated mesh will be topological disks. Examples of some of those templates can be found on Figure 1 (bottom left frame). The geometrical constraints on these models reflect the planarity of roof planes, alignment of gutters, horizontality of the roof’s ridge, *etc.*, and can be expressed as equalities or inequalities linking the coordinates of the template’s vertices. The data we use are raw LIDAR acquisitions over regions of interest. In addition to the roof we want to fit, the point clouds contain other structures ranging from chimneys and antennas to trees, ground areas or even neighboring roofs. The fitting method is therefore required to be robust whatever the presence of outliers.

In this section, we describe our problem as the minimization of an energy function. This function takes the form of a sum of two terms, which are respectively defined in subsections § 3.2 and § 3.3.

3.1. A distance between a point cloud and a surface

To be able to state the problem as an optimization, the very first thing we need is to define a distance between the point cloud P and a surface S . One natural quantity to consider when dealing with distance between two geometrical objects P and S is the Hausdorff distance defined as:

$$H(S, P) := \max \left(\sup_{q \in S} \inf_{p \in P} \|p - q\|, \sup_{p \in P} \inf_{q \in S} \|p - q\| \right)$$

The Hausdorff distance is a maximum over two terms: its minimization ensures that every point of S is not too far from the point cloud, but also that every point of P is close to the surface. This behavior is however not robust to outliers as a single outlier point in P can make the distance

arbitrarily large. We additionally require that our objective function is smooth for the sake of being able to use an efficient optimizer, which is not the case of the Hausdorff distance here.

Since the point cloud P corresponding to the data is fixed in our case, we propose a variation of the Hausdorff distance in the form of an energy $E(S)$, function of the (moving) surface S . The underlying idea is the same as in the Hausdorff distance (refer to Figure 3), but we replace the max by a sum to obtain a smooth energy:

$$E(S) := \mathcal{F}(S) + \gamma \mathcal{G}(S), \quad (1)$$

where \mathcal{F} quantifies the distance from the point cloud to the surface, \mathcal{G} is the distance from the surface to the point cloud, and the parameter γ tunes relative weights of \mathcal{F} and \mathcal{G} (more details are given in Section 3.3).

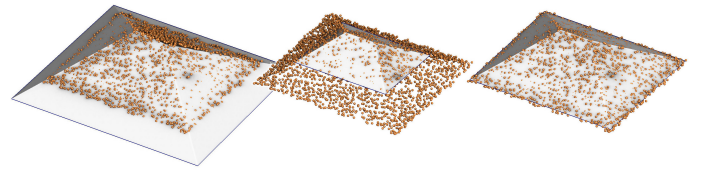


Figure 3: We need both terms $\mathcal{F}(S)$ and $\mathcal{G}(S)$ in the energy $E(S)$, otherwise the fitting does not produce the expected result. **Left:** $\text{argmin } \mathcal{F}$; **middle:** $\text{argmin } \mathcal{G}$; **right:** $\text{argmin } E$.

To define the terms $\mathcal{F}(S)$ and $\mathcal{G}(S)$, we need to introduce two projection functions. Let $q \in S$ be a point on the surface S , and $p \in P$ a point from the 3D point cloud P .

- $\pi_P(q) := \text{argmin}_{p \in P} \|p - q\|$ is the projection of q to the point cloud, i.e. the closest point of P from q ;
- $\pi_S(p) := \text{argmin}_{q \in S} \|q - p\|$ is the projection of p to the surface, i.e. the closest point of S from p .

Figure 4 illustrates the projection functions. We will define shortly both terms $\mathcal{F}(S)$ and $\mathcal{G}(S)$ as integrals of these projections, they can be computed in a very efficient manner with aid of restricted Voronoi diagrams.

3.2. $\mathcal{F}(S)$: distance from P to S

Fitting a surface primitive onto a point data set is a problem that can be seen as a special three-dimensional case of a more general regression problem. Unlike more simple primitives like planes or spheres, our roof templates do not allow for the computation of a closed form of the optimal result. Yet, the sum of squared distances from points of P to S is still a natural thing to consider. The issue of robustness to outliers for a regression problem has been extensively studied in various works on M-estimators and robust least-square regression [37]. As this part of our problem shares many characteristics with the framework of M-estimators, we introduce a potential function σ and define our first energy term \mathcal{F} as follows:

$$\mathcal{F}(S) := \sum_{p \in P} \sigma (\|p - \pi_S(p)\|^2) \quad (2)$$

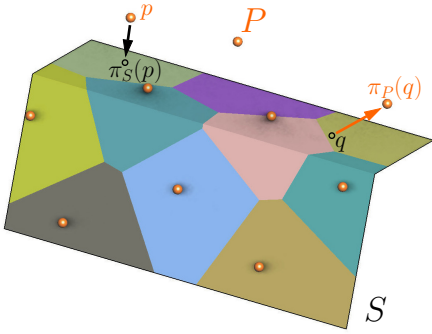


Figure 4: Our energy measures the distance between the point cloud P (in orange) and a surface S (the gable roof). To compute the distance, we need projection functions π_P and π_S that give the closest point of the point cloud and the surface, respectively. Then, we integrate the functions for each point of the surface (resp. point cloud), and this computation is very efficient due to a restricted Voronoi diagram calculation (shown in color).

where σ is Tukey’s biweight function [38] defined as follows:

$$\sigma : x \mapsto \begin{cases} 1 - (1 - (x/\delta)^2)^3 & \text{if } |x| < \delta \\ 1 & \text{otherwise} \end{cases} \quad (3)$$

In contrast to many other functions used for M-estimators, Tukey’s biweight is of class C^2 and thus matches our requirements for a smooth energy function. The biweight function σ presents several properties of interest in our case. First, $\sigma(0) = 0$ and $\sigma'(0) = 0$, which means that the energy associated with points that are near the surface is low and has low gradient. Second, $\sigma(x) = 1$ and $\sigma'(x) = 0$ for any $x > \delta$, which means that δ naturally defines a threshold for points to be considered outliers and not contributing to the evolution of the energy. In practice, δ is estimated as the mean distance between points and their k -closest neighbors in the point cloud, with $k = 10$. Minimization of the term $\mathcal{F}(S)$ guarantees that the model is stretched to capture the full extent of the roof points without being sensible to irrelevant points that are too far away.

3.3. $\mathcal{G}(S)$: distance from S to P

While minimizing the \mathcal{F} term allows to finding satisfactory position and orientation of the surface S , the regression approach does not allow to fit the boundary of the surface to feature lines in the point cloud (see Figure 3). Indeed, large portions of the surface area far away from the points are not penalized by \mathcal{F} . To avoid this situation, we add a second term \mathcal{G} to our energy function, defined as:

$$\mathcal{G}(S) := \int_S \|q - \pi_P(q)\|^2 dS(q) \quad (4)$$

This energy term is inspired by the Voronoi Square Distance Minimization (VSDM) algorithm [18], where a similar energy is used for fitting a surface onto another surface. As depicted in Figure 3, \mathcal{G} alone could not be considered as an energy function, since points of P that are not considered as the closest point by a point in S have no influence on the energy, and thus will never be considered in a minimization

process. As a consequence, it would not correctly capture the edges of the roof inside the point cloud, and could even reach a global minima $\mathcal{G} = 0$ by moving the roof to a single point of P . Both terms therefore play a different but complementary role in the overall solution.

The parameter γ tunes relative weights of \mathcal{F} and \mathcal{G} . To be independent of global scaling or the point set density, we set $\gamma = 1$ for our point cloud of resolution 30 pts/m², and preserve the same ratio for decimated point clouds. This value was used to generate all results, except in Figure 8 and the last line of Figure 12 where tuning it allows to capture the step edge of the roof.

It is interesting to notice that although both terms of the energy defined as sums of distances over every point of P (for \mathcal{F}) and S (for \mathcal{G}), their expression differs naturally in the same way that P and S are two different objects, one being a discrete set of points and the other one being a continuous surface. It is also worth noting that we are placing ourselves in a context where outliers can only appear in P while the whole surface of S should be considered a relevant part to be fitted onto the points. This logically implies that the outlier robustness is only expressed in \mathcal{F} and not in \mathcal{G} .

4. Constrained optimization

In summary, given a (mobile) surface S and a (constant) point cloud P , we have defined the energy $E(S)$, whose minimization fits S onto a part of P .

