Robust normal vector estimation in 3D point clouds through iterative principal component analysis

Julia Sanchez, Florence Denis, David Coeurjolly, Florent Dupont, Laurent Trassoudaine, Paul Checchin

Robust and edge-aware normal vector estimation in 3D point clouds

Julia Sanchez[1], Florence Denis[1], David Coeurjolly[1], Florent Dupont[1], Laurent Trassoudaine[2], Paul Checchin[2]

[1] *Univ Lyon, LIRIS, UMR 5205 CNRS, Université Claude Bernard Lyon 1, 43, bd du 11 novembre 1918, 69622 Villeurbanne CEDEX, France; firstname.lastname@univ-lyon1.fr*
[2] *Institut Pascal, UMR 6602, Université Clermont Auvergne, CNRS, SIGMA Clermont, F-63000 Clermont-Ferrand, France; firstname.lastname@uca.fr*
julia.sanchez@univ-lyon1.fr

Abstract

This paper introduces a robust normal vector estimator for point cloud data. It can handle sharp features as well as smooth areas. Our method is based on the inclusion of a robust estimator into a Principal Component Analysis in the neighborhood of the studied point which can detect and reject outliers automatically during the estimation. A projection process ensures robustness against noise. Two automatic initializations are computed leading to independent optimizations making the algorithm robust to neighborhood anisotropy around sharp features. An evaluation has been carried out in which the algorithm is compared to state-of-the-art methods. The results show that it is more robust against a low and/or a non-uniform sampling, a high noise level and outliers. Moreover, our algorithm is fast relatively to existing methods handling sharp features. The code and data sets will be available online.

Keywords: Normal vector, Point cloud, Edge-aware, sharp features, M-estimator, Weighted PCA

1. Introduction

Recent developments of laser scanning devices lead to an increasing quantity of 3D point cloud data. These data can be used in various contexts such as indoor scenes modeling (Ochmann et al., 2016), navigation (Cadena et al., 2016), or default detection among many others. They can be acquired using

various sensors, from very accurate ones yielding millions of points to poorly accurate ones, yielding tens of thousands of points and introducing more noise. To handle all these data, multiple processes have been developed and many of them require surface normal estimation, such as denoising (Jones et al., 2003), registration (Bae and Lichti, 2008), segmentation (Rabbani et al., 2006), surface reconstruction (Berger et al., 2014) or visualization (Remondino, 2003).

In a point cloud, normal vectors are computed locally at a point using a defined neighborhood. Although many works have been proposed in this research field as detailed in the related works section, normal estimation still has to deal with various issues. Firstly, the accuracy on smooth surfaces could be improved, notably for noisy point clouds or non-uniformly sampled areas. Secondly, the computational efficiency should be bettered and a complete automatization should be reached to process very dense point clouds. Thirdly, the curvature discontinuities must be taken into account in the estimation.

This work presents a method for accurate local normal vector estimation in point clouds sampled on piecewise smooth surfaces with singularities. The estimation is based on a robust optimization integrating a weighted PCA to smoothly separate the neighbors into inliers and outliers. The anisotropy around sharp features is then handled through a double automatic initialization. The proposed method does not need any prior local curvature information and requires few parameters tuning. It is particularly adapted to analyze patterns or objects in structured scenes such as indoor environments scanned by LiDAR sensors but also gives reliable results on objects with curved surfaces. The proposed algorithm is more accurate than the existing methods which compute normals handling the issue of edges and can handle very dense point clouds (cf. Section 6.2). The method is validated on synthetic data, on CAD models, on LiDAR simulations and finally on real data acquired with three types of sensors to demonstrate its good performances for a wide range of applications (cf. Section 6). Moreover, a wide comparison is performed against 8 state-of-the-art methods on all kinds of data emphasizing the main advantages and drawbacks of all methods.

Section 2 briefly describes related methods. In Section 3, we recall some concepts which are used in our optimization. Section 4 contains the description of the proposed method. In Sections 5 and 6, the evaluation procedure is detailed and the results are provided. We finally conclude in Section 7.

2. Related works

To estimate a normal vector on a surface sampled by points, a local neighborhood must be selected (Rabbani et al., 2006; Weinmann et al., 2015), the direction of the normal must be determined from the neighborhood and finally the normal orientation can be deduced (Kim et al., 2018; Hoppe et al., 1992). This work focuses on the second step and our method seeks the direction of the normals related to a point cloud.

There exist four main categories of methods which aim at solving this problem: methods based on surface fitting, post-processes to correct an initial normals guess, methods based on the analysis of a set of raw normals and methods based on a Voronoi diagram construction.

The methods of the first category fit different models onto the local neighborhood. The simplest model to fit is a plane (Hoffman and K., 1987) and a classical method is to analyze the neighborhood covariance through a PCA to deduce the direction of the smallest local variation of points (Hoppe et al., 1992). It is equivalent to minimize the distance of the neighbors to the plane in a least squares sense (Mitra et al., 2003). More complex surfaces can be fitted such as spheres (Guennebaud and Gross, 2007), quadric surfaces (Yang and Lee, 1999), (Ouyang and Yung, 2005) or local 2-jets (Cazals and Pouget, 2005) to increase precision on curved areas.

Although the ℓ_2 norm is commonly used to fit surfaces because of its robustness to Gaussian noise and its simplicity of implementation, other methods have been proposed in order to better reject outliers. Fleishman et al. (2005), for instance, use an extended Least Median of Squares (LMS) to initialize their algorithm. Lipman et al. (2007) go further with the Locally Optimal Projection (LOP) to deal with outliers when performing consolidation. They solve an ℓ_1-median problem to project points onto the supporting surface uniformly. Another possibility is to introduce weights into the optimization. In the plane fitting case, Pauly et al. (2002) introduce a weighted version of PCA to give more impact to the closest points of the neighborhood. In a similar vein, the Moving Least Square (MLS) Projection is developed in Lipman et al. (2003) and Alexa et al. (2003). This method consists in fitting iteratively a reference plane to the neighborhood using the projection of the studied point onto it, to weight the neighbors. Then, a polynomial surface is fitted to the weighted neighbors. Belton (2008) also uses an iterative weighting scheme, however, the weights are actualized with a defined step value relatively to the local planes similarity.

Although these methods can reach high precision on smooth surfaces, they remain sensitive to noise and do not handle the singularities issue. Fleishman et al. (2005) handle sharp features separating the neighborhood into various surface pieces in the Robust Moving Least-Squares method (RMLS). Before using an MLS step, an extended LMS is performed and points are added to the initial model while their distance to the surface is lower than a threshold. This method needs a dense sampling and relies on the first LMS estimation which is not robust to noise nor sampling anisotropy. Mederos et al. (2003) extend the plane fitting introducing a robust estimator called M-estimator (Huber, 2011) calibrated to reject outliers in the least squares approach. We will detail the use of such an estimator in the next sections. They solve the M-estimation with a Newton method. However, this method is sensitive to local minima and the convergence strongly depends on the size of the kernel estimator. Moreover, it is not robust to anisotropy.

The second category of methods holds post-processes: starting from an initial normal field smoothed, these methods aims at smoothing the normal field while recovering the curvature discontinuities. Some authors propose filters to correct the normal field locally. Yagou et al. introduce three filters: the mean filter, the median filter and the alpha-trimming filter which is a compromise between the two previous filters to reduce the sensitivity to noise while handling curvature discontinuity (Yagou et al., 2002, 2003). Other extensions of these filters have been proposed to enhance such features (Yuzhong and E., 2004; Sun et al., 2007). Jones et al. (2003) and Fleishman et al. (2003) apply bilateral filters in order to smooth a mesh taking account of sharp features. Then, Jones et al. (2004) derive the normal estimation from the differentiation of the projection procedure. Zheng et al. (2011) directly apply the bilateral filter to the normals, the weights are computed relatively to the distance of the neighbors to the studied points and the likeliness of their normals. Zheng et al. (2018) and Öztireli et al. (2009) use this process to initialize respectively their denoising and reconstruction methods. This kind of method requires a reliable input normal field, a dense sampling and cannot reach high accuracy on curved surfaces. After a PCA estimation, Zhang et al. (2018) seek planes which minimize a cost function implying points pairs in the neighborhood, weighted depending on their normals likeliness. The authors also resort to M-estimators to reject outliers and the sampling anisotropy issue is handled through a new weighting scheme. This algorithm named Pair Consistency Voting (PCV) is efficient on low noise point clouds but it is slow compared to the actual existing methods.

Other normals correcting methods rely on a global optimization of the normal field which is expected to be piecewise smooth. In order to reflect the sparsity issue, Zheng et al. (2011) use a weighted ℓ_2 optimization, Avron et al. (2010) use the ℓ_1 norm and Sun et al. (2015) go further using the ℓ_0 norm. However, these global approaches tend to distort smooth curved parts of the point clouds.

In the context of mesh denoising, some authors propose to couple normal filters iteratively with vertices displacement in a procedure "two-steps, one-stage" (Sun et al., 2007). It consists, at each iteration, in updating alternatively the normal field and all vertices position so they fit the new normal field using the connection relations in the mesh (Taubin, 2001) (Yagou et al., 2002). In the point cloud consolidation context, Huang et al. (2013) use this strategy coupling the bilateral filter (Zheng et al., 2011) and the weighted LOP to move the points away from edges. The sharp areas are upsampled afterwards using an integration algorithm.

The third category of methods relies on the Voronoi diagram of the point cloud. The normal vector at a point can then be estimated from the geometry of its Voronoi cell: using polar Voronoi vertices (Amenta and Bern, 1999; Dey and Sun, 2006), a PCA (Alliez et al., 2007), or a more advanced convolutional approach (Mérigot et al., 2011). The later approach intrinsically detects sharp features but it still yields smoothed normals in the neighborhood of the sharp features and its robustness to noise is limited as it will be shown in the evaluation section. Che and Olsen (2018) introduce an edge detection with a local triangulation for the purpose of normal estimation improvement.

The fourth category is based on the estimation of a raw normals set from randomly selected triplets of points in the local neighborhood. Then the suitable normal can be deduced as being the average of the set (Gouraud, 1971), or the weighted average (Jin et al., 2005). Nevertheless, these methods cannot handle sharp features and they tend to smooth the edges. Boulch and Marlet (2012) build a histogram on raw normals and use a voting strategy inspired by the Hough transform to extract the proper normal. This method correctly enhances the discontinuities but it can lead to significant errors on curved surfaces. Moreover, its robustness to noise is limited and the handling of non-uniform samplings implies a great increase of computational time and requires new parameters. An extension of this method is proposed by Boulch and Marlet (2016), replacing the voting process by a neural network to take the final decision of the normal direction. Unfortunately, this method is not as accurate as the former one when the noise is limited and, for real

LiDAR data acquisitions, the network relies on the quality of the ground truth normals which are difficult to obtain.

Recently, Guerrero et al. (2018) have proposed a new method based on neural networks named PCP-net. They apply symmetric functions and rotations to the neighborhood to make it independent of the spatial order at the network input. However, this method is more adapted to objects with many curvature details and/or high levels of noise than simple piecewise smooth surfaces.

