Anisotropic Octrees: a Tool for Fast Normals Estimation on Unorganized Point Clouds

Joris Ravaglia
CARTEL
Université de Sherbrooke
Canada
Aix-Marseille Université
LSIS
France
joris.ravaglia@usherbrooke.ca

Alexandra Bac
Aix-Marseille Université
Laboratoire des sciences de l'information et des systèmes (LSIS)
UMR CNRS 7296
France
alexandra.bac@univ-amu.fr

Richard A. Fournier
Centre d'Applications et de Recherche en Télédétection (CARTEL)
Université de Sherbrooke
Sherbrooke (QC) J1K 2R1
Canada
richard.fournier@usherbrooke.ca

ABSTRACT

With the recent advances in remote sensing of objects and environments, point cloud processing has become a major field of study. Three-dimensional point cloud collected with remote sensing instruments may be very large, containing up to several tens of billions of points. This imposes the use for efficient and automatic algorithms to extract geometric or structural elements of the scanned surfaces. In this paper, we focus on the estimation of normal directions in an unorganized point cloud and provide a curvature indicator. We avoid point-wise operations to accelerate the running time for normals estimation. Instead, our method rely on an innovative anisotropic partitioning of the point cloud using an octree structure guided by the geometric complexity of the data and generates patches of points. These patches are then approximated by a quadratic surface in order to estimate the normal directions and curvatures. Our method has been applied to six models of various types presenting different characteristics and performs, in average, 2.65 times faster than multi-threads implementations available in current pieces of software. The results obtained are a compromise between running time efficiency and normals accuracy. Moreover, this work opens up promising perspectives and can be easily inserted in wide range of workflows.

Keywords

Point cloud, normals, curvature, octree, anisotropy, quadratic surface

1 INTRODUCTION

With the current advances in photogrammetry and the availability of instruments recording 3D information (e.g. depth sensors and laser scanning devices from airborne, terrestrial or mobile platforms), point clouds are becoming commonly used in various fields. Data acquired with these tools often contain several tens of millions of points. Processing these large data sets is still a challenge and computing geometric information such as normals or curvature is a real issue. A major difficulty in such processing is the absence of neighbourhood information: unlike meshes, point clouds are unstructured raw data. Consequently, extracting information from a point cloud can be time consuming. Thus, there is a need for new algorithms oriented towards fast point cloud processing.

Normal directions are crucial geometric information for surface analysis. They are a key variable for a large range of algorithms and point clouds analysis approaches [Li08]. Therefore, normal estimation techniques have been widely studied (see review by [Kl09]). However, the estimations of point's normal can be challenging and computationally expensive. Normals estimation can be divided into three major approaches: optimisation-based, local-weighting-based, Hough transform, and Voronoï-based.

The optimisation-based approach locally approximate the point cloud with a predefined surface model from which normals can be computed. The two main models fitted to the data are planes [Ho92], and quadratic surfaces [Kl09]. However other second-order surfaces such as spheres have also been used to approximate the point cloud [Zh16]. Planes can be fitted by using either classical least square fitting, or principal component analysis (PCA) [Gu01]. In local-weighting-based approach, normal directions are estimated by different combinations of the normals deduced from a point and its neighbours [Ji05]. This approach can be ei-

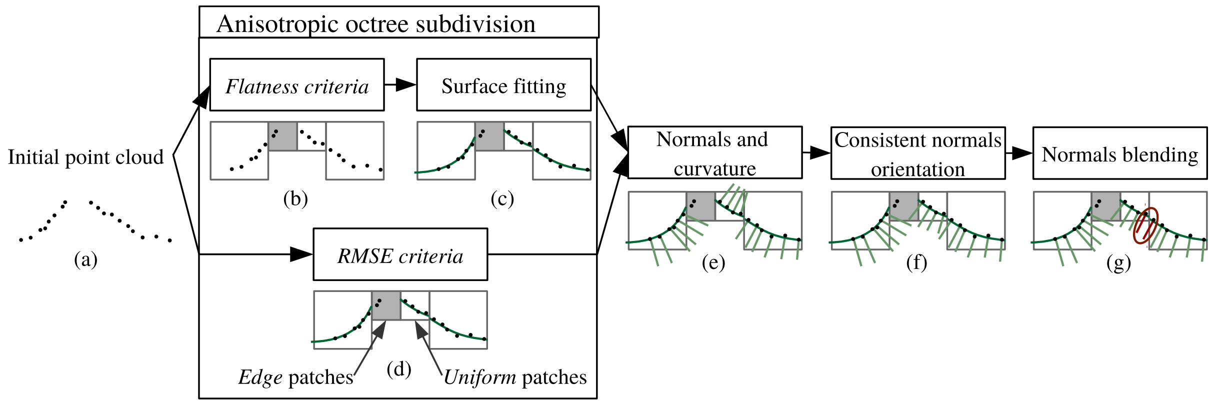

Figure 1: Overview of the methodology.