In practice, S is a fixed connectivity triangular mesh, and its geometry is defined by the position of the vertices. We define the position of the i -th vertex as $S_i := RX_i + T$, where $X_i = (x_i, y_i, z_i)^T \in \mathbb{R}^3$ is the coordinates of the vertex in a local template basis and R, T represent a rigid transformation, i.e. a rotation matrix $R \in \mathbb{R}^{3 \times 3}$ and a translation vector $T \in \mathbb{R}^3$ of this local basis to the canonical basis of \mathbb{R}^3 .

Our objective is to fit the template onto the most suitable position inside the point cloud via optimizing for these coordinates $X \in \mathbb{R}^{3n}$ (template deformation), i.e. the concatenation of the X_i as well as for the rigid transformation (R, T) of the template. We then rewrite Equation (1) as follows:

$$E(X, R, T) := \mathcal{F}(X, R, T) + \gamma \mathcal{G}(X, R, T) \quad (5)$$

Recall that we have yet to incorporate geometrical constraints put on the template to fit, so our optimization problem can be written as:

$$\operatorname{argmin}_{X, R, T} E(X, R, T) \quad \text{subject to constraints on } X. \quad (6)$$

In this section, we first show how we deal with the geometrical constraints (§ 4.1) via a reduction of variables, then we detail the computation of the energy (5) as well as its gradient using restricted Voronoi diagrams (§ 4.2), which enables us to call a quasi-Newtonian solver.

4.1. Constraints

As depicted in Figure 2, our roof models have to comply to a variety of constraints put on their vertices (planarity of the roof panes, alignment of gutters, horizontality of the roof's ridge, *etc.*). These conditions can be expressed as equations that link the coordinates of X . In our optimization we support linear constraints on X that take the form $\sum_{i,j} \alpha_{ij} X_{ij} = 0$ or $\sum_{i,j} \beta_{ij} X_{ij} \geq 0$.

Note that some constraints are readily linear in terms of vertex coordinates of our surface S (e.g. equality of the gutter vectors), whereas others are not (e.g. orthogonality or collinearity). Separating the local basis rigid transformation R, T from the mesh deformation X is a common practice [35, 13, 14] for managing such non linearities. This gives to all the considered constraints a linear form when expressed in the local basis.

Let us show how we model the constraints shown in Figure 2. First we define the local basis so that \overrightarrow{AD} is collinear to the axis Ox , namely $y_A = y_D$. The constraints can then be expressed as follows (refer to Figure 5):

- $\overrightarrow{AD} = \overrightarrow{BC}$ translates into $x_D = x_C$ and $y_B = y_C$;
- $\overrightarrow{AD} \cdot \overrightarrow{AB} = 0$ can be developed as the two equations $x_A = x_B$ and $y_A = y_D$;
- The collinearity $\overrightarrow{EF} = \lambda \overrightarrow{AD}, 0 \leq \lambda \leq 1$ as $y_E = y_F$ and $x_A < x_E < x_F < x_D$.

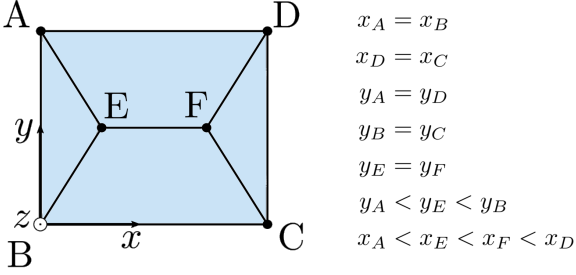


Figure 5: Roof model of Figure 2 where constraints have been expressed as equalities and inequalities between coordinates in the local basis.

Linear Equalities. The m linear constraints expressed onto the $3n$ variables in X can be written in a $m \times 3n$ matrix C such that $CX = 0$ (for example, in Figure 5, $m = 5$). From this, our goal is to compute a set of $3n - m$ reduced variables Y that are independent from each other, as well as a transformation between Y and X such that for any Y , the X obtained through this transformation satisfies $CX = 0$. To this end, we seek a linear relationship of form:

$$X = MY \quad (7)$$

where M is a $3n \times (3n - m)$ full rank matrix. This yields $CMY = 0$ which has to be true for all $Y \in \mathbb{R}^{3n-m}$. A sensible choice for M is therefore to take an orthonormal basis of the null space of C arranged in columns. This makes it easily computable through a QR decomposition of C .

Linear Inequalities. In addition to the linear equalities, we want to prevent the roof template from getting into unrealistic positions, for instance where triangular faces intersect each other. Those positions are often the result of two variables switching the order of their values. Given our roof initialization and parametrization, such a flip is never required in order to reach to global minima of the energy. This is where inequality constraints of general form $\sum_{i,j} \beta_{ij} X_{ij} \geq 0$ come into place. While they are not strictly necessary, they act as a regularization on the energy by removing local minima associated with non-realistic positions of the template. As we are not interested in solutions where two vertices coincide, those inequalities are not thought to be tight, but more as a guideline where $\sum_{i,j} \beta_{ij} X_{ij}$ should be sufficiently large.

It is unfortunately impossible to enforce this type of constraints in the same way as equalities. Since inequality expressions only have to be loosely positive, we choose to handle them by adding an extra term to the energy function.

For a constraint of form $\sum_{i,j} \beta_{ij} X_{ij} \geq 0$, we add to the energy E a logarithmic-exponential barrier term \mathcal{I} defined as:

$$\mathcal{I}(X) = a \log(1 + \exp(-b \sum_{i,j} \beta_{ij} X_{ij})), \text{ where } a, b > 0 \quad (8)$$

Tweaking hyperparameters a and b allows us to control the slope and the magnitude of the barrier term. Having a large value for a allows to efficiently suppress local minima of E that present flipping or unacceptable positions. The value of b controls the magnitude of the gradient of \mathcal{I} . If b is too large, solutions with vertices very close to each other are not effectively discarded, but with b too low, the slope of the gradient when variables do not respect the constraint might not be enough to enforce it. In practice, we take $a = 10^5$ and $b = 3$.

In conclusion, taking into account both equality and inequality constraints in the system, our constrained minimization problem (6) can be fully stated as follows:

$$\underset{Y, R, T}{\operatorname{argmin}} \left\{ \mathcal{F}(MY, R, T) + \gamma \mathcal{G}(MY, R, T) + \sum_k \mathcal{I}_k(MY) \right\} \quad (9)$$

4.2. Energy Minimization

All the terms \mathcal{F} , \mathcal{G} and \mathcal{I} in Equation (9) are of class C^2 , which is a non-trivial result for \mathcal{G} [39]. We propose to minimize it using second-order quasi-Newtonian methods like L-BFGS algorithm [19]. This implies a computation of both the energy E and its gradient ∇E .

Throughout this section, we adopt the following convention for partial derivatives: if $u = (u_x, u_y, u_z)$ is a point of \mathbb{R}^3 and f is a real-valued function depending on variables u_i , we will note $\frac{\partial f}{\partial u}$ the vector $\left(\frac{\partial f}{\partial u_x}, \frac{\partial f}{\partial u_y}, \frac{\partial f}{\partial u_z} \right)$.

We start by computing the gradient according to the coordinates $S_i = RX_i + T$ of vertices of S expressed in the global basis. By an abuse of notations, we will denote by S the vector made from the concatenation of the S_i when

it does not introduce an ambiguity. Namely, we start by computing $\nabla_S E$, and the gradient of E according to the relevant variables will be later obtained via the chain rule.

Computation of \mathcal{F} . The computation of the first energy term \mathcal{F} (Equation (2)) is straightforward: the squared distance $\|p - \pi_S(p)\|^2$ can be computed with a point-to-triangle distance looping over every triangle in S , before being fed inside the σ function (Equation (3)).

For the gradient $\nabla_S \mathcal{F}$, given a point $p \in P$, we obtain the vector

$$\frac{\partial \mathcal{F}}{\partial \pi_S(p)} = 2(p - \pi_S(p))\sigma'(\|p - \pi_S(p)\|^2) \in \mathbb{R}^3$$

where σ' is the derivative of σ . Noting by $C(c_1, c_2, c_3)$ the triangular face of S containing $\pi_S(p)$ (c_1, c_2 and c_3 being three triplets of variables S_i), and $(\lambda_1, \lambda_2, \lambda_3)$ the barycentric coordinates of $\pi_S(p)$ in C , the contribution of p to the gradient $\nabla_S \mathcal{F}$ concerns only the coordinates of the c_i with:

$$\frac{\partial \mathcal{F}}{\partial c_i} = 2\lambda_i(p - \pi_S(p))\sigma'(\|p - \pi_S(p)\|^2), \quad (10)$$

and these expressions need to be summed for each $p \in P$.