Contributions: we propose an algorithm to get an accurate estimation of the normal vectors in a noisy point cloud taking account of sharp features. Our method is particularly adapted to piecewise planar surfaces but it can handle smooth curved surfaces with a good accuracy. It performs the estimation at each point using only its neighbor's position and does not require any prior local surface information. It uses an iterative weighted process which estimates the normal while dividing the neighborhood between inliers and outliers. A projection process makes it robust to noise and two automatic initializations make it robust to anisotropy. Its low computation time makes it suitable for very dense point clouds processing.

3. Preliminaries

We rapidly recall two of the tools we will use in the normal estimation algorithm: the weighted PCA and the M-estimators.

3.1. Weighted PCA

We formulate the problem as follows: let $\mathbf{p}_0$ be a point of a point cloud and $\{\mathbf{p}_i\}_{i=1\ldots N}$ its N neighbors. We seek the normal $\mathbf{n}$ computed at $\mathbf{p}_0$ using its neighborhood.

The concept of PCA (Principal Component Analysis) for normal estimation (Hoppe et al., 1992; Mitra et al., 2003) is recalled in Appendix A in order to set up the notation.

We consider here a weighted version of the covariance matrix. More precisely, for any vector M of weights $\{m_i\}_{i=0\ldots N}$ associated to the neighbors, the weighted covariance matrix is defined as:

$$C_M := \frac{1}{\sum_i m_i} P^T \mathrm{diag}(M) P \,. \tag{1}$$

The estimated normal vector $\mathbf{n}^*$ at $\mathbf{p}_0$ is still the eigenvector associated with the smallest eigenvalue and this corresponds to the solution of the following least squares minimization:

$$\mathbf{n}^* = \underset{||\mathbf{n}||=1}{\operatorname{argmin}} \sum_i m_i(\mathbf{n} \cdot (\mathbf{p}_i - \mathbf{p}_0))^2 \,. \tag{2}$$

This process is equivalent to the classical weighted Principal Component Analysis (PCA) performed by Pauly et al. (2002) modifying the PCA reference point which is the neighborhood centroid by $\mathbf{p}_0$. This modification leads to better results when introducing a robust estimator (cf. Section 4.1).

3.2. M-estimators

The M-estimators are robust statistic tools which allow to fit a model onto points while rejecting outliers (Huber, 2011). Let θ be a set of parameters we seek and $\{e_i\}_{i=1...N}$ be output errors of a system, relative to θ (e_i will be a fitting residual in our method). Then, the M-estimation is a method which outputs the parameters θ^* seeking the minimum-likelihood of the relative errors such that:

$$\theta^* := \underset{\theta}{\operatorname{argmin}} \sum_i \rho(e_i(\theta)) \,, \tag{3}$$

where the function $\rho(x)$ is a user-specified robust filtering kernel, so called the M-estimator. This minimization is usually achieved through an Iteratively Reweighted Least Squares IRLS procedure (William, 1979). At each iteration k, two steps are performed successively: first, the weights are set using Eq. (4) from θ^{k-1}, then, the optimization problem Eq. (5) is solved to update the current solution θ^k:

$$w_i{}^k := \frac{1}{e_i(\theta^{k-1})} \frac{\partial \rho}{\partial e_i}(e_i(\theta^{k-1})) \tag{4}$$

$$\theta^k := \underset{\theta}{\operatorname{argmin}} \sum_i w_i{}^k \cdot e_i(\theta)^2. \tag{5}$$

The reader is invited to refer to (William, 1979) for more information. Then, quadratic solvers (such as Gauss-Newton or Levenberg-Marquardt) can be applied and the difficulty of the optimization problem in Eq. (5) mostly depends on the regularity of e_i with respect to θ.

4. Robust piecewise smooth normal vector estimation

Our algorithm uses a fitting procedure with an M-estimator in the neighborhood of the studied point which yields a normal direction and a separation of the neighbors into inliers and outliers. We explain how the optimization of such an estimator leads to the integration of a weighted PCA to an iterative process in Section 4.1. Then, a detailed description of the algorithm is given in Section 4.2.

4.1. Iterative weighted PCA

Our method aims at separating the neighborhood between inliers, which represent the neighbors lying on the same smooth surface piece as $\mathbf{p}_0$ below some noise level, and outliers which can be noisy points or points lying on another smooth surface piece. Therefore, an M-estimation is used to characterize such outliers and obtain refined normal vector estimations such as in the work of Mederos et al. (2003). In that context, let us suppose that we have an estimate of the normal vector at $\mathbf{p}_0$, denoted $\mathbf{n}$. Thus, the residual relative to the neighbor $\mathbf{p}_i$ is given by $r_i(\mathbf{n}) := \mathbf{n} \cdot (\mathbf{p}_i - \mathbf{p}_0)$.

The choice of the estimator kernel is critical to insure the good convergence of the optimization. We chose the following M-estimator:

$$\rho(r_i(\mathbf{n})) := \frac{\mu r_i(\mathbf{n})^2}{2(\mu + r_i(\mathbf{n})^2)}\,, \tag{6}$$

which is a scaled version of the Geman-McClure estimator (He et al., 2014, Chapter 5) parameterized by a scalar μ. Contrary to the work of Mederos et al. (2003) in which the authors resort to a Newton method to solve the optimization, an IRLS minimization is performed. Each weight w_i is computed as:

$$w_i(\mathbf{n}) := \left(\frac{\mu}{\mu + r_i(\mathbf{n})^2}\right)^2\,, \tag{7}$$

and then, the following expression is minimized considering constant weights:

$$\sum_i w_i \cdot r_i(\mathbf{n})^2\,. \tag{8}$$

Eq. (8) is exactly Eq. (2), hence, it can be solved by PCA (cf. Section 3.1). Starting from a well selected initial direction $\mathbf{n}^{init}$ at $\mathbf{p}_0$, two steps are alter-

nated: the weights are updated relatively to the current estimated normal and the new estimation is computed through a weighted PCA. The weighting function Eq. (7) is represented in Figure 1 for three different values of the parameter μ. It can be observed that if $r_i^2 \ll \mu$, w_i is close to 1 and $\mathbf{p}_i$ strongly impacts the normal estimation. On the contrary, if $r_i^2 \gg \mu$, the weight w_i is low and $\mathbf{p}_i$ is excluded from the computation. If μ is fixed for all iterations to too high a value, the normal estimation will remain far from the real normal direction. If μ is fixed to too low a value, local minima may appear and the algorithm may not converge to the optimal solution. Therefore, the parameter μ in Eq. (7) is decreased at each iteration so the weighting function is calibrated on the inlier residuals which decrease too.

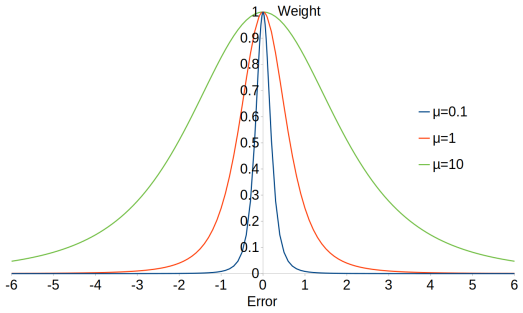


Figure 1: Weighting function depending on the error. Representation for three values of the parameter μ.

4.2. Algorithm description

In this section, we explain how the optimization procedure is divided into two successive steps: a rough estimation of $\mathbf{n}$ is obtained and a refinement is performed. Then, we explain how to handle the problem of face confusion in a sharp edge neighborhood.

4.2.1. Elementary steps

In these steps, the normal vector $\mathbf{n}$ is iteratively estimated using a weighted PCA and actualizing the weights at each iteration with Eq. (7). The initial estimated vector normal is computed with a classical PCA (cf. Figure 2a) and is called $\mathbf{n}_1^{init}$ (see the next section for the rationale behind the "1" subscript). In the first step, only the direction of the normal vector is optimized. In the second step, the position of the PCA reference point is also modified to refine the normal vector estimation in the presence of noise.

- **Rough estimation:** μ is initialized as the square of the maximum residual computed in the neighborhood: $\mu_{init} = (\max_i(r_i))^2$. At each iteration, μ is decreased to scale the weights onto the residual values. If the decrease is too fast, the algorithm can fall into a local minimum. If the decrease is too slow, the algorithm might be longer than necessary. We have used a scaling factor of 1.01 per iteration in all our experiments. The effect of this scaling factor is detailed in Figure B.22. The iterations are performed until μ reaches a threshold μ_{lim} which depends on the curvature and the noise in the model. The tuning of this parameter is discussed in Section 4.2.3. The number of iterations is then: $\frac{\ln(\mu_{init}\,/\,\mu_{lim})}{\ln(1.01)}$. The resulting estimated normal is shown in Figure 2b.

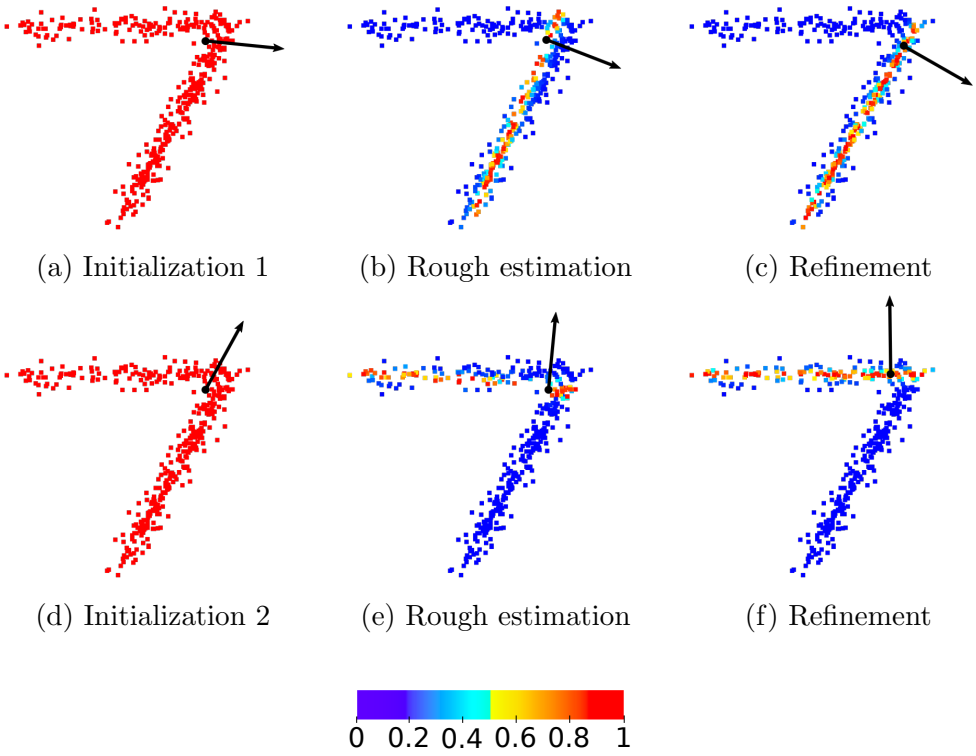
- **Refinement:** this second part is a refinement step to deal with the noise on the reference point used to perform PCA. So far, the normal vector is given using a weighted PCA with a fixed reference point $\mathbf{p}_0$ (cf. Eq. (2)). We can further improve the stability of the results by updating the reference position in an iterative framework: a candidate normal vector $\mathbf{n}$ is estimated, we shift the reference point along $\mathbf{n}$ by $\sum_i w_i r_i(\mathbf{n}) / \sum_i w_i$, and we iterate. The resulting estimated normal is denoted $\mathbf{n}_1$ and an example is given in Figure 2c. The final PCA reference position is denoted $\mathbf{p}_{01}$. Note that $\mathbf{p}_{01}$ is only used to refine the estimation of the normal vector and we do not actually move the input point $\mathbf{p}_0$ (mandatory for some LiDAR applications). The effect of the refinement step is evaluated in Appendix B, Figure B.18.