ther applied on a local Delaunay triangulation of the point cloud, on kNN graphs, or on Reinmann graphs [Kl09]. Hough transform is an alternate approach that considers a point and its neighbourhood. For each point of the cloud a reduced amount of triplets is selected, the Hough transform then provide a reliable accumulator from which robust normal directions can be extracted [Bo12]. This point-wise approach can be supported by a neuronal network to improve its robustness to noise and outliers [Bo16]. The Voronoï-based approach considers the shape of each Voronoï cells [Am99]. The algorithms using Voronoï poles to estimate normal directions may also provide curvature additional information [de05, Me11]. The above approaches are based on point-wise normal estimations and require neighbours finding procedures. While a neighbourhood query can be achieved efficiently with support of accelerating structures such as octrees or KD-trees, the repetition of this procedure for each point is time consuming.

In the present study we propose a new method, following an optimisation-based approach, as a solution to improve the efficiency of normals estimation on large in-core point clouds. Our first specific objective is to avoid point-wise operations. To reach this goal, we developed two distinct procedures that rely on an innovative adaptive subdivision of the point cloud. We propose two novel rules to guide this subdivision in accordance to the local geometric complexity of the point cloud rather than the local density. Our second specific objective is to analyse their efficiency and accuracy in different situations to evaluate their behaviour and the comparative gain in efficiency with currently available algorithms.

2 METHODOLOGY

The normal estimation method we developed is built on a simple idea: surfaces are composed of a combination of *uniform* areas (with respect to the geometrical variation and hence, containing no sharp elements) and sharp features. Consequently, the point clouds resulting from remote sensing of these surfaces also contain uniform and sharp parts. The proposed fast normal estimation procedure takes advantage of this characteristic. We avoided costly point-wise fittings by partitioning the input point cloud into patches in an anisotropic division. Unlike classical subdivisions that are only guided by the point density, our subdivision takes into account the geometry of the data. Indeed, it is designed to split the point cloud anisotropically according to the set of normals. The fast normal estimation procedure is then applied on patches of points instead of point-wise. More precisely, to achieve our objective efficiently, we developed the methodology summarised by the flow diagram of Fig. 1. This methodology is a compromise between running time and normal directions accuracy. The proposed method improves the running time efficiency based on two main aspects. First, it estimates normal directions at a patch level which lowers the computing time . It should also mitigates the impact of noise on normals estimation on large patches in the sense that with larger areas to approximate comes more information about the underlying surface. Selecting a local scale for surface fitting. Second, it provides a curvature indicator and principal curvature directions, which are important geometric features for further processing.

The proposed method has four distinct steps. As demonstrated by Zhao et al. (2016), the use of a point cloud subdivision prior to shape fitting shortens the computational time. The first step of our method addresses patches creation by diverting the octree structure from its initial purpose. Unlike Zhao et al. (2916), who use an octree partitioning guided by the point density, our patch partitioning builds an anisotropic subdivision of the data according to its geometry.

We calibrated the octree subdivision of a point cloud on its local uniformity rather than adopting the usual strategy using the point density. Figure 2 illustrates this anisotropic subdivision: areas containing sharp features such as ground-wall, wall-wall or wall-roof junctions are highly subdivided. whereas bare ground or inner

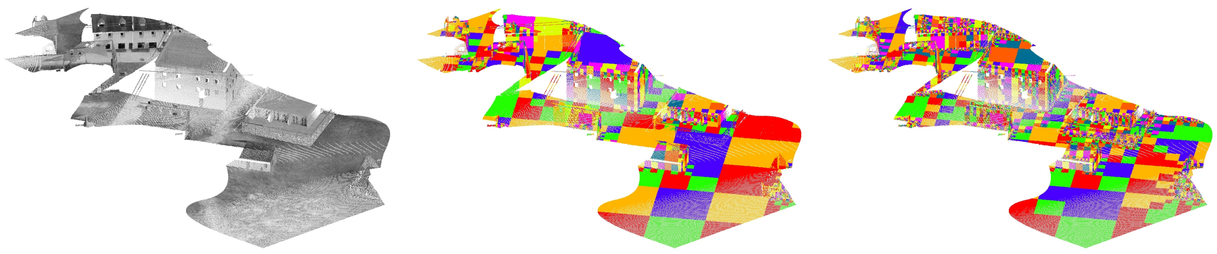

Figure 2: Anisotropic subdivision of an input urban point cloud (left), using σ_3 criterion (centre), or *RMSE* criterion (right). Sharp features creates small patches while uniform surfaces generates large patches. On an almost constant point density, the octree sudvision still creates patches of various size since it is ruled by geometric complexity.