Computation of \mathcal{G} . For the second term \mathcal{G} (Equation (4)), we follow [18] and compute Voronoi diagram of P restricted to S . Let us denote by Ω_p the Voronoi cell of $p \in P$, the diagram partitions S into polygonal regions $\Omega_p \cap S$ where the closest point $\pi_S(p)$ is constant. The integral therefore can be decomposed as follows:

$$\mathcal{G} = \int_S \|q - \pi_P(q)\|^2 dS(q) = \sum_{p \in P} \int_{\Omega_p \cap S} \|p - q\|^2 dS(q) \quad (11)$$

For a given $p \in P$, we partition again the region $\Omega_p \cap S$ into triangles $T(v_1, v_2, v_3)$ (see Fig. A.15) and we can express the analytical value of the integral over each triangle using Theorem 2.1 of [40]:

$$\mathcal{G}_T = \frac{\text{area}(T)}{6} \sum_{1 \leq i \leq j \leq 3} (v_i - p)(v_j - p) \quad (12)$$

The computation of $\frac{\partial \mathcal{G}_T}{\partial v_k} \in \mathbb{R}^3$, $\nabla_S \mathcal{G}_T$ and finally $\nabla_S \mathcal{G}$ are detailed in Appendix Appendix A.

For the penalty terms \mathcal{I} , values and gradients need to be computed using coordinates X in the local basis. Their expressions are straightforward.

Computing gradients along the rigid transformation (R, T) and local variables X . Coordinates X_i in the local basis are defined from the global basis coordinates S_i through a rigid transformation:

$$S_i = R(\theta)X_i + T \in \mathbb{R}^3 \quad (13)$$

For the sake of clarity, we restrict our attention only to rotations around the vertical axis Oz . This is sufficient

to handle our main application case (roofs fitted into airborne LIDAR acquisition), and the generalization to an arbitrary rotation parametrized by three Euler angles is straightforward. Let θ be the angle of rotation along Oz and $T = (T_x, T_y, T_z)$ the translation vector. We obtain:

$$\frac{\partial E}{\partial X_i} = R_z(-\theta) \frac{\partial E}{\partial S_i} \quad (14)$$

which yields the gradient $\nabla_X E$ given the previously computed gradient $\nabla_S E$. Additionally, we are interested in the derivatives $\frac{\partial E}{\partial \theta}$ and $\frac{\partial E}{\partial T}$ according to the rigid transformation. For the rotation component, we get:

$$\frac{\partial E}{\partial \theta} = \sum_i \frac{\partial E}{\partial S_i} \cdot R_z\left(\frac{\pi}{2}\right)(S_i - T) \quad (15)$$

and for the translation component, we have:

$$\frac{\partial E}{\partial T} = \sum_i \frac{\partial E}{\partial S_i} \quad (16)$$

Details for the computation of $\frac{\partial E}{\partial \theta}$ and $\frac{\partial E}{\partial T}$ are given in Appendices Appendix B and Appendix C.

Computing gradients along reduced variables Y . With the above expressions, we have all the ingredients to minimize the energy w.r.t the variables (X, θ, T) , which is only a half of the problem since it does not take into account the linear equality constraints (7). While the computation of $\nabla_S E$ and $\nabla_X E$ are necessary steps, we are really interested in the partial derivatives along the real $3n - m$ degrees of freedom Y (assuming m is the number of distinct equality constraints) along the derivatives corresponding to the rotation R and the translation T . Given Equation (7) $\nabla_Y E$ can be easily computed using the chain rule as follows:

$$\nabla_Y E = M^\top \nabla_X E \quad (17)$$

which concludes on the computation of E and its gradients according to the relevant variables.

5. Results and Applications

Evaluating the quality of our method is challenging for two reasons. Firstly, the fitting of a roof template on a point cloud dataset is an ill-posed problem since the ground-truth information has been lost during acquisition and is not easy to retrieve.

Secondly, no clear and global quantitative metric has been established in the state of the art to evaluate the quality of a fitting, and therefore objectively compared to existing methods.

Our energy E (defined by Equation (6)) is an attempt at defining such a metric. In order to test the quality of our metric, we define as our reference position either a manually fitted template or the result of the fitting on data that has been manually cleaned (like in Tables 1 and 2). We then evaluate the final position of a fitting by computing the Hausdorff distance between its set of vertices and the set of

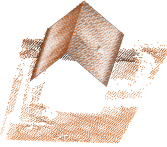
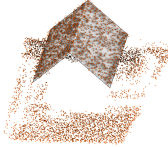
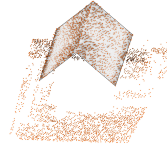
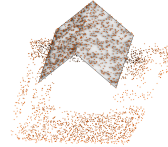
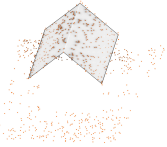
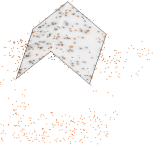
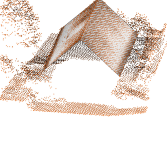
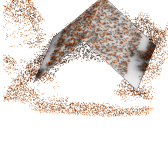
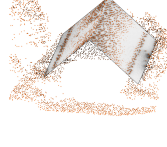
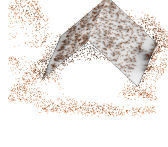
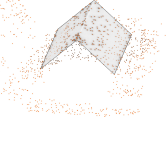
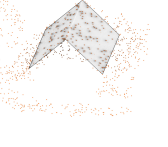

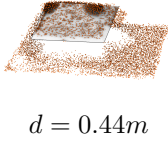
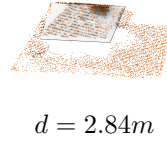
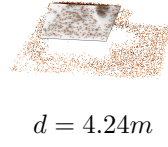
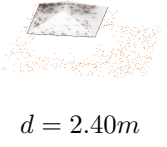
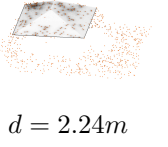
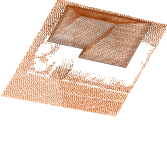
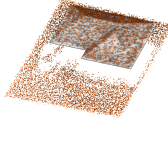
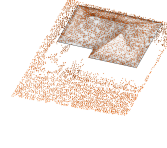
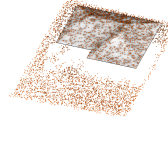
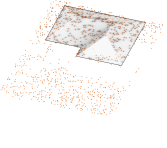
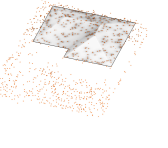
| | 100% | Noisy 100% | 50% | Noisy 50% | 10% | Noisy 10% |
|-----------------------|---|---|---|--|---|---|
| Gable roof |  |  |  |  |  |  |
| | Reference | $d = 0.27m$ | $d = 6.80 \times 10^{-2}m$ | $d = 0.20m$ | $d = 0.53m$ | $d = 0.44m$ |
| Gable roof with trees |  |  |  |  |  |  |
| | Reference | $d = 2.94m$ | $d = 1.93m$ | $d = 2.13m$ | $d = 0.53m$ | $d = 0.66m$ |
| Hipped roof |  |  |  |  |  |  |
| | Reference | $d = 0.44m$ | $d = 2.84m$ | $d = 4.24m$ | $d = 2.40m$ | $d = 2.24m$ |
| Cross hipped roof |  |  |  |  |  |  |
| | Reference | $d = 0.43m$ | $d = 0.34m$ | $d = 0.46m$ | $d = 0.95m$ | $d = 0.87m$ |

Table 1: Experiment showing the robustness of our method to poor quality point clouds. *Noisy* indicates the presence of uniform distribution of noise of amplitude 30cm added to the position of each points. Percentages indicate the proportion of points remaining in the cloud. Point clouds are shown in orange and we represent the final position of the roof model on the point cloud. The initial position of the roof is defined by aligning the cloud barycenter with the model barycenter, and translating the model 3m upwards. Rotation was the same for all test cases, but arbitrary. The error d is estimated quantitatively as the Hausdorff distance between the fitted template and a reference template that is either human generated or the result of a fit on the cleaned point cloud.

vertices of this reference, called d . Although this distance only takes into account vertices of the roof template and not the whole surface, areas of interest will often be edges of the roof, which this metric correctly captures. In all this section, distance and numerical scores given in text and figures will correspond to this Hausdorff distance and will be given in meters (m).