4.2.2. Anisotropy challenge: second initialization

If $\mathbf{p}_0$ lies in the neighborhood of an edge, the resulting normal of the previous algorithm might be attracted to the side where more points are lying and/or where the surface piece has less curvature. For these reasons, a second initial direction is computed and a second optimization is performed to get another estimated normal vector (cf. bottom row of Figure 2). In order to minimize the chances to retrieve the first alternative normal, the second initial normal must be 90° away from $\mathbf{n}_1$ and from the edge direction ν_1. The edge direction is derived from the cross product between $\mathbf{n}_1^{init}$ and $\mathbf{n}_1$, making the hypothesis that the normal vector was optimized following a plane perpendicular to the edge due to its symmetry. The second initial normal, given by

$$\mathbf{n}_2^{init} := \mathbf{n}_1 \wedge \nu_1 \,,$$

is shown in Figure 2d. Then, the two elementary steps are performed leading respectively to the results in Figure 2e and Figure 2f.



(a) Initialization 1 (b) Rough estimation (c) Refinement

(d) Initialization 2 (e) Rough estimation (f) Refinement

Figure 2: Evolution of the normal and the weights in a sharp feature with non-uniform sampling. The color scale represents the neighbors' weights from 0 (blue) to 1 (red). (a), (b) and (c) correspond to the computation steps leading to $\mathbf{n}_1$, (d), (e), and (f) correspond to the computation steps leading to $\mathbf{n}_2$. (a) and (d) correspond to the two initialization steps before weighting the process. (b) and (e) correspond to the rough estimation results. (c) and (f) correspond to the refinement results.

For the second optimization, μ must be initialized to a smaller value than in the first one, in order to get a different estimation from $\mathbf{n}_1$. In all our experiments, we chose to initialize μ to the value of the 33^{th} percentile of the squared residuals. We call this second estimated normal $\mathbf{n}_2$ and the new PCA reference position $\mathbf{p}_{02}$. After the second optimization, a choice must be made between the two estimations $\mathbf{n}_1$ and $\mathbf{n}_2$. Firstly, the estimated normals

are oriented to the exterior of the curvature: both vectors $\mathbf{n}_1$ and $\mathbf{n}_2$ must satisfy the condition Eq. (9).

$$\sum_i \left(\mathbf{n}_k \cdot (\mathbf{p}_i - \mathbf{p}_0)\right) < 0, \;\; k \in \{1, 2\}\,. \tag{9}$$

Secondly, the values $\{\mathbf{n}_k \cdot (\mathbf{p}_{0k} - \mathbf{p}_0)\}_{k=1,2}$ are computed and the minimum value corresponds to the selected normal. The effect of this double initialization is evaluated in Appendix B.

4.2.3. Convergence properties and parameters setting

Our algorithm stops when μ reaches the prescribed value μ_{lim}. We recall that μ represents the bandwidth of the estimator, *i.e.* it defines inliers in the neighborhood. If μ_{lim} is too high, the algorithm might not converge to the optimal normal. If μ_{lim} is too low, the algorithm can diverge. The error is bounded by a classical PCA estimation error when increasing μ_{lim}. An optimal value for μ_{lim} can be sought as it only depends on the noise level and the curvature of the model. To help to derive this value for all models, the maximum acceptable residual r_{max} can be sought in the neighborhood. Then, μ_{lim} can be computed as $\mu_{lim} = {r_{max}}^2$, so the weight is 0.5 for neighbors which have a residual equal to r_{max}, it tends to 0 for neighbors which have an residual much greater than r_{max} and to 1 for neighbors which have a very low residual. If the studied model can be represented by pieces of planes, r_{max} only depends on the Gaussian noise level of the model represented by the standard deviation σ. For other models, the effect of the curvature has to be taken into account. Indeed, as shown on the 2D example of smooth curved surface in Figure 3, all neighbors need to be taken into account in the normal vector estimation to preserve the symmetry of the inliers while converging. Hence, for all models containing smooth curvature, a minimal curvature radius R_{min} must be defined by the user. If the points have no noise, r_{max} can be defined as the residual of the farthest neighbor if the neighborhood were on a surface of curvature radius R_{min}. Let X_{max} be this residual, it is represented in Figure 3.

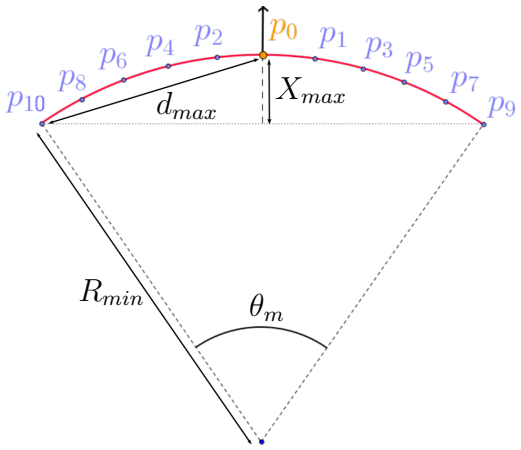


Figure 3: Representation of a curved surface projected in 2D (red). X_{max} is the maximum acceptable residual for neighbors to be considered as inliers. The normal is computed in $\mathbf{p}_0$ using neighbors $\{\mathbf{p}_1 \dots \mathbf{p}_{10}\}$. If some neighbors are not taken into account, the orientation of the estimated normal vector can be corrupted.

It is linked with R_{min} and the distance d_{max} of the farthest neighbor from $\mathbf{p}_0$ with the following relation: $X_{max} = \frac{{d_{max}}^2}{2R_{min}}$. Therefore, we integrate the curvature and the noise information using: $r_{max} = X_{max} + 0.5\sigma'$, with $\sigma' = \frac{\sigma}{\sqrt{3}}$ being the noise standard deviation in one direction of the space. Therefore, R_{min} and σ are the parameters of our algorithm. In practice, σ must be tuned relatively to the LiDAR sensor used and a high R_{min} is suitable for piecewise planar objects. In Appendix B, the effect of μ_{lim} on the results is evaluated and this corroborates the practical choice.

4.2.4. Pre-selection of input points

In order to make the algorithm faster, the iterative optimization can be only applied for points detected on edges. Indeed, after the first classical PCA initialization, we compute the standard deviation of the neighbors' residuals relatively to the plane defined by the normal in the centroid location. If it is lower than a threshold τ, we suppose that the neighborhood does not contain any curvature discontinuity and the iterations can be avoided. τ is defined as the theoretical standard deviation of all points of the continuous underlying shape with a specific curvature radius R_{min}. This theoretical

standard deviation can then be estimated by integration as:

$$\tau := \sqrt{\frac{R_{min}^2}{2}\left(1 + \frac{\sin(\theta_m)}{\theta_m}\right) - \overline{p}^2}\,. \tag{10}$$

With θ_m the aperture angle defined as:

$$\theta_m := 2\arccos\left(\frac{R_{min} - X_{max}}{R_{min}}\right)\,. \tag{11}$$

And $\overline{p}$ being the theoretical mean defined as:

$$\overline{p} := 2R_{min}\frac{\sin(\frac{\theta_m}{2})}{\theta_m}\,. \tag{12}$$

The higher R_{min} is fixed, the more neighbors will be processed by the optimization step. This pre-selection is evaluated in Appendix B.

5. Evaluation procedure

To evaluate our algorithm, we compare it with 8 other methods: PCA (Hoppe et al., 1992), 2-jets (Cazals and Pouget, 2005), VCM (Mérigot et al., 2011), the bilateral filter extension (Huang et al., 2013), the Hough based method (Boulch and Marlet, 2012), its neural network extension (Boulch and Marlet, 2016), PCV method (Zhang et al., 2018) and PCP-net neural network (Guerrero et al., 2018). These methods represent the categories described in Section 2, namely: surface fitting algorithms, Voronoi cells based methods, post-processes, random selection based methods and neural networks. All methods were evaluated with the same number of neighbors for each model excepting PCP-net which is trained with a fixed neighborhood radius. PCA and 2-jets only need this parameter. VCM is only evaluated on synthetic models. It takes two radii as parameters. The offset radius is set to 5 times the mean distance between nearest neighbors. The convolution radius r is computed so it corresponds to the number of neighbors required in the other methods. An approximation is made: the area of the neighborhood named A_{neigh} is linked with the total area of the object named A_{tot} as follows:

$$A_{neigh} = \pi r^2 = \frac{N_{neigh}}{N_{tot}} A_{tot}\,,$$

with N_{neigh} and N_{tot} being respectively the number of points lying in the neighborhood and in the total cloud. Hence, r can be deduced from this equation when the sampling is uniform. The bilateral filter extended with the LOP process is applied after a PCA initialization. Its sharpness parameter is set to 25° and 5 iterations are performed. The Hough based method is used with a maximum pair selection number of 20 000 and 10 bins compose the histogram. The clustering method is used to deal with the discretization issues. The CNN extension of this method is used with the pretrained network supplied by the authors. We use the Computational Geometry Algorithms Library (CGAL) implementations to test PCA, 2-jets, VCM methods and the bilateral filter. We use the online code supplied by the authors to test the Hough based method, its deep learning extension, PCP-net and the code provided by the authors to test PCV with default parameters. Our algorithm is evaluated modifying two parameters depending on the tested data set: the estimated standard deviation of the noise and the minimum curvature radius tolerated (cf. Section 4.2.3). If not specified, the selection of input points is performed. It is to note that the bilateral filter internally moves the point $\mathbf{p}_0$. However, all point clouds are shown with their initial position. As evaluation tools, we use the mean angle error, the Root Mean Squared (RMS) angle error which are well-known statistical tools and the thresholded RMS (RMSτ) as introduced by Boulch and Marlet (2012) and used by Zhang et al. (2018) with the same threshold, *i.e.* 10°. This last value is an RMS computation but the angles greater than the threshold are set to 90° in the averaging. It penalizes the smoothing effect of the algorithms.

In the following, first we evaluate our algorithm on synthetic basic models in order to study the relevant features of each algorithm. Then, the method is evaluated on point clouds extracted from CAD meshes of objects which contain curved parts and sharp features. A LiDAR scanner is also simulated in indoor scenes represented by meshes in order to get point clouds close to real acquisitions with high levels of sampling anisotropy while preserving a reliable ground truth. Finally, real data provided by LiDAR scanners are used to validate the proposed method visually. All values for our algorithm's parameters are shown in Table 1.

Table 1: Setting of our algorithm's parameter values for all data sets.

	Estimated s.d. of noise (cm)	Min. curvature radius tolerated (cm)
Planes model	$[0, 2]$	∞
Cylinders model	$[0, 1]$	10
CAD data set	$[0, 0.2]$	8
Simulated LiDAR data	0.1	∞
Real LiDAR data	1	∞

6. Results

In this section, we first evaluate the methods in terms of accuracy and then in terms of computational efficiency.