walls are less divided and result in larger patches. This *anisotropic octree* recursively divides a point cloud into two sets of patches (Fig. 1b, 1d): (1) locally smooth patches, and (2) patches containing sharp features (we will refer to them as *edge* patches). The recursive subdivision also includes a surface fit for each generated patch (Fig. 1c, 1d). This way, we generate large patches in even areas and process them at once. Unlike point-wise approaches, our method does not require the computation of a neighbourhood for each point of the input cloud. As a consequence, the complexity of our approach is drastically lower than point-wise approaches. The second step of the method involves estimating normals and curvatures inside each resulting uniform patches using the fitted surfaces (Fig. 1e). Since these surfaces are not oriented, the resulting normal directions are possibly inconsistent across neighbouring patches. Therefore, the third step is designed to improve the coherence of the normal field. To do so, we take advantage of the fast node neighbourhood access given by the octree structure to generate a consistent orientation of normals (Fig. 1f). Eventually, the forth step blends the computed normals and curvatures along nodes boundaries in order to guarantee the smoothness of the resulting normal field (Fig. 1g).

2.1 Anisotropic subdivision

The first step of our method divides the point cloud along an octree subdivision guided by the local uniformity of the point cloud. Starting from the root of the octree $\mathscr{O}$, which contains the entire point cloud, nodes are split into eight octant patches until a uniformity criterion is satisfied.

We developed two procedures to create the anisotropic subdivision. Each of them uses a different criterion to stop the recursive subdivision. The first procedure stops the subdivision when the flatness of the patches is sufficient (Fig. 1b): the corresponding criterion is based on a flatness index (Fig. 3). The second procedure stops the subdivision as soon as the patches can be accurately approximated by a surface model (Fig. 1d): the corresponding criterion thus is based on the modelling error.

Both criteria are designed to avoid an over-segmentation of the initial point cloud. Indeed, only patches containing sharp features will be further divided, hence generating large patches for uniform areas, and small patches for sharp features, that is, an anisotropic subdivision. The two procedures have different strengths and balance running time and accuracy, as discussed in the results and the discussion sections. While the first procedure is based on flatness, it requires less computations but captures fewer details on the point clouds. Indeed, figure 2 shows that the σ_3 criterion is more permissive regarding the presence of sharp features and creates patches containing a small amount of sharp features such as ground-wall junctions. The second procedure based on modelling error is more accurate, but involves more calculations. While the verification of the first criterion requires the use of plane fitting operations, quadratic surface fitting is necessary for the evaluation of the second criterion.

2.1.1 *Local plane fitting*

Plane fitting

Let $o \in \mathscr{O}$ be an octree node, $\{p_1, p_2, \cdots, p_m\}$ be the points contained in o, and $\overline{p}$ be their centroid. It has been proven that computing the least square fitted plane P going through $\overline{p}$ and approximating the points p_i can be obtained by a centred PCA of the p_i [Pa02]. Indeed, such a plane is completely determined by its normal $\vec{n}$. Hence, least square fitting is equivalent to minimising the following least square error:

$$\begin{aligned}\sum_{i=1}^{m} \mathrm{d}(p_i, P)^2 &= \sum_{i=0}^{m} \left(\frac{(p_i - \overline{p}) \cdot \vec{n}}{\|\vec{n}\|} \right)^2 \\ &= \frac{1}{\|\vec{n}\|^2} \vec{n}^t \times \mathrm{Cov}(\overline{p_i}) \times \vec{n}\end{aligned} \qquad (1)$$

where Cov denotes the covariance matrix, and $\overline{p_i} = p_i - \overline{p}$. Given $\lambda_1 \geqslant \lambda_2 \geqslant \lambda_3$ the eigenvalues of the covariance matrix, and $\vec{v}_1, \vec{v}_2, \vec{v}_3$ the associated eigenvectors, the fitted plane P is actually the plane containing $\overline{p}$ and whose normal direction is $\vec{v}_3$.