Our point clouds have been selected from Lidar acquisitions of the towns of Breuschwickersheim and Strasbourg in France that are freely available online [1]. In order to isolate the effect of various defects on the point clouds, we extract by hand some reference point clouds and manually get rid of different layers of outliers in order to generate different distributions admitting the same optimal solution for the roof placement. In addition to that, test cases were selected for semi-detached roofs for the purpose of testing different types of junctions for composite roofs (Figure 8 and supplementary materials).

We base ourselves on a pre-built library of roof templates (Figure 6) available as triangulated mesh on which we manually define geometrical constraints (Figure 7). These models cover the majority of the real world detached roofs, with the exception of flat roofs, which can be handled with a flat rectangular model. For more complicated roof shapes, say

in dense urban areas, our method considers the connected roof as a cluster of those models and aims at fitting them separately.

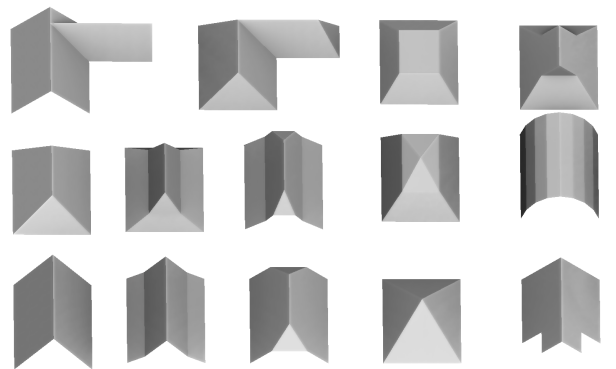


Figure 6: Our method utilizes a library of diverse roof models, allowing us to cover the majority of real world cases.

In this section, we evaluate the robustness and quality of our method over two families of parameters. We first focus on the behavior of the fitting with relation to the quality of the point cloud. Secondly, we perform experiments on the initial position of the templates, and the ability of the

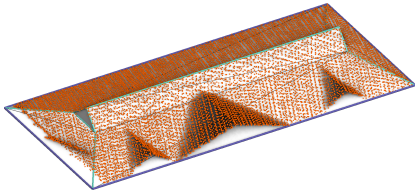


Figure 7: Example of a Dutch roof fitting.

method to converge to a good solution when this initialization is far away from the optimal. Finally, we will discuss the performance, the advantages and the failure cases of the method.

5.1. Robustness to quality of the point cloud

We start by testing the convergence behavior of the method against the quality of the point cloud. Recall that we designed the energy function to be robust to outliers (Section 3.2), that is to say large portions of the point cloud that do not correspond to the specific roof shape we want to fit. This has to be true even if the outliers are coherent, for instance in the case of a tree or a neighboring roof. In addition to that, we test our method on altered data, to check that it is fairly independent on points cloud density, as well as robust against acquisition noise. Tests are therefore split into three categories:

- Point cloud alteration (noise and decimation)
- Presence of other structures (antennas, shutters, chimneys, ground, trees, etc.)
- Presence of other roofs

Noisy acquisition and point cloud resolution. The first test we run can be seen as a sanity check where we assess that the result obtained by our method does not change much when the density of the point cloud is changed and when noise is added to the point cloud. To this end, we compare in Table 1 the fitting results of the same model with the same initialization on a point cloud that has been artificially altered. Three setups of point cloud decimation are considered: full point cloud, randomly pruning 50% and randomly pruning 90% of the points. The artificial noise added follows a uniform distribution with zero mean and a maximal amplitude of 30cm.

The initial position of the template was obtained by matching the barycenters of both the point cloud and the template, and then applying a fixed (but arbitrary) rotation of 0.2 radians and an upwards translation of 3 meters. This ensures that the convergence does not benefit from an initial position that is too well-aligned nor too close from the optimal position. Starting from above the point cloud is also a common practice in our tests, since most outliers are laid beneath or next to the area of interest. Experiments shown in Table 1 have been run on three different models of roofs (a gable roof, a rectangular hipped roof and a cross-hipped roof) and three corresponding points clouds. In the case of the gable roof, we run the experiment on a cloud that presents trees and on the same cloud where trees

were manually deleted. We observe that in the first case, the noisy points of the trees are captured by the model, which leads to a great imprecision on the edges of the roof. In a context with fewer outliers near the roof (Gable roof without trees and cross-hipped roof), we observe that the deviation ranges from a few centimeters to a few dozen centimeters even with 90% of the point cloud pruned, which is acceptable given the range of the perturbation. On the hipped roof, however, scores of the Hausdorff distance are high as we observe that perturbing the point cloud leads to the roof being placed at a right angle as its ridge becomes perpendicular. This leads to visually close results but we consider these cases as failed ones.

Robustness to coherent outliers. In most real world airborne Lidar point cloud, the acquisition is made over every object present in the landscape. While modern Lidar technology gives a form of segmentation, the presence of non-relevant object for our uses is still the norm. When fitting a roof model, structures on the roof like antennas or windows as well as adjacent structures like trees and cars should be ignored.

Given point clouds that were manually cut into different layers of outliers, we perform an optimization with our method on those different versions, thus comparing the performance with and without certain classes of non-relevant structures. We consider four cases in addition to the perfectly segmented reference: small additional structures on the roof (chimneys, antennas, etc.), large additional structures, ground and finally ground and trees. Results are presented in Table 2. Initialization was similar for all test cases and set like in the previous experiment: 3m up the barycenter of the point cloud and arbitrarily rotated of 0.2 radians. For every case, we measure the Hausdorff distance from the perfectly segmented case. We observe that in the large majority of cases, the influence of outliers remains small, with an observed deviation in our metric of the order of the centimeter. On the hipped roof, we observe that the presence of an adjacent structure (like a porch or a veranda) leads to imprecision on the edge. The most remarkable failure cases happen on the hipped and the L-shaped hipped roofs with presence of both trees and ground, where a tree directly touching the edge of the roof makes the algorithm consider those points as valid, which also lead to a great offset.

Roof clusters and adjacencies. In dense areas where buildings are adjacent to one another, we can use our method to fit one roof at a time. Other roofs present in the point cloud have to be considered as outliers, but the only prior telling which roof point cloud should be fitted is the template initialization. In Figure 8, we present the result we obtained on some selected configurations of interest, presenting results for both a gable roof and a cross hipped roof. While correct results can be obtained by tuning the parameter γ and starting close from the wanted position, we conclude that these configurations remain especially hard to optimize with our method, since they can lead to acceptable solutions that do not correspond to the reality (see Figure 12). They,




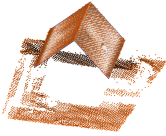
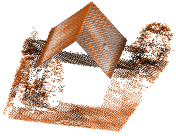



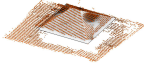
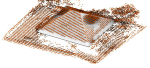
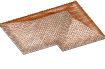

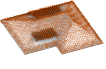
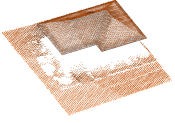
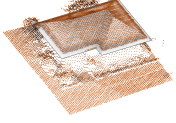
| | Segmented | Small Structures | Large Structures | Ground | Ground & Trees |
|-------------------|---|---|---|---|---|
| Gable Roof |  |  |  |  |  |
| | Reference | $d = 2.73 \times 10^{-5}m$ | $d = 2.96 \times 10^{-3}m$ | $d = 2.06 \times 10^{-5}m$ | $d = 2.91 \times 10^{-5}m$ |
| Hipped Roof |  |  |  |  |  |
| | Reference | $d = 8.37 \times 10^{-2}m$ | $d = 9.30 \times 10^{-1}m$ | $d = 1.93m$ | $d = 2.60m$ |
| Cross Hipped Roof |  |  |  |  |  |
| | Reference | $d = 2.31 \times 10^{-2}m$ | $d = 9.42 \times 10^{-2}m$ | $d = 1.48 \times 10^{-2}m$ | $d = 1.01m$ |

Table 2: Experiment showing the robustness of our method to various outlier distributions present in real world data. Point clouds are shown in orange and we represent the final position of the roof model on the point cloud. The initial position of the roof is defined by aligning the cloud barycenter with the model barycenter, and translating the model 3m upwards. Rotation was the same for all test cases, but arbitrary. The error d is defined as in the Table 1.

however, can greatly benefit from preprocessing steps such as plane segmentation.