6.1. Accuracy and robustness

6.1.1. Synthetic models

A first basic synthetic model is used for evaluation. It is an intersection of two planes containing 15 000 points randomly sampled. Each of the planes is 1 m long and 0.5 m large. The edge is 1 m long. The angle between the two planes is 90°. Gaussian noise is added to the points with a standard deviation of 108% of the mean distance between nearest neighbors. The neighborhood is defined by 300 points. First, the different categories of methods (described in Section 2) are tested to enhance visually their features and a part of this model is extracted for visibility (cf. Figure 4). The corresponding normals are presented in Figure 5. Surface fitting techniques (PCA and 2-jets) cannot handle sharp features and smooth the edge. Voronoi cells based method (VCM) yields an improvement but the normals stay far from their ground truth orientations. The post-processes such as the bilateral filter tested here, usually correct the normals whose neighborhood contains the edge but tend to corrupt the accuracy of the other normals. The random trials based methods (Hough based method here) seem to enhance correctly the discontinuity but

some points are allocated to the incorrect smooth surface piece. With a neural network (Hough-CNN method here), some normals remain smoothed near the edge. Finally, our method can handle this kind of situation with a good accuracy.

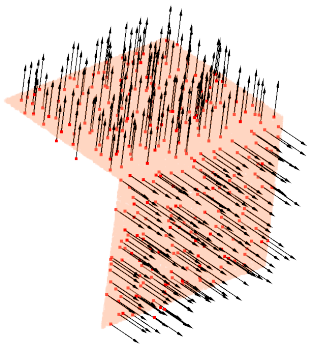


Figure 4: Planes intersection with ground truth normals used as synthetic model to evaluate our algorithm.

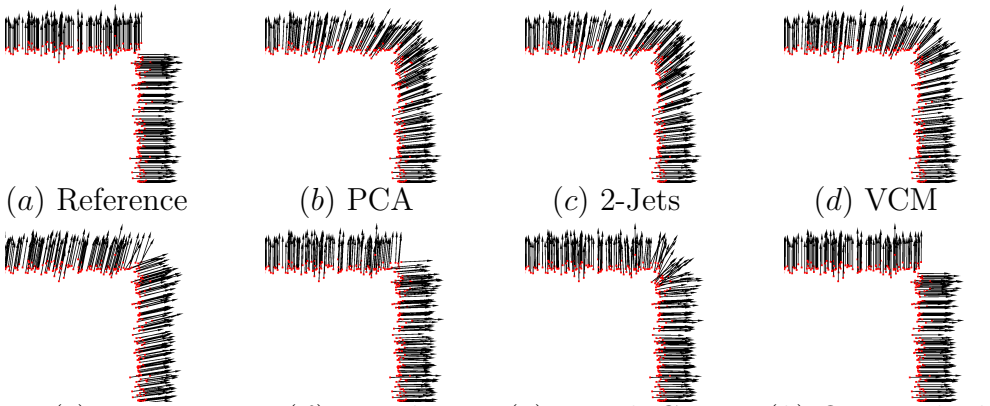
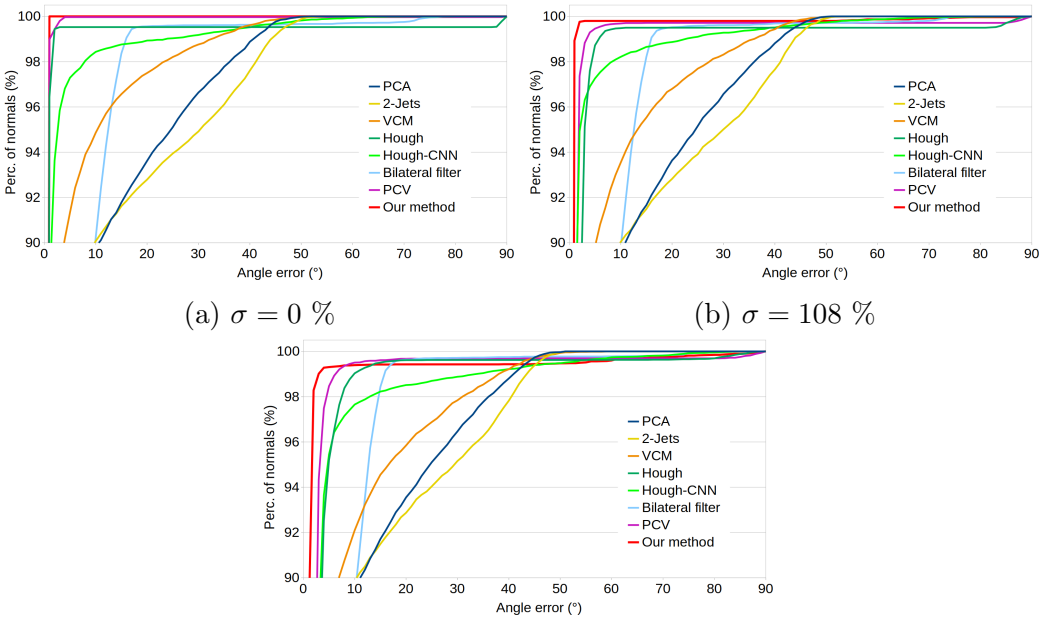


Figure 5: 2D projection of the normal estimations on the intersection of two planes.

Then, the cumulative histograms of normals relatively to the angular error for three different levels of Gaussian noise are represented in Figure 6 and the graph of Figure 7 shows more statistic results for 16 Gaussian noise levels for the same model. As all algorithms have good performances on planes and 84% of the points have a neighborhood exclusively on a plane, the interesting parts of the graphs in Figure 6 lie in the interval [90%,100%]. PCA, 2-jets and VCM methods lead to errors always lower than 50° but the number of normals having an error greater than 5° is high because of the smoothing effect near the edge. The Hough based method well handles the edge of the model but it has a significant rate of inverted normals, *i.e.*, belonging to the other surface piece lying in their neighborhood. This leads to various points having a high angle error (> 80°). The extension does not separate as clearly the normals on edges as the classical method and leads to more angular error



(a) $\sigma = 0$ %

(b) $\sigma = 108$ %

(c) $\sigma = 217$ %

Figure 6: Cumulative histograms of normals evaluated on the intersection of two planes (1×0.5 m) containing 15 000 points randomly sampled showing the percentage of normals having an error lower than the x-axis value. 3 different Gaussian noise levels are added to the model. The best results correspond to the curves on the top left. The neighborhood is defined by 300 points. σ corresponds to the standard deviation of the Gaussian noise evaluated in % of the mean distance between the nearest neighbors.

in the noise-free case. However, it is more robust against a certain level of noise on smooth areas ($< 200\%$ of the mean distance between nearest neighbors). The bilateral filter corrects the normals on sharp features and is robust against Gaussian noise. However, it leads to numerous normals having a low angle error ($< 10°$) even in the noise-free case. Regarding PCV algorithm, the results are reliable on this model but its robustness to noise is not as good as our algorithm's. Our method has the best behavior in this experiment.

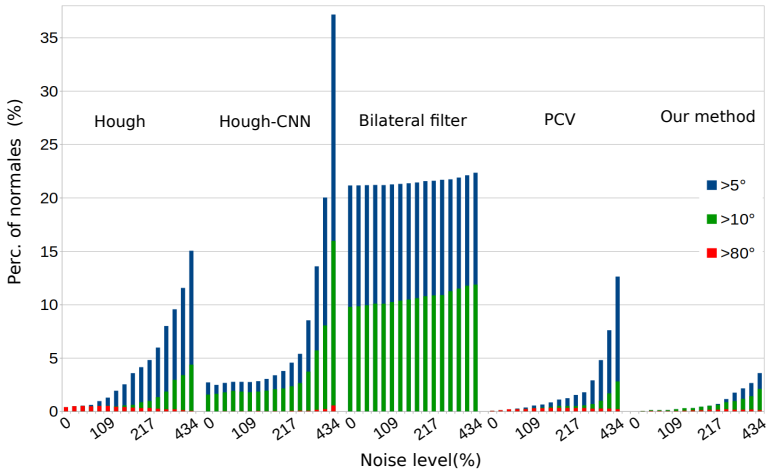


Figure 7: Percentage of resulting normals above three angular error thresholds (5°, 10°, 80°) on the intersection of two planes (1×0.5 m) containing 15 000 points randomly sampled, for 16 Gaussian noise levels (from 0% to 400% of the mean distance between the nearest neighbors). The neighborhood is defined by 300 points.

To compare the results of all algorithms on planar piecewise models and on structures with smooth curvature, we introduce a second synthetic model: a pipe bend containing 100 000 points randomly sampled. Each bend side is 0.2 m long with a radius of 0.1 m and the pipes intersect at 90°. The neighborhood is defined by 200 points. The angular errors are represented for this model in Figure 8. It confirms the features of each algorithm observed above and it allows to see that VCM, the bilateral filter, Hough and Hough-CNN based methods lead to unwanted errors on the smooth parts of the pipes. PCV separates correctly the smooth parts of the surface but its accuracy remains lower than our method's. Our algorithm reaches a good accuracy on the smooth parts as well as on the neighborhood of the curvature discontinuity. Moreover, it can handle a small angle between two smooth parts as

it can be seen on the side of the pipes bend.

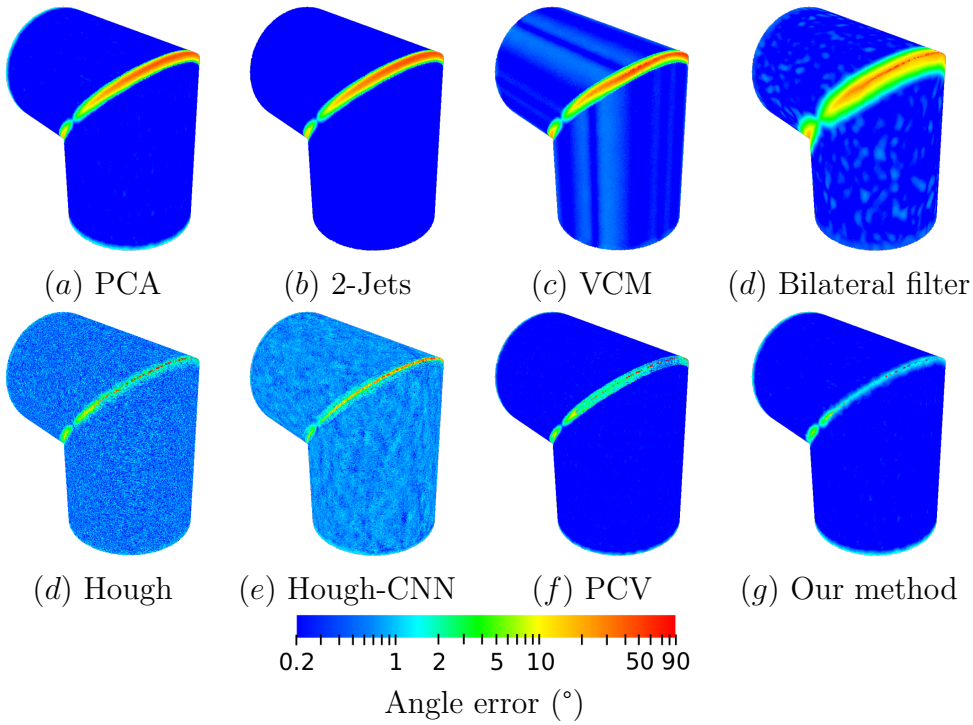


Figure 8: Angular error evaluated on the pipe bend. The neighborhood is defined by 200 points.

In Figures 9 and 10, the robustness to Gaussian noise is tested on both models. Our algorithm yields a mean angle error lower than the other for all noise levels for both the planes model and the pipes model and the accuracy decreases slower than with the Hough based method or PCV when the noise level increases. The neural network extension has a bad behavior for high levels of noise and does not reach the accuracy of the basic Hough based method in the noise-free cases. PCV accuracy decreases faster when processing curved surfaces than planes relatively to the noise level. It is to note that the pre-selection of points leads to a small decrease of accuracy when the noise level is greater than 100% of the mean distance between nearest neighbors.

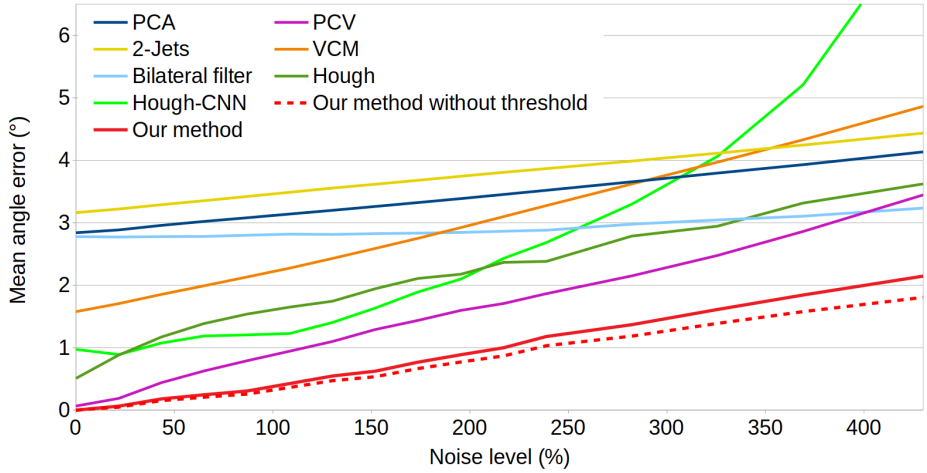


Figure 9: Mean angle error evaluated on the intersection of two planes (1×0.5 m) randomly sampled by 15 000 points, for increasing Gaussian noise levels (the noise level is expressed as a percentage of the standard deviation relatively to the mean distance between the nearest neighbors). The neighborhood is defined by 300 points. Our algorithm is evaluated pre-selecting or not the input points for the optimization.

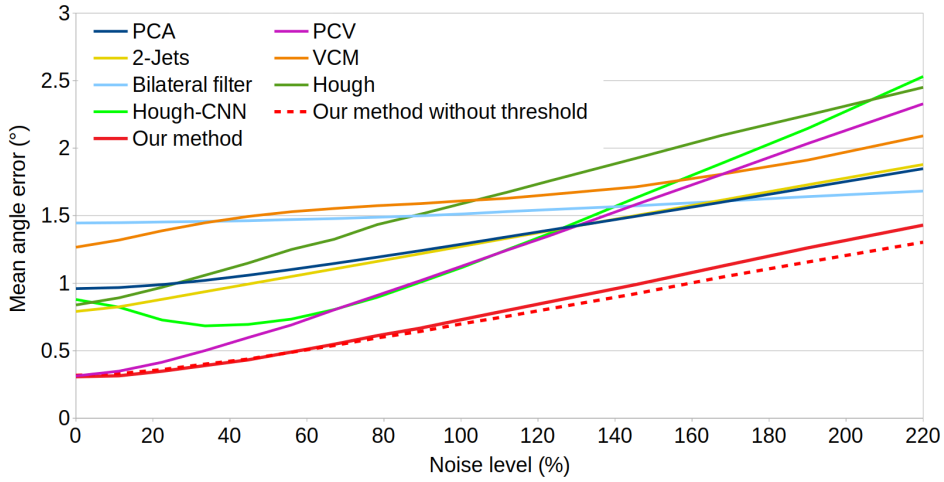


Figure 10: Mean angle error evaluated on the pipe bend (0.2 m long and 0.1 m of radius) randomly sampled by 100 000 points, for increasing Gaussian noise levels (noise level expressed as a percentage of the standard deviation relatively to the mean distance between the nearest neighbors). The neighborhood is defined by 200 points. Our algorithm is evaluated pre-selecting or not the input points for the optimization.

Figure 11 allows to study the behavior of the algorithms when the neighborhood is increased. PCA, 2-jets and the bilateral filter reach their best results for a particular neighborhood size and then lead to an increasing error. On the contrary, the results of the other methods are improved when the neighborhood is increased. The Hough-CNN based method is very sensitive to the reduction of neighbors selected as it can be seen on Figure 11 with a mean angle error multiplied by three when the neighborhood is reduced from 400 points to 100 points.

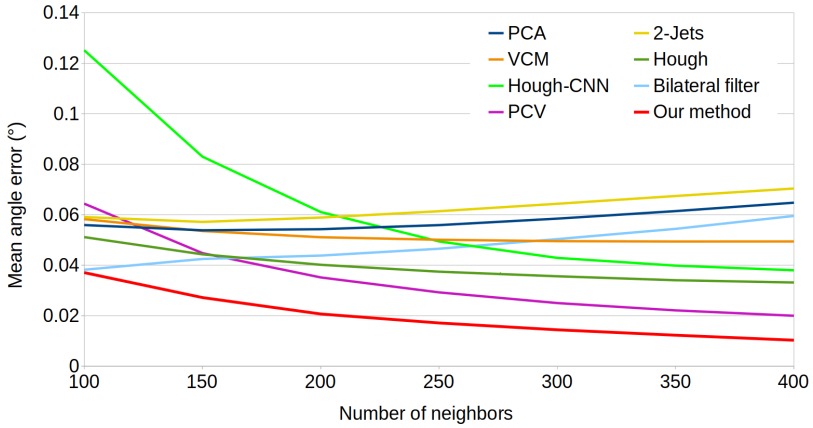


Figure 11: Mean angle error evaluated when varying the neighborhood size on the intersection of two planes (1×0.5 m) randomly sampled by 15 000 points. An average of the mean angle error is performed for 16 Gaussian noise levels from 0% to 400% of the mean distance between the nearest neighbors.

Note that PCP-net is not included in this evaluation section because of its poor performances on synthetic smooth models (cf. Appendix C).

6.1.2. CAD meshes

Then, the methods are evaluated on 14 CAD meshes mixing planes, smooth curved areas and including discontinuities in the curvature. An overview of these data is given in Figure 12. Each mesh is scaled for the diagonal length of the bounding box to be 1 and upsampled using the midpoint method until all the edges have a length smaller than 1% of this diagonal. The face centroids and their normals are extracted. At the end, the objects contain between 54 000 and 493 000 points. Gaussian noise is added to each mesh with a standard deviation within $[0\%, 0.2\%]$ of the diagonal length with 9 values regularly spaced. For each CAD model, the neighborhood is defined

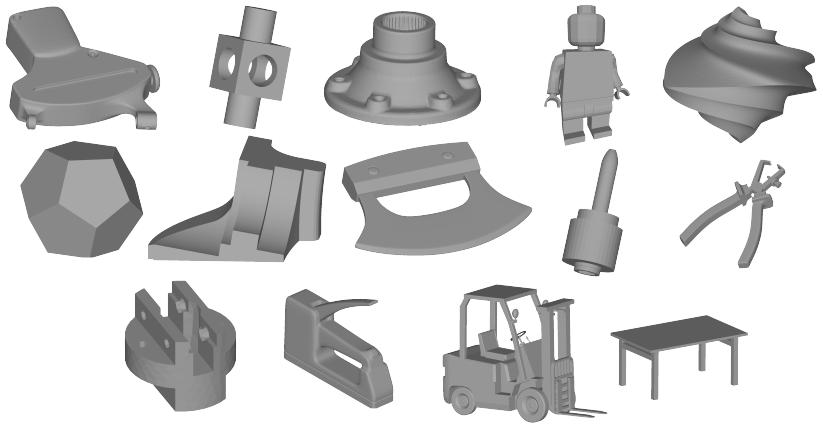
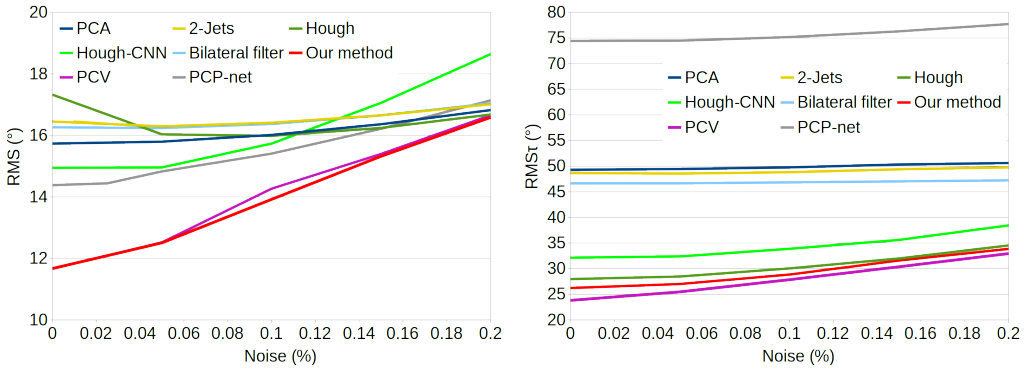


Figure 12: CAD meshes used for evaluation. The meshes are normalized so the diagonal of their bounding box is 1. They contain between 54 000 and 493 000 points.



(a) RMS angular error

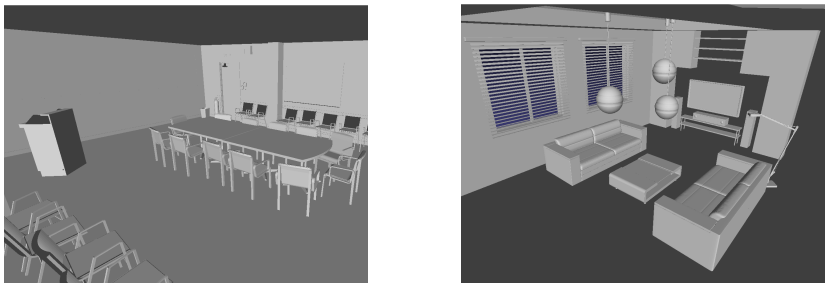
(b) RMSτ angular error

Figure 13: Angular error depending on the Gaussian noise level on a data set of 14 CAD objects containing between 54 000 and 493 000 points. The noise level is expressed as a percentage of the standard deviation relatively to the bounding box diagonal length.

by 200 points. The results are presented in Figure 13. The curvature radius parameter of our algorithm is set to 8% of the diagonal length in all cases. Our algorithm reaches the minimal RMS angular error on this data set. PCV obtains similar results to ours but its computation time is up to 25 times higher (cf. Section 6.2). Due to its poor accuracy on smooth regions, PCP-net obtains the worst RMSτ result.