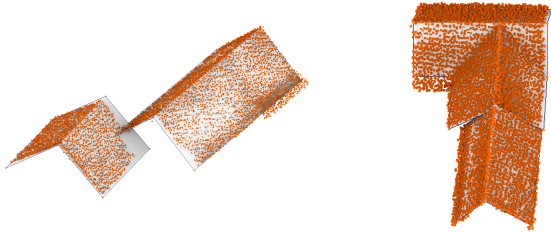


Figure 8: Fitting results obtained on adjacent roofs. Left : two gable roofs. Right : a gable T-shaped roof and a simple gable roof. When fitting onto a specific roof, the other one is considered as part of the outlier distribution.

5.2. Impact of initialization

Until now, experiments were performed using an arbitrary initialization roughly registered to the given point cloud. Yet, minimizing our energy function is a task that is highly dependent on the starting position in our variable space. After evaluating our method against variations in the point cloud, we present here a series of tests in order to validate that the method is able to find the correct position of the roof for a range of reasonable initializations.

Recall that in Equation (9), we optimize our energy function along three sets of variables: translation T , rotation R and reduced variables Y obtained after subjecting vertex positions to linear constraints. We present here a protocol to evaluate the optimization quality in function of the initial values of those three sets of variables independently. Namely, we consider variations of the initial position

of the roof in terms of translation, rotation and stretching of the model.

Translation. We initialize the roof model by applying a translation to the reference position. We test twelve different directions evenly distributed on a sphere for three different magnitudes of translation: 1, 3, and 5 meters from the optimal position (see Figures 9). Next to the point cloud used, we represent the translation as twelve vectors of corresponding length. Colors match the computed Hausdorff distance, with 0m being fully green and 2m being fully red. We observe that in practice, translations of small magnitude pose no problem optimizing, while translations of greater magnitude tend to fail. When the template is far from the cloud, it considers only a part of the cloud as most points are considered outliers and thus may converge to only a partial fitting (see Figure 12).

Rotation. For the rotation component, we initialize the model with ten different angles around the Z axis, centered on the middle of the optimal roof position. The layout of the results is depicted in Figure 10, where the initial positions form the inner ring of templates, and the final positions correspond to the outer ring. We observe that for the gable roof, the optimization is able to retrieve the correct orientation whatever the orientation, while on non-symmetrical roofs like a cross-hipped roof, the best position is only achieved for an angle of $2i\pi/n$ from the optimal, with $i \in [0, n - 1]$ and $n = 10$. A greater angle usually leads to a local minimum where the roof is inverted (see Figure 12).

Stretching. Finally, we carry out tests by stretching the reference model. Stretching is applied by a factor of $[0.5, 1.,$

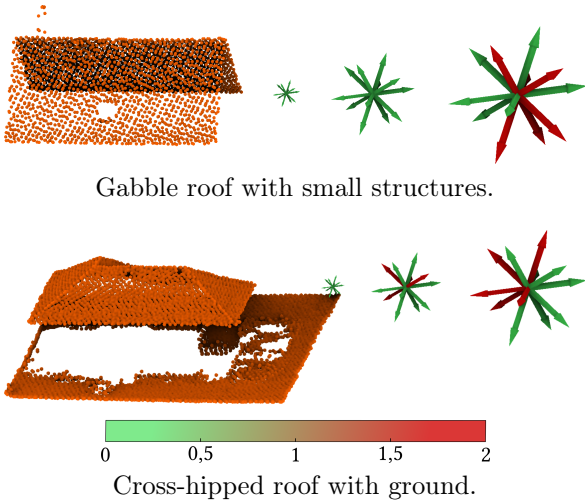


Figure 9: Translation experiments on a gabble roof with small structures and a cross-hipped roof with ground. Arrow color represents the success of our method when initialized to the reference position, translated by the corresponding vector.

1.5] in the x, y and z directions, leading to 27 different cases. On Figure 11, we present the results for a cross-hipped roof with ground. We observe that failure cases occur for shrunk models, which tend to converge to only a part of the point cloud.

5.3. Discussion and future work

Running time. Our algorithm is implemented in C++, and uses the restricted Voronoi diagram and HL-BFGS present in the Geogram library [41]. The experiments were carried out on an Intel(R) Xeon(R) E-2276M CPU (clocked at 2.8 GHz). Time to convergence can take up to 20 seconds for initializations that are far from the optimal position since the solver needs more steps to achieve convergence, and the restricted Voronoi diagram also takes longer to be computed when the model also spans the cells of outliers. When a good initialization is available, the timing drops to 2 – 3 seconds per fit. The most time-consuming part of our algorithm is the computation of the restricted Voronoi diagram (80% of runtime) needed to evaluate \mathcal{G} and $\nabla_S \mathcal{G}$. Typical point cloud in our data contains 1k to 10k points depending on the considered distribution of outliers.

From these observations, we consider as a future work an approximate computation of this energy where the restricted Voronoi diagram is no longer necessary. This could be done for instance by sampling the surface of the model and making the energy pointwise only.

Typical failure cases. We presented successful as well as failed cases in all of this section. From our experiments, we observed that failures occur when:

1. The roof model fits a part or the integrity of the outlier distribution. This especially happens when the initialization is too close from the said distribution (for instance, for a model initialized in the ground), or if the minimal distance between an outlier and a relevant

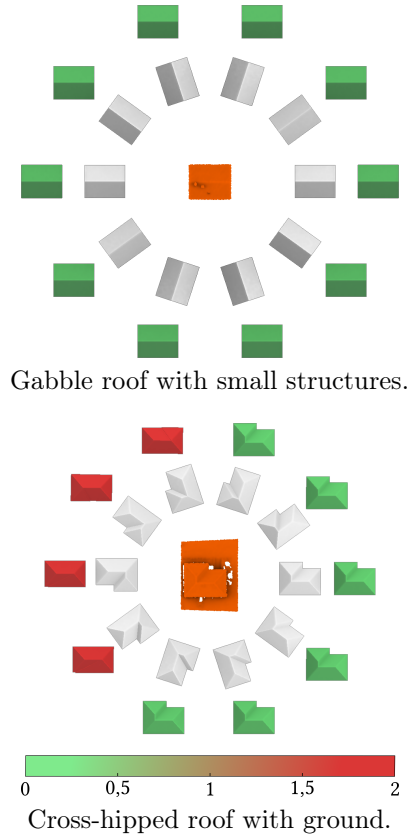


Figure 10: Rotation experiments. The point cloud is shown in orange in the middle. The first roof on the right of the point cloud is the ground truth. Other roofs in the inner ring (gray) are obtained by rotation of the ground truth. The second ring of roofs shows our result for each initialization. Whatever the initial rotated position of the template, the minimization scheme is converging to the correct position. Top ring : gabble roof with rotation. Bottom ring : cross-hipped roof.

point is small and the biweight function (Equation 3) is no longer able to distinguish the one from the other.

2. The roof model is initialized in a bad orientation that sticks the optimization in a local minima.
3. The point cloud presents an adjacent roof that is easily fitted by the template. While this gives a very low energy and good visual results, this is not wanted in most use cases.

Those three cases are shown in order in Figure 12. Fortunately, they can be easily avoided for the most part by:

1. Considering initializations that are above the point cloud (or more fine-tuned initialization that could be available in a processing pipeline, while out of the scope of this paper)
2. Testing the best of two initializations where we considered two initial angles α and $\alpha + \pi$
3. Tuning the hyperparameter γ and thus decrease the sensibility to neighboring points.