6.1.3. LiDAR simulations

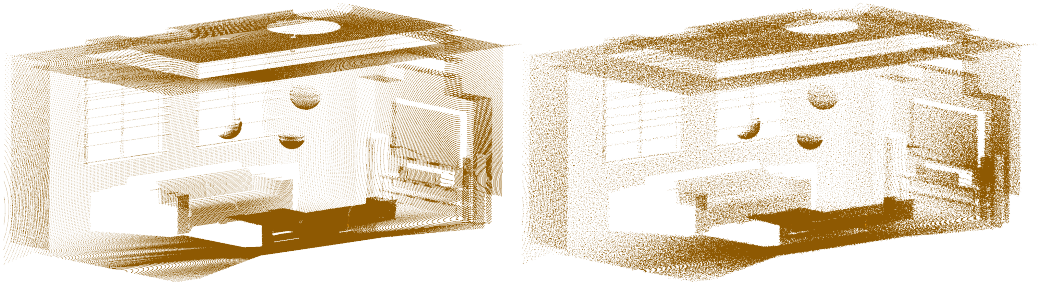
To evaluate our algorithm in conditions close to reality, a LiDAR scanner is simulated in 4 CAD environments representing indoor scenes which are shown in Figure 14. The point clouds have high levels of anisotropy due to the internal working of a LiDAR device (cf. Figure 15a). On each mesh, we sampled 320 000 points adding a Gaussian noise of 0.1% of the bounding box diagonal, as it is a common noise in real data (cf. Figure 15b). As the models



(a) Conference room (b) Living room 1

(c) Living room 2 (d) House

Figure 14: Meshes of indoor scenes. A LiDAR device is simulated for evaluation.



(a) No noisy version (b) Noisy version

Figure 15: Point cloud obtained after a LiDAR scanner simulation in a CAD mesh model of an indoor scene called Living room 1.

are mostly planar, the curvature radius parameter of our algorithm is set to infinity. The results are shown in Figure 16. It appears that PCV and Hough-CNN algorithms are unstable on noisy inputs with non-uniform samplings. PCA and 2-jets yield reliable results but the angular error induced by the normals smoothing is still high. Hough based method and the bilateral filter limit this smoothing effect, but our algorithm reaches the highest accuracy on this data set.

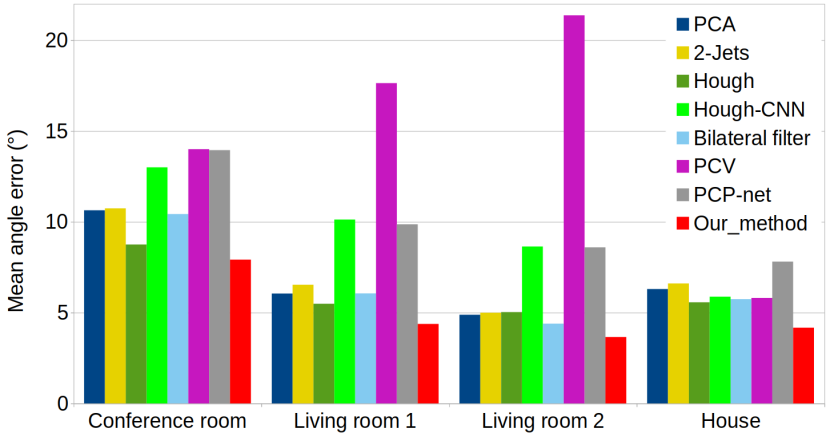


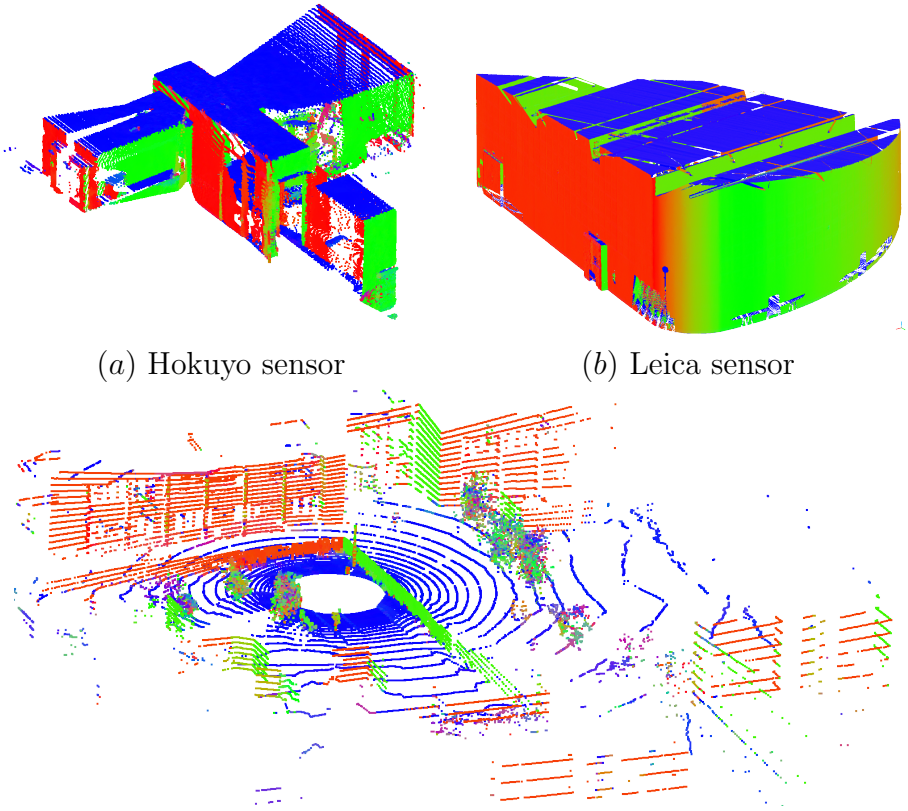
Figure 16: Mean angle error computed on a LiDAR point cloud simulated in indoor scenes represented by meshes. Gaussian noise is added with a standard deviation of 0.1% of the bounding box diagonal. The neighborhood is defined by 200 points.

6.1.4. Real LiDAR point clouds

To evaluate our algorithm on real data containing more points with various levels of noise and sensor artifacts, we tested three different point clouds whose features are summarized in Table 2. The first one is extracted from the data set "apartment", available on the Autonomous System Lab website (Pomerleau et al., 2012). The second and the third ones are extracted from the data sets "Patio" and "PAVIN" respectively, available online (Sanchez et al., 2017). All point clouds were acquired in structured scenes. The local neighborhoods are defined by 100 points for all real data acquisitions, the curvature radius parameter of our algorithm is set to infinity and the noise estimation parameter is set to 1 cm for the Hokuyo and the Leica point clouds, and to 2 cm for the Velodyne point cloud. The results of our algorithm are shown in Figure 17. These results show that it is robust to a

Data set	Sensor	Accuracy	Points	Resolution
Apartment	Hokuyo UTM-30LX	±3 cm	370 000	6 mm
Patio	Leica P20	±3 mm	3 210 000	3 mm
PAVIN	Velodyne HDL-32E	±2 cm	60 000	8 cm

Table 2: Features of the three point clouds extracted from LiDAR data sets.



(a) Hokuyo sensor (b) Leica sensor

(c) Velodyne sensor

Figure 17: Normals estimated by our algorithm on real data acquired with three different sensors in structured scenes. The RGB components correspond to the projection value of the normals on the x, y, and z axis respectively.

low and anisotropic sampling (particularly notable for the Hokuyo and Velodyne sensors) and that it recovers curved parts of the structures with a good accuracy. Moreover, it can detect narrow planes such as the door boundaries of the room scanned by the Leica sensor in Figure 17.

6.2. Computation time

For the model of pipes bend, the average for each method of the computation times for different noise levels are shown in Table 3 for estimations using respectively 50, 100, 200, 300 and 400 neighbors (processor: Intel i5–7440HQ, 2.80 GHz). Each algorithm is run in a single thread. The Gaussian noise level is increased from 0% to 200% of the mean distance between nearest neighbors with 16 values. Note that the algorithms which cannot handle sharp features are always faster than the others. Our algorithm is faster than the Hough based methods on these kinds of data. The bilateral filter is faster when the neighborhood is small but gets slower when the neighborhood contains more than 100 points. The Hough based method is the only one which has a decreasing computation time with respect to the neighborhood size. Indeed, its duration mainly depends on the number of tested pairs of points in the neighborhood to reach a certainty threshold.

Method	Number of neighbors				
	50	100	200	300	400
PCA	0.5	1	2	3	4
2-Jets	24	4	8	11	14
VCM	10	10	11	12	11
Bilateral Filter	7	13	27	42	52
Hough	111	76	57	56	54
Hough-CNN	36	38	42	40	43
Our method	11	13	16	22	29

Table 3: Mean computation times (in seconds) for the pipe bend (0.2 m long and 0.1 m of radius) randomly sampled by 100 000 points, for 16 Gaussian noise levels with standard deviations from 0% to 200% of the mean distance between the nearest neighbors.

The computation times are also evaluated on the CAD objects data set with the maximum Gaussian noise level (0.2%) and on the simulated indoor

acquisitions (cf. Table 4). Our algorithm is slower than the Hough-CNN method and the bilateral filter on both data sets because of the great number of sharp edges in the models, however, it remains faster than the Hough based method on the objects data set. As PCV algorithm was implemented for Matlab and PCP-net was implemented to run on GPU by the authors, their computation times could not be properly integrated to this study. However, PCV is slow relatively to all considered algorithms in this work. Indeed, for the CAD data set, PCV algorithm computation time can reach up to 6h for one model when the maximum computation time of the Hough based method is 40 minutes and our algorithm's is 12 minutes. For simulated indoor scenes, PCV computation time reaches 80 minutes for one model when the maximum durations of Hough based method and our algorithm are 25 minutes and 35 seconds respectively. PCP-net maximum computation times are 65 minutes for the CAD data set and 5 minutes for the LiDAR simulations data set, this makes it much slower than our algorithm but faster than PCV.

	PCA	2-Jets	Hough	Hough-CNN	Bilateral filter	Our method
Objects	2	9	410	46	34	141
Indoor scenes	6	20	477	105	73	520

Table 4: Mean computation times (in seconds) for 14 CAD objects with 0.2% Gaussian noise and for simulated LiDAR acquisitions with 0.1% Gaussian noise.

Finally, the computation times are evaluated on real data acquisitions. They are gathered in Table 5. Our algorithm has higher computation times than the Hough-CNN method for the first and the third point cloud. However, it is faster for the second one. The basic Hough based method is the slowest one on the first and the third point clouds, however this method and the bilateral filter lead to too high a memory consumption on the second one and could not be validated while our algorithm can handle this kind of data in a limited time.

It is to note that the proposed method can be entirely parallelized because each normal estimation is independent of the others.

Method	LiDAR sensor		
	Hokuyo	Leica	Velodyne
PCA	4	29	0.6
2-Jets	16	120	2
VCM	37	308	9
Bilateral Filter	46	n.a.	79
Hough	1235	n.a.	106
Hough-CNN	135	1117	22
Our method	147	630	134

Table 5: Mean computation times (in seconds) to estimate normals on three point clouds acquired with different LiDAR sensors in structured environments. *n.a.* indicates that no results have been obtained (memory consumption exceeding).

7. Conclusion

This paper proposes a new approach to estimate normal vectors in a point cloud sampled on a piecewise smooth surface. It is based on a weighted PCA integrated in an iterative algorithm. The process is divided into two phases to make it robust to Gaussian noise and a double automatic initialization is performed to solve the issue of anisotropy in the neighborhood of sharp features. Our algorithm has been evaluated on synthetic data, on CAD objects, on point clouds obtained by LiDAR simulations and on data acquired with different LiDAR sensors. On all these tested data, it handled correctly noisy data and low samplings. It has a relatively low computation time which makes it suitable to process large point clouds such as real LiDAR acquisitions. In future work, we plan to extend the method with more advanced surface fittings such as sphere or higher degree polynomials, keeping the iterative weighting framework in order to improve its performance on smooth surfaces. Moreover, we will use this new normal estimation to improve well-known processes on LiDAR point clouds such as registration, segmentation and indoor scene modeling.

References

Alexa, M., Behr, J., Cohen-Or, D., Fleishman, S., Levin, D., Silva, C., 2003. Computing and rendering point set surfaces. IEEE Transactions on Visualization and Computer Graphics 9, 3–15.

Alliez, P., Cohen-Steiner, D., Tong, Y., Desbrun, M., 2007. Voronoi-based variational reconstruction of unoriented point sets. Proceedings of the fifth Eurographics symposium on Geometry processing , 39–48.

Amenta, N., Bern, M., 1999. Surface reconstruction by Voronoi filtering. Discrete and Computational Geometry 22, 481–504.

Avron, H., Sharf, A., Greif, C., Cohen-Or, D., 2010. ℓ_1-Sparse reconstruction of sharp point set surfaces. ACM Transactions on Graphics , 1–12.

Bae, K.H., Lichti, D.D., 2008. A method for automated registration of unorganised point clouds. ISPRS Journal of Photogrammetry and Remote Sensing 63, 36–54.

Belton, D., 2008. Improving and extending the information on principal component analysis for local neighborhoods in 3D point clouds. ISPRS Congress , 477–484.

Berger, M., Tagliasacchi, A., Seversky, L.M., Alliez, P., Levine, J.A., Sharf, A., Silva, C., 2014. State of the art in surface reconstruction from point clouds. Proceedings of the Eurographics STARs 1, 161–185.

Boulch, A., Marlet, R., 2012. Fast normal estimation for point clouds with sharp features using a robust randomized Hough transform. Computer Graphics Forum 31, 1765–1774.

Boulch, A., Marlet, R., 2016. Deep learning for robust normal estimation in unstructured point clouds. Computer Graphics Forum 35, 281–290.

Cadena, C., Carlone, L., Carrillo, H., Latif, Y., Scaramuzza, D., Neira, J., D.Reid, I., Leonard, J.J., 2016. Past, present, and future of simultaneous localization and mapping: towards the robust-perception age. IEEE Transactions on Robotics 32, 1–27.

Cazals, F., Pouget, M., 2005. Estimating differential quantities using polynomial fitting of osculating jets. Computer Aided Geometric Design 22, 121–146.

Che, E., Olsen, M.J., 2018. Multi-scan segmentation of terrestrial laser scanning data based on normal variation analysis. ISPRS Journal of Photogrammetry and Remote Sensing 143, 233–248.

Dey, T., Sun, J., 2006. Normal and feature approximations from noisy point clouds. Foundations of Software Technology and Theoretical Computer Science 4337, 21–32.

Fleishman, S., Cohen-Or, D., Silva, C., 2005. Robust moving least-squares fitting with sharp features. ACM SIG-GRAPH 24, 544–552.

Fleishman, S., Drori, I., Cohen-Or, D., 2003. Bilateral mesh denoising. ACM Trans. Graph. 22, 950–953.

Gouraud, H., 1971. Continuous shading of curved surfaces. IEEE Transactions on Computers C, 623–629.

Guennebaud, G., Gross, M., 2007. Algebraic point set surfaces. ACM Transactions on Graphics 26, 23.

Guerrero, P., Kleiman, Y., Ovsjanikov, M., Mitra, N.J., 2018. Learning local shape properties from raw point clouds. Computer Graphics Forum 37, 75–85.

He, R., Hu, B., Yuan, X., Wang, L., 2014. Robust recognition via information theoretic learning, The Netherlands. Springer.

Hoffman, R., K., J.A., 1987. Segmentation and classification of range images. IEEE Transactions on Pattern Analysis and Machine Intelligence 9, 608–620.

Hoppe, H., DeRose, T., Duchamp, T., McDonald, J., Stuetzle, W., 1992. Surface reconstruction from unorganized points. ACM SIGGRAPH Computer Graphics 21, 71–82.

Huang, H., Wu, S., Gong, M., Cohen-Or, D., Ascher, U., (Richard), H.Z., 2013. Edge-aware point set resampling. ACM Trans. Graph. 32, 1–12.

Huber, P.J., 2011. Robust statistics. International Encyclopedia of Statistical Science , 1248–1251.

Jin, S., Lewis, R.R., West, D., 2005. A Comparison of algorithms for vertex normal computation. The Visual Computer 21, 71–82.

Jones, T., Durand, F., Desbrun, M., 2003. Non-iterative, feature-preserving mesh smoothing. ACM Trans. Graph. 22, 943–949.

Jones, T., Durand, F., Zwicker, M., 2004. Normal improvement for point rendering. Computer Graphics Forum 24, 53–56.

Kim, S., Kim, H.G., Kim, T., 2018. Mesh modelling of 3D point cloud from UAV images by point classification and geometric constraints. International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences - ISPRS Archives 42, 507–511.

Lipman, Y., Cohen-Or, D., Levin, D., Tal-Ezer, H., 2007. Parameterization-free projection for geometry reconstruction. ACM Transactions on Graphics 26, 6.

Lipman, Y., DanielCohen-Or, Levin, D., Tal-Ezer, H., 2003. Mesh-independent surface interpolation. Geometric Modeling for Scientific Visualization , 37–49.

Mederos, B., Velho, L., de Figueiredo, L.H., 2003. Robust smoothing of noisy point clouds. SIAM Conference on Geometric Design and Computing .

Mérigot, Q., Ovsjanikov, M., Guibas, L.J., 2011. Voronoi-based curvature and feature estimation from point clouds. IEEE Transactions on Visualization and Computer Graphics 17, 743–756.

Mitra, N.J., Nguyen, A., Guibas, L., 2003. Estimating surface normals in noisy point cloud data. Nineteenth Annual Symposium on Computational Geometry , 322–328.

Ochmann, S., Vock, R., Wessel, R., Klein, R., 2016. Automatic reconstruction of parametric building models from indoor point clouds. Computers and Graphics (Pergamon) 54, 94–103.

Ouyang, D., Yung, F.H., 2005. On the normal vector estimation for point cloud data from smooth surfaces. Computer Aided Design 37, 1071–1079.

Öztireli, A., Guennebaud, G., M.Gross, 2009. Feature preserving point set surfaces based on non-linear kernel regression. Proceedings of Eurographics 28, 493–501.

Pauly, M., Gross, M., Kobbelt, L., 2002. Efficient simplification of point-sampled surfaces. IEEE conference on Visualization , 163–170.

Pomerleau, F., Liu, M., Colas, F., Siegwart, R., 2012. Challenging data sets for point cloud registration algorithms. The International Journal of Robotics Research 31, 1705–1711.

Rabbani, T., van den Heuvel, F., Vosselman, G., 2006. Segmentation of point clouds using smoothness constraint. International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences - Commission V Symposium 'Image Engineering and Vision Metrology' 36, 248–253.

Remondino, F., 2003. From point cloud to surface the modeling and visualization problem. International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences 26.

Sanchez, J., Denis, F., Checchin, P., Dupont, F., Trassoudaine, L., 2017. 3D point cloud repository. `projet.liris.cnrs.fr/pcr/`.

Sun, X., Rosin, P., Martin, R., Langbein, F., 2007. Fast and effective feature-preserving mesh denoising. Transactions on Visualization and Computer Graphics, IEEE 13, 925–938.

Sun, Y., Schaefer, S., Wang, W., 2015. Denoising point sets via l0 minimization. Computer Aided Geometric Design 35, 2–15.

Taubin, G., 2001. Linear anisotropic mesh filters. RC22213 Computer Science , 110–51.

Weinmann, M., Jutzi, B., Hinz, S., Mallet, C., 2015. Semantic point cloud interpretation based on optimal neighborhoods, relevant features and efficient classifiers. ISPRS Journal of Photogrammetry and Remote Sensing 105, 286–304.

William, S.C., 1979. Robust locally weighted regression and smoothing scatterplots. Journal of the American Statistical Association , 859–836.

Yagou, H., Ohtake, Y., Belyaev, A., 2002. Mesh smoothing via mean and median filtering applied to face normals. Geometric modeling and processing. Theory and applications , 124–131.

Yagou, H., Ohtake, Y., Belyaev, A., 2003. Mesh denoising via iterative alpha-trimming and nonlinear diffusion of normals with automatic thresholding. Computer Graphics International, IEEE , 28–33.

Yang, M., Lee, E., 1999. Segmentation of measured point data using a parametric quadric surface approximation. Computer Aided Design 31, 449–457.

Yuzhong, S., E., B.K., 2004. Fuzzy vector median-based surface smoothing. Computer Graphics International, IEEE 10, 252–265.

Zhang, J., Cao, J., Liu, X., Chen, H., Li, B., Liu, L., 2018. Multi-normal estimation via pair consistency voting. IEEE Transactions on Visualization and Computer Graphics , 1077–2626.

Zheng, Y., Fu, H., Au, O.K., Tai, C., 2011. Bilateral normal filtering for mesh denoising. IEEE Transactions on Visualization and Computer Graphics 17, 1521–1530.

Zheng, Y., Li, G., Xu, X., Wu, S., Nie, Y., 2018. Rolling normal filtering for point clouds. Computer Aided Geometric Design 62, 16–28.

Acknowledgements

Julia Sanchez held a doctoral fellowship from la Région Auvergne-Rhône-Alpes. The authors would like to thank Junjie Cao for providing the code used for comparison. All meshes used in this paper are courtesy of `free3d.com`.

Appendix A. PCA

If the plane to fit contains $\mathbf{p}_0$, the equivalence in the least squares sense is to find $\mathbf{n}^*$ (such that $||\mathbf{n}^*|| = 1$) which minimizes the following function:

$$\sum_i \left(\mathbf{n} \cdot (\mathbf{p}_i - \mathbf{p}_0)\right)^2 . \tag{A.1}$$

The covariance matrix is defined as:

$$C := \frac{1}{N} P^T P \,, \tag{A.2}$$

with

$$P := \begin{pmatrix} (\mathbf{p}_{1_x} - \mathbf{p}_{0_x}) & (\mathbf{p}_{1_y} - \mathbf{p}_{0_y}) & (\mathbf{p}_{1_z} - \mathbf{p}_{0_z}) \\ \vdots & & \\ (\mathbf{p}_{N_x} - \mathbf{p}_{0_x}) & (\mathbf{p}_{N_y} - \mathbf{p}_{0_y}) & (\mathbf{p}_{N_z} - \mathbf{p}_{0_z}) \end{pmatrix}$$

Thus, the minimization of (A.1) can be rewritten in bilinear form:

$$\min_{||\mathbf{n}||=1} \mathbf{n}^T C \mathbf{n} \,. \tag{A.3}$$

The solution of Eq. (A.3) is the eigenvector corresponding to the smallest eigenvalue of the matrix C.