As expected, outliers that are close to the surface are hard to classify using only a function of the distance to the

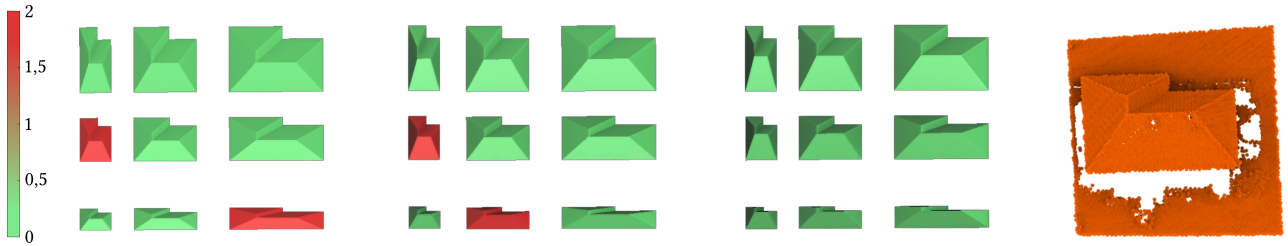
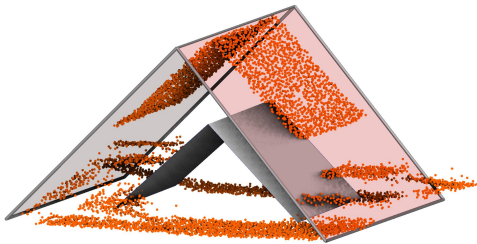


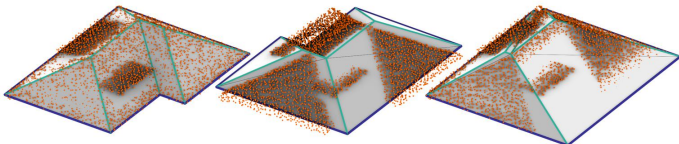
Figure 11: Stretching experiments on a crossed-hipped roof with floor. From left to right each set of 9 roofs has an increasing z stretch factor of $[0.5, 1., 1.5]$. The x stretch factor increases inside each line, and the y stretch increases in each row. Consequently, the center result is not stretched at all and represents our reference. Templates shown are the roof **initialization** and not its final position. Color map ranges from 0m (green) to 2m (red). The point cloud used is depicted on the right.

roof. In our future work, we expect a better behavior with energy functions that would be anisotropic in the normal direction of the plane. Another future direction is to fit more than a roof at once, in order to automatically segment a set of adjacent roofs.

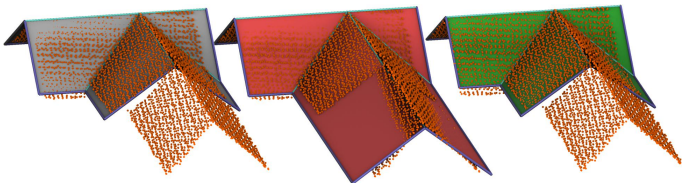
Better initialization. Experiments detailed in Subsection 5.2 show that the basin of attraction of the best fit is pretty large in general, even in the presence of noise and outliers. However, it is important to notice that a full automatic reconstruction algorithm would require a fair initial guess to start with, which could for instance be computed with footprints or cadastral data, plane fitting, etc. or given by a human operator in a human-machine interaction context.



Initialized to the solid white roof, our algorithm finds a local minimum (transparent roof) that includes ground points.



The cross-hipped roof (left), initialized by a rotation of π (middle), found a local minimum that mismatches the faces.



The step edge (left) is not detected by our algorithm (middle), but dividing γ by 10 solves the problem (right).

Figure 12: Typical failure cases of our algorithm.

Future works. Our method is designed to be as general as possible, opening three research opportunities. It can be applied to similar problems such as partial CAD object match-

ing as in Figure 13. However, since CAD models are more complex than roofs (higher number of vertices), it would be more efficient to compute the constraints automatically and optimize the algorithm to decrease its execution time. We can also improve the performance of roof reconstruction by using more sources of information such as cadastral data (example provided in Figure 14), colors, LIDAR reflexing intensity, or even results of other algorithms (segmentation, plane detection, etc.). This can simply provide good initializations for our method, or have a deeper impact by modifying the objective function. Finally, an automatic template selection based on the estimation of high level geometrical properties of the point cloud could also be a great improvement towards a fully automated pipeline.

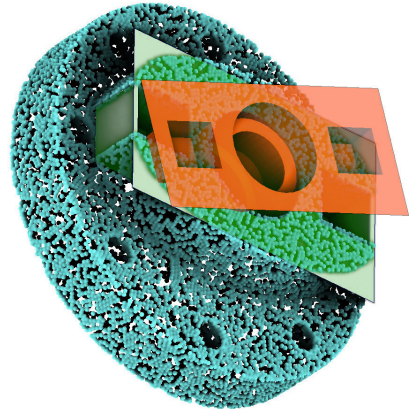


Figure 13: Our template fitting can be used for other applications, like partial CAD fitting. The initialization of the template is shown in red, the result in green.

Conclusion

The fast improvement of real-world acquisition technologies enabled the massive availability of point cloud data of all sorts, making the challenging task of structuring them a problem of first importance. In this work, we show that fitting a roof endowed by geometrical constraints onto LIDAR-acquired point clouds could be done efficiently through the optimization of an energy function, using quasi-Newton optimizers like L-BFGS and the fast computation of a restricted Voronoi diagram. Partial fitting of the point cloud,

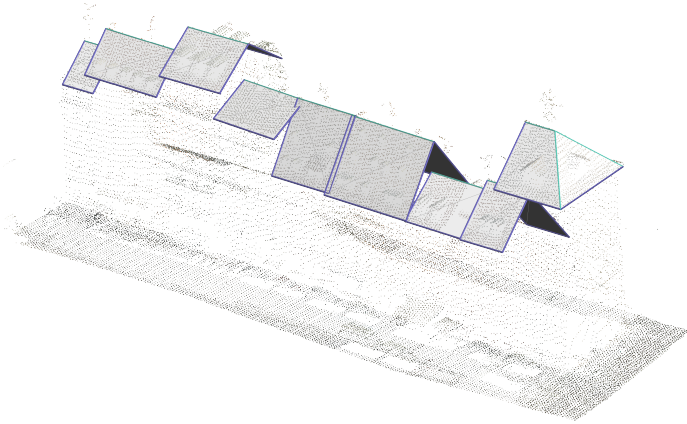


Figure 14: Our matching method can be used with cadastral data, allowing to treat more complex cases such as streets of the city center of Strasbourg [1].

that is, fitting in the presence of outliers, can also be performed. Our method yields satisfactory results as a standalone fitting procedure over a diverse panel of real world cases, especially for detached roofs. In more dense areas, embedding this method into a pipeline with state-of-the-art methods for finding fair initialization and/or applying segmentation pre-processing is far from being irrelevant.

Acknowledgements

We thank the company RhinoTerrain for supporting our research and providing us with roofs models. We also thank Philippe Slisse for the various point cloud datasets present in this article.

References

- [1] Lidar data, Ville et Eurometropole de Strasbourg https://data.strasbourg.eu/explore/dataset/odata3d_lidar/information (Nov. 2016).
- [2] M. Berger, A. Tagliasacchi, L. M. Seversky, P. Alliez, G. Guennebaud, J. A. Levine, A. Sharf, C. T. Silva, A survey of surface reconstruction from point clouds, *Comput. Graph. Forum* 36 (1) (2017) 301–329. doi:10.1111/cgf.12802. URL <https://doi.org/10.1111/cgf.12802>
- [3] S. Fleishman, D. Cohen-Or, C. T. Silva, Robust moving least-squares fitting with sharp features, *ACM Trans. Graph.* 24 (3) (2005) 544–552. doi:10.1145/1073204.1073227. URL <https://doi.org/10.1145/1073204.1073227>
- [4] R. Schnabel, R. Wahl, R. Klein, Efficient ransac for point-cloud shape detection, *Computer Graphics Forum* 26 (2) (2007) 214–226.
- [5] T. Le, Y. Duan, A primitive-based 3d segmentation algorithm for mechanical cad models, *Computer Aided Geometric Design* 52 (2017) 231–246.
- [6] Y. Li, X. Wu, Y. Chrysathou, A. Sharf, D. Cohen-Or, N. J. Mitra, Globfit: Consistently fitting primitives by discovering global relations, *ACM Trans. Graph.* 30 (4). doi:10.1145/2010324.1964947. URL <https://doi.org/10.1145/2010324.1964947>
- [7] M. Arıkan, M. Schwärzler, S. Flöry, M. Wimmer, S. Maierhofer, O-snap: Optimization-based snapping for modeling architecture, *ACM Trans. Graph.* 32 (1). doi:10.1145/2421636.2421642. URL <https://doi.org/10.1145/2421636.2421642>
- [8] G. K. L. Tam, Z. Cheng, Y. Lai, F. C. Langbein, Y. Liu, D. Marshall, R. R. Martin, X. Sun, P. L. Rosin, Registration of 3d point clouds and meshes: A survey from rigid to nonrigid, *IEEE Transactions on Visualization and Computer Graphics* 19 (7) (2013) 1199–1217.
- [9] P. J. Besl, N. D. McKay, A method for registration of 3-d shapes, *IEEE Transactions on Pattern Analysis and Machine Intelligence* 14 (2) (1992) 239–256.
- [10] A. Segal, D. Hähnel, S. Thrun, Generalized-icp., in: J. Trinkle, Y. Matsuoka, J. A. Castellanos (Eds.), *Robotics: Science and Systems*, The MIT Press, 2009. URL <http://dblp.uni-trier.de/db/conf/rss/rss2009.html#SegalHT09>
- [11] C. Stoll, Z. Karni, C. Rössl, H. Yamauchi, H.-P. Seidel, Template Deformation for Point Cloud Fitting, in: M. Botsch, B. Chen, M. Pauly, M. Zwicker (Eds.), *Symposium on Point-Based Graphics*, The Eurographics Association, 2006. doi:10.2312/SPBG/SPBG06/027-035.
- [12] R. B. Rusu, N. Blodow, M. Beetz, Fast point feature histograms (fpfh) for 3d registration, in: *2009 IEEE International Conference on Robotics and Automation*, 2009, pp. 3212–3217.
- [13] O. Sorkine, M. Alexa, As-rigid-as-possible surface modeling, in: *Proceedings of the Fifth Eurographics Symposium on Geometry Processing*, SGP '07, Eurographics Association, Goslar, DEU, 2007, p. 109–116.
- [14] R. W. Sumner, J. Schmid, M. Pauly, Embedded deformation for shape manipulation, in: *ACM SIGGRAPH 2007 Papers, SIGGRAPH '07*, Association for Computing Machinery, New York, NY, USA, 2007, p. 80–es. doi:10.1145/1275808.1276478. URL <https://doi.org/10.1145/1275808.1276478>
- [15] H. Li, B. Adams, L. J. Guibas, M. Pauly, Robust single-view geometry and motion reconstruction, *ACM Trans. Graph.* 28 (5) (2009) 1–10. doi:10.1145/1618452.1618521. URL <https://doi.org/10.1145/1618452.1618521>
- [16] S. Pellegrini, K. Schindler, D. Nardi, A generalisation of the icp algorithm for articulated bodies, in: *Proceedings of the British Machine Vision Conference*, BMVA Press, 2008, pp. 87.1–87.10, doi:10.5244/C.22.87.
- [17] F. Buonamici, M. Carfagni, R. Furferi, L. Governi, A. Lapini, Y. Volpe, Reverse engineering of mechanical parts: A template-based approach, *Journal of Computational Design and Engineering* 5 (2) (2018) 145 – 159. doi:https://doi.org/10.1016/j.jcde.2017.11.009. URL <http://www.sciencedirect.com/science/article/pii/S2288430017301392>
- [18] V. Nivoliens, D.-M. Yan, B. Lévy, Fitting polynomial surfaces to triangular meshes with voronoi squared distance minimization, *Engineering with Computers* 30 (3) (2014) 289–300.
- [19] D. C. Liu, J. Nocedal, On the limited memory bfgs method for large scale optimization, *Math. Program.* 45 (1–3) (1989) 503–528.
- [20] G. Vosselman, Building reconstruction using planar faces in very high density height data, *International Archives of Photogrammetry and Remote Sensing* 32 (3; SECT 2W5) (1999) 87–94.
- [21] K. Zhang, J. Yan, S.-C. Chen, Automatic construction of building footprints from airborne lidar data, *Geoscience and Remote Sensing*, *IEEE Transactions on* 44 (2006) 2523 – 2533. doi:10.1109/TGRS.2006.874137.
- [22] A. F. Elaksher, J. S. Bethel, et al., Reconstructing 3d buildings from lidar data, *International Archives Of Photogrammetry Remote Sensing and Spatial Information Sciences* 34 (3/A) (2002) 102–107.
- [23] P. Dorninger, N. Pfeifer, A comprehensive automated 3d approach for building extraction, reconstruction, and regularization from airborne laser scanning point clouds, in: *Sensors*, 2008.
- [24] Q.-Y. Zhou, U. Neumann, Fast and extensible building modeling from airborne lidar data, in: *Proceedings of the 16th ACM SIGSPATIAL international conference on Advances in geographic information systems*, ACM, 2008, p. 7.
- [25] Q.-Y. Zhou, U. Neumann, 2.5d dual contouring: A robust approach to creating building models from aerial lidar point clouds, in: K. Daniilidis, P. Maragos, N. Paragios (Eds.), *Computer Vision – ECCV 2010*, Springer Berlin Heidelberg, Berlin, Heidelberg, 2010, pp. 115–128.
- [26] F. Tarsha-Kurdi, T. Landes, P. Grussenmeyer, Extended ransac algorithm for automatic detection of building roof planes from

lidar data.

- [27] F. Lafarge, C. Mallet, Creating large-scale city models from 3d-point clouds: a robust approach with hybrid representation, *International journal of computer vision* 99 (1) (2012) 69–85.
- [28] D. Chen, L. Zhang, P. T. Mathiopoulos, X. Huang, A methodology for automated segmentation and reconstruction of urban 3-d buildings from als point clouds, *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing* 7 (10) (2014) 4199–4217.
- [29] M. Shahzad, X. X. Zhu, Automatic detection and reconstruction of 2-d/3-d building shapes from spaceborne tomosar point clouds, *IEEE Transactions on Geoscience and Remote Sensing* 54 (3) (2016) 1292–1310.
- [30] J. Martín-Jiménez, S. Del Pozo, M. Sánchez-Aparicio, S. Lagüela, Multi-scale roof characterization from lidar data and aerial orthoimagery: Automatic computation of building photovoltaic capacity, *Automation in Construction* 109 (2020) 102965.
- [31] J. Milde, C. Brenner, Graph-based modeling of building roofs, in: *Proceedings of the 12th AGILE Conference on GIScience, Hannover, Germany (on CD-ROM)*, 2009.
- [32] B. Xiong, S. O. Elberink, G. Vosselman, A graph edit dictionary for correcting errors in roof topology graphs reconstructed from point clouds, *ISPRS Journal of photogrammetry and remote sensing* 93 (2014) 227–242.
- [33] Q. Zhou, U. Neumann, 2.5d building modeling by discovering global regularities, in: *2012 IEEE Conference on Computer Vision and Pattern Recognition, Providence, RI, USA, June 16-21, 2012*, IEEE Computer Society, 2012, pp. 326–333. doi: 10.1109/CVPR.2012.6247692. URL <https://doi.org/10.1109/CVPR.2012.6247692>
- [34] A. Henn, G. Gröger, V. Stroh, L. Plümer, Model driven reconstruction of roofs from sparse lidar point clouds, *ISPRS Journal of Photogrammetry and Remote Sensing* 76 (2013) 17 – 29, terrestrial 3D Modelling. doi:<https://doi.org/10.1016/j.isprsjprs.2012.11.004>. URL <http://www.sciencedirect.com/science/article/pii/S0924271612002043>
- [35] H. Huang, C. Brenner, M. Sester, 3d building roof reconstruction from point clouds via generative models, 2011, pp. 16–24. doi: 10.1145/2093973.2093977.
- [36] H. Huang, C. Brenner, M. Sester, A generative statistical approach to automatic 3d building roof reconstruction from laser scanning data, *ISPRS Journal of Photogrammetry and Remote Sensing* 79 (2013) 29 – 43. doi:<https://doi.org/10.1016/j.isprsjprs.2013.02.004>. URL <http://www.sciencedirect.com/science/article/pii/S0924271613000476>
- [37] P. W. Holland, R. E. Welsch, Robust regression using iteratively reweighted least-squares, *Communications in Statistics-theory and Methods* 6 (9) (1977) 813–827.
- [38] A. E. Beaton, J. W. Tukey, The fitting of power series, meaning polynomials, illustrated on band-spectroscopic data, *Technometrics* 16 (2) (1974) 147–185.
- [39] Y. Liu, W. Wang, B. Lévy, F. Sun, D.-M. Yan, L. Lu, C. Yang, On centroidal voronoi tessellation—energy smoothness and fast computation, *ACM Transactions on Graphics (ToG)* 28 (4) (2009) 1–17.
- [40] J. B. Lasserre, K. E. Avrachenkov, The multi-dimensional version of $\int_b^a x^p dx$, *The American Mathematical Monthly* 108 (2) (2001) 151–154.
- [41] ALICE, Geogram, <http://alice.loria.fr/index.php/software/4-library/75-geogram.html>.
- [42] V. Nivoliens, Phd thesis: Échantillonnage pour l’approximation de fonctions sur des maillages, Ph.D. thesis, INRIA (2012).