Appendix B. Internal evaluation

Firstly, the resulting normals of the proposed algorithm are evaluated with and without the refinement step (cf. Figure B.18) on the first synthetic model: an intersection of two planes containing 15 000 points randomly sampled (please refer to Section 6 for more information about the models). Gaussian noise is added to the points with a standard deviation lying between 1 mm and 2 cm. The neighborhood is defined by 300 points. Figure B.18 shows that the refinement step leads to more robustness against noise. The mean angular error with respect to the ground truth normals is reduced up to six times when it is added to the process.

Secondly, in Figure B.19, the accuracy of our algorithm is compared with and without the second optimization on the same synthetic model without noise, varying the density of points independently on the two planes. We

observe that this step leads to a reduction of the mean angle error. Notably, when one face contains 2 times more points than the other, the second optimization allows reducing 90 times the mean angle error.

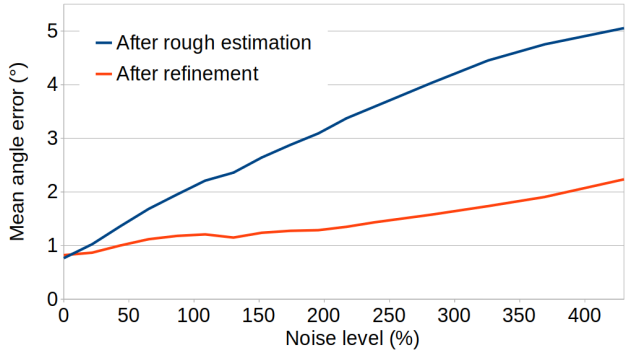


Figure B.18: Mean angle error resulting from our algorithm on the intersection of two planes (1×0.5 m) sampled by 15 000 points with increasing Gaussian noise (noise levels expressed as a percentage of the standard deviation relatively to the mean distance between the nearest neighbors). The first optimization is evaluated with and without the refinement step using 300 neighbors.

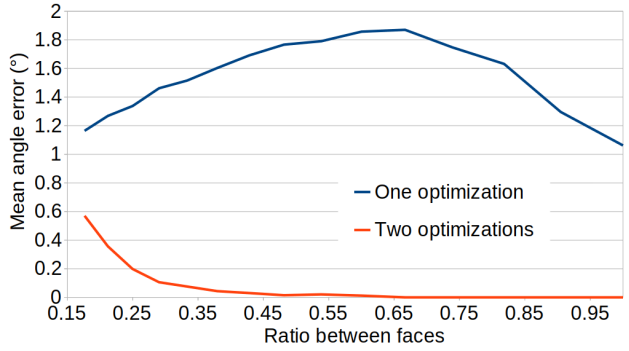


Figure B.19: Mean angle resulting from our algorithm on the intersection of two planes (1 × 0.5 m) sampled by 15 000 points in a non-uniform way: one face contains more points than the other. The ratio between the density of the two faces is increased. The neighborhood is defined by 300 points.

Thirdly, the convergence parameters are studied. Figure B.20 shows the variation of the mean angle error depending on the value of r_{max} (cf. Section 4.2.3) on the planes model, for four different noise levels. It can be noticed that the mean angle error decreases when μ_{lim} decreases, until an optimal

point. When μ_{lim} is too small, the algorithm may diverge and the error can increase. Moreover, this graph confirms that a good value for r_{max} is $r_{max} = 0.5\sigma'$ when the model does not contain curvature, with σ' the standard deviation of the noise in one direction of the space.

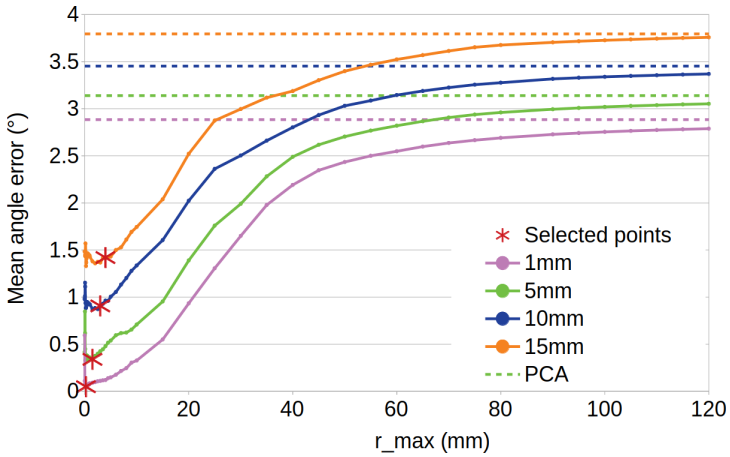


Figure B.20: Mean angle error evaluated on the intersection of two planes (1×0.5 m) randomly sampled by 15 000 points. The value of r_{max} is increased (please refer to text for variables definition). The neighborhood is defined by 300 points. The evaluation is performed for 4 Gaussian noise levels (noise level expressed as the standard deviation in mm). The errors resulting from PCA normal estimations are represented with dashed lines.

Then, for models including curvature, we define $r_{max} = X + 0.5\sigma'$ with X the effect induced by curvature. A model of two intersecting pipes is sampled by 100 000 points to evaluate the effect of curvature (please refer to Section 6 for more information about the models). Figure B.21 shows the variation of the mean angle error depending on the value of X for four Gaussian noise levels. It shows that the best results are obtained for a value close to $X = 0.8$ mm which corresponds to the length X_{max} defined in Section 4.2.3. Hence, it confirms that X_{max} is a good choice to set up the convergence parameter μ_{lim}. Furthermore, in these two graphs, it can be verified that the PCA estimations correspond to the asymptotes of the curves relative to the proposed algorithm when μ_{lim} tends to infinity.

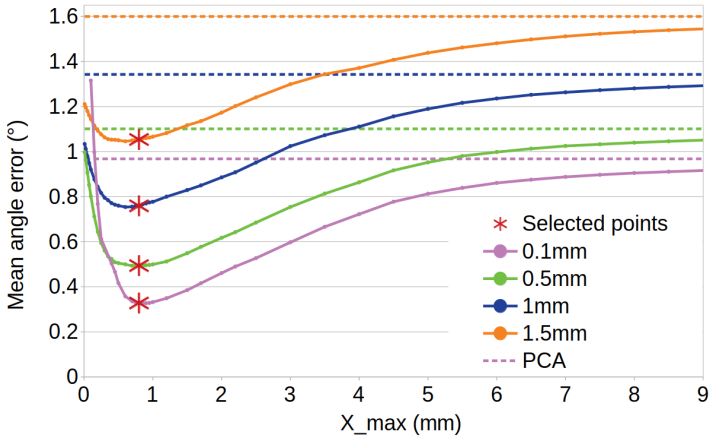
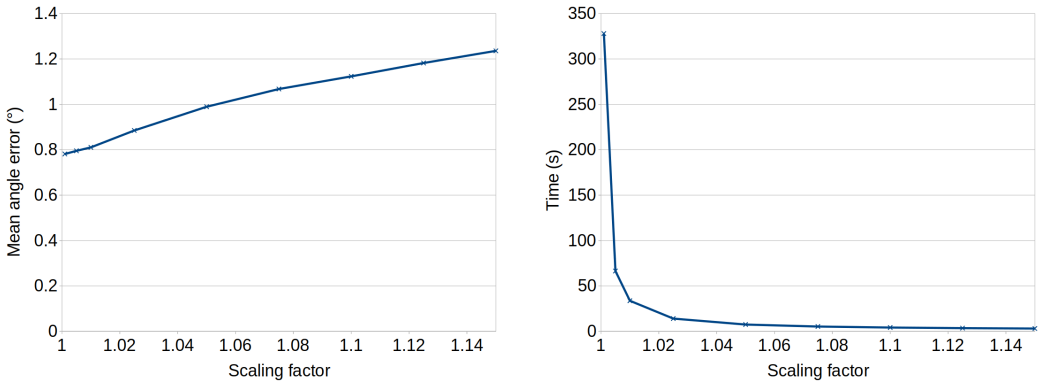


Figure B.21: Mean angle error evaluated on the pipe bend (0.2 m long and 0.1 m of radius) randomly sampled by 100 000 points. The neighborhood is defined by 200 points. The value of X is increased (please refer to text for variables definition). The evaluation is performed for 4 Gaussian noise levels (noise level expressed as the standard deviation in mm). The errors resulting from PCA normal estimations are represented with dashed lines.

The effect of the scaling factor on the accuracy is shown in Figure B.22a. Its impact on computation time is represented in Figure B.22b.



(a) Mean angle error. (b) Computation time.

Figure B.22: Evaluation of the algorithm when varying the value of the scaling factor (please refer to text for variables definition) on the intersection of two planes (1×0.5 m) sampled by 15 000 points for 16 Gaussian noise levels with standard deviations from 0% to 400% of the mean distance between the nearest neighbors. The neighborhood is defined by 300 points.

These figures show that the division factor has a limited influence on the accuracy of the result. Nevertheless, a compromise must be found to limit the computation time while optimizing the accuracy of the algorithm. We experimentally chose 1.01 in all experiments of this work.

Finally, we present an evaluation in Table B.6 to prove the efficiency of the selection of points to limit the number of optimizations performed as explained in Section 4.2.4. This table shows that there is little difference between mean angle errors when the input points are all processed by the iterative optimizations or when input points are pre-selected. However, the time can be significantly reduced (up to 4 times). Then, the computation time of the algorithm is directly linked with the geometry of the model: the algorithm gets slower as the number of neighborhoods containing a curvature discontinuity increases.

		Planes	Pipes
PCA	Mean error (°)	3.35	1.27
	Time (s)	1	2
Without threshold	Mean error (°)	0.72	0.68
	Time (s)	53	59
With threshold	Mean error (°)	0.83	0.71
	Time (s)	34	14

Table B.6: Evaluation of the selection of input points of the optimization step on two synthetic models (please refer to text for more information about these models), giving the computation time and average of the mean angle errors for 16 Gaussian noise levels. The standard deviation of the noise is increased from 0% to 400% of the mean distance between the nearest neighbors for the planes model and from 0% to 200% for the pipes model.

Appendix C. PCP-net evaluation

	PCA	Our method	PCP-net
Planes intersection	3.35	0.81	23.86
Pipes bend	1.27	0.71	3.02

Table C.7: Mean angular error (in degrees) of normals resulting from PCP-net algorithm for synthetic data.

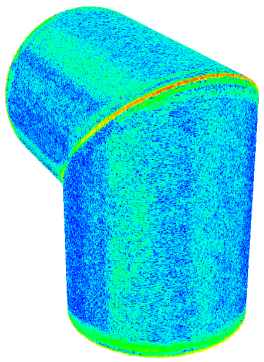


Figure C.23: Angle error on the pipe bend model with the same color scale as used in Figure 8 (cf. Section 6.1.1).