Appendix A. Computation of $\nabla_S \mathcal{G}$

Computations of the gradient $\nabla_S \mathcal{G}$ were first made in [18] and [42], but have been adapted to our notations. We first recall the expression of \mathcal{G} in Equation (11) and its decomposition as a sum over polygonal regions where $\pi_P(q)$ is constant:

$$\mathcal{G} = \int_S \|q - \pi_P(q)\|^2 dS(q) = \sum_{p \in P} \int_{\Omega_p \cap S} \|p - q\|^2 dS(q)$$

The polygonal areas $\Omega_p \cap S$ can be partitioned into triangles, on which we can calculate the exact formula for the integral. Let $p \in P$ be a point and let $T(v_1, v_2, v_3)$ be a triangle of the subdivision of $\Omega_p \cap S$. The contribution of triangle T on the energy \mathcal{G} , noted \mathcal{G}_T is:

$$\mathcal{G}_T = \frac{\text{area}(T)}{6} \sum_{1 \leq i < j \leq 3} (v_i - p)(v_j - p) \quad \text{and} \quad \mathcal{G} = \sum_{p \in P} \sum_{T \in \Omega_p \cap S} \mathcal{G}_T$$

The expression of \mathcal{G}_T depends on the coordinates of the v_k , which are not coordinates of X in general. Using the chain rule, we get:

$$\frac{\partial \mathcal{G}_T}{\partial x_i} = \sum_k \frac{\partial \mathcal{G}_T}{\partial v_k} \cdot \frac{\partial v_k}{\partial x_i}$$

The first term $\frac{\partial \mathcal{G}_T}{\partial v_k}$ is obtained by directly differentiating \mathcal{G}_T according to the v_k , $k=1,2,3$, to obtain the following partial derivatives:

$$\begin{aligned} \frac{\partial \mathcal{G}_T}{\partial v_k} &= \frac{1}{12} (2v_k + v_{k+1} + v_{k+2} - 4p) \|n_T\| \\ &+ \sum_{1 \leq i < j \leq 3} (v_i - p)(v_j - p)(v_{k+1} - v_{k+2}) \frac{n_T}{\|n_T\|} \end{aligned}$$

where indices k are taken modulo 3 and $n_T = (v_1 - v_2) \times (v_1 - v_3)$ the normal vector of triangle T .

For the computation of $\frac{\partial v_k}{\partial x_i}$, recall that X is the vector of the coordinates of vertices of S , and $T(v_1, v_2, v_3)$ is a triangle that is a subset of a triangular face $C(c_1, c_2, c_3)$ of S . With these notations, c_1, c_2 and c_3 are three triplets of coordinates of X . Let v be one of the vertices v_1, v_2, v_3 . It is clear that $\frac{\partial v}{\partial x_i} = 0$ if x_i is not present in c_1, c_2 or c_3 . In other words, the gradient computed on triangle T only contributes to the points of the face in which T belongs. In the opposite case, the partial derivative $\frac{\partial v}{\partial c_i}$ is non-zero, and three configurations appear: (see Figure A.15)

(1) v is one of the vertices c_i . Then $\frac{\partial v}{\partial c} = I_{3 \times 3}$ if $c = c_i$ and 0 otherwise.

(2) v is on an edge $[c_i, c_j]$. This means that v is on the intersection of the segment $[c_i, c_j]$ and an edge delimiting two Voronoi cells Ω_{p_1} and Ω_{p_2} . We have:

$$\begin{cases} \frac{\partial v}{\partial c_i} &= ew_i^\top + (1 - u)I_{3 \times 3} \\ \frac{\partial v}{\partial c_j} &= ew_j^\top + uI_{3 \times 3} \\ \frac{\partial v}{\partial c} &= 0 \text{ if } c \neq c_i, c_j \end{cases}$$

with:

$$\begin{cases} e &= c_j - c_i \\ n &= p_2 - p_1 \\ b &= n \cdot e \\ h &= n \cdot (p_1 + p_2)/2 \\ u &= (h - n \cdot c_2)/b \\ w_i &= \frac{1}{b^2}(n \cdot c_j - h)n \\ w_j &= \frac{1}{b^2}(h - n \cdot c_i)n \end{cases}$$

(3) v_k is in general position in the face (c_1, c_2, c_3) .

This means that v is the intersection of three Voronoi cells Ω_{p_1} , Ω_{p_2} and Ω_{p_3} . Therefore:

$$\begin{cases} \frac{\partial v}{\partial c_i} &= ew_i^\top, \quad i = 1, 2, 3 \\ \frac{\partial v}{\partial c} &= 0 \text{ if } c \neq c_1, c_2, c_3 \end{cases}$$

with:

$$\begin{cases} e &= (p_1 - p_2) \times (p_1 - p_3) \\ n &= (c_1 - c_2) \times (c_1 - c_3) \\ b &= n \cdot e \\ w_1 &= ((c_2 - c_3) \times (c_1 - v) + n)/b \\ w_2 &= ((c_3 - c_1) \times (c_1 - v))/b \\ w_3 &= ((c_1 - c_2) \times (c_1 - v))/b \end{cases}$$

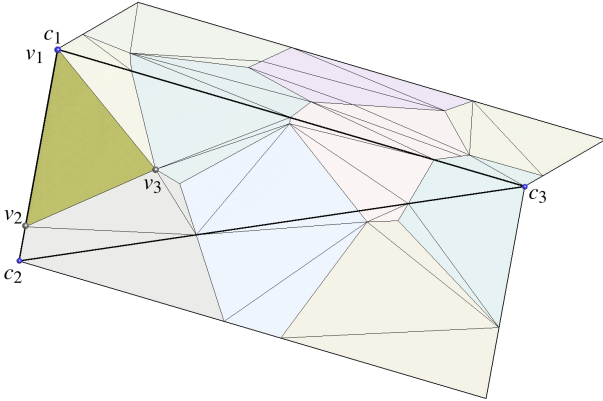


Figure A.15: Illustration of the three cases for one triangle. v_1 is in case (1), v_2 in case (2) and v_3 in case (3).

This allows us to conclude for the computation of $\nabla_X \mathcal{G}_T$. The sum to retrieve $\nabla_X \mathcal{G}$ is then straightforward.

Appendix B. Computation of $\frac{\partial E}{\partial \theta}$

Using the chain rule, we begin by writing:

$$\frac{\partial E}{\partial \theta} = \sum_{i=1}^n \frac{\partial E}{\partial S_i} \cdot \frac{\partial S_i}{\partial \theta} = \nabla_X E \cdot \frac{\partial S}{\partial \theta}$$

According to Equation (13):

$$S_i = R_z(\theta)X_i + T$$

the vector S only depends on θ through the application of the rotation matrix R . Differentiating according to a rotation can be done using its exponential form. Given a unit vector $u \in \mathbb{R}^3$, let

$$L_u = \begin{pmatrix} 0 & -z & y \\ z & 0 & -x \\ -y & x & 0 \end{pmatrix}$$

then, for any $\theta \in \mathbb{R}$, the matrix $R_u(\theta) = \exp(\theta L_u)$ is the rotation of angle θ along axis u . From this, we can deduce that:

$$\frac{\partial R_u(\theta)}{\partial \theta} = \frac{\partial \exp(\theta L_u)}{\partial \theta} = L_u \exp(\theta L_u) = L_u R_u(\theta)$$

In our case, we restrict ourselves to rotations along the axis Oz , as it is the only axis we needed in our applications, but this result is general for any axis Zu . Let L_z and $R_z(\theta)$ be the skew and rotation matrix of angle θ associated with the axis $Oz = (0, 0, 1)$. Notice that $L_z = R_z(\frac{\pi}{2})$. Combining this property with Equation (13), we get:

$$\frac{\partial S_i}{\partial \theta} = L_z R_z(\theta)X_i = L_z(S_i - T) = R_z(\frac{\pi}{2})(S_i - T)$$

This allows us to conclude for axis Oz . In order to add axes Oy and Ox , one has to apply the three rotations in fixed order and inserting L_x and L_y at the correct position in (15).

Appendix C. Computation of $\frac{\partial E}{\partial T}$

Let μ be an index x, y or z . Expressing the chain rule for the translation coordinate T_μ yields:

$$\frac{\partial E}{\partial T_\mu} = \sum_{i=1}^n \frac{\partial E}{\partial S_i} \cdot \frac{\partial S_i}{\partial T_\mu}$$

and according to Equation (13):

$$\begin{aligned} \frac{\partial S_i}{\partial T_x} &= (1, 0, 0) \\ \frac{\partial S_i}{\partial T_y} &= (0, 1, 0) \\ \frac{\partial S_i}{\partial T_z} &= (0, 0, 1) \end{aligned}$$

hence the result of Equation (16)