Flatness index

The residuals of the PCA plane fitting are given by λ_3. Thus these residuals may be considered as a flatness indicator. However, λ_3 not only depends on the number

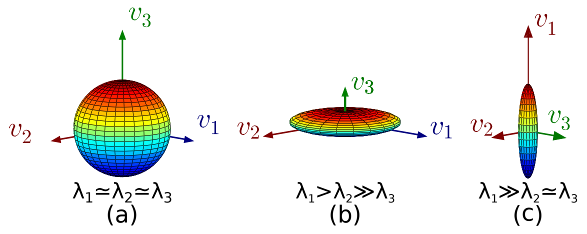

Figure 3: Different point cloud dispersion represented as ellipsoids and corresponding PCA results. Eigenvectors are scaled according the eigenvalue. The colours represents the height map of each ellipsoid for a better appreciation of the shape.

of points considered, but also on the size of the point cloud. Hence we selected an indicator σ_3 called surface variation [Pa02], providing an homogeneous, orientation, and scale independent indicator:

$$\sigma_3 = \frac{\lambda_3}{\lambda_1 + \lambda_2 + \lambda_3} \qquad (2)$$

From the PCA properties, σ_3 can be seen as the ratio of inertia kept by the plane normal over the total inertia of the point set. This normalisation produces an index $\sigma_3 \in \left[0, \frac{1}{3}\right]$ that is independent of the patch size and of the number of points considered: the lower σ_3, the flatter the corresponding sampled surface (Fig. 3b). This index can be applied as a criterion to regulate the anisotropic recursive subdivision.

2.1.2 *Quadratic surface fitting*

The second procedure models patches as quadratic surfaces for a closer approximation of their geometry. Computing a best-fit surface requires a local frame. An efficient way to obtain this system is to compute an approximate tangent plane using the previously described PCA plane fitting, and to consider the orthogonal resulting basis $B = (\vec{v}_1, \vec{v}_2, \vec{v}_3)$, along with $\overline{p}$ its origin.

Given a local frame and a point cloud $\mathscr{P} = \{p_1, p_2, \cdots, p_m\}$ expressed in this basis, we use the classical least square method ([Pr93]) to fit an elevation surface S of equation $S(x,y) = a^2 + bxy + cy^2 + dx + ey + f$ minimising the following distance:

$$d(S, \mathscr{P}) = \sum_{i=1}^{n} (z_i - S(x_i, y_i))^2 \qquad (3)$$

The accuracy of this fitting procedure can be evaluated with the corresponding root mean square error (RMSE), which can be used as a criterion to guide the octree subdivision.

2.1.3 *Anisotropic subdivision*

Let us now introduce our anisotropic octree subdivision. Whereas standard octree subdivision iteratively divides voxels according to the relative density of points, we control the subdivision and eventually stop it according to either of the two criteria presented: (1) the σ_3 criterion, and (2) the *RMSE* criterion. The octree creation using the σ_3 criterion performs a PCA plane fitting and stops the subdivision according to a threshold over σ_3 (Fig. 1b). In the *RMSE* criterion, after the initial PCA plane fitting a quadratic surface is fitted and subdivision is stopped according to a threshold over the resulting *RMSE* (Fig. 1d).

The σ_3 criterion therefore requires less computations and as resulting patches have a σ_3 value lesser than a given threshold, they are relatively flat and do not contain folds or sharp features. The *RMSE* criterion requires more calculations since both PCA plane fitting and quadratic surface fitting are computed at each level of the recursive subdivision. However, this criterion assesses more accurately the patches and improves the anisotropy of the resulting subdivision.
Starting from this subdivision, each patch is them modelled by an elevation surface to compute normals. Whereas the *RMSE* criterion directly provides an approximating surface (Fig. 1d), in the case of the σ_3 criterion, we eventually fit a quadratic surface in the least squares sense over each patch. (Fig. 1c).

Regardless of the criterion applied, we set two additional conditions for a patch to be split. At critically small scales, point clouds no longer represent the sampled surfaces. Therefore we use a threshold on the patches dimensions D_{min} to avoid dividing further small nodes. Finally, since we use PCA to locally fit a plane, we set a threshold n_{min} on the number of points contained within the patch to ensure a reliable geometric fitting.

As a result of the recursive binary space division, two main reasons exist for an octree node to include a non uniform patch that cannot be modelled by a surface. The first reason is that nodes may contain less than n_{min} points. This happens: (1) mainly when a node containing an edge is recursively split without reaching the desired level of smoothness, or (2) when a node splitting isolates a few points inside an octant, even if the other nodes reach the smoothness criteria. The second reason for a node not to contain a uniform patch is that the patch is more likely to represent a 3D curve than a surface (Fig. 3c). In order to detect these patches, we use another indicator derived from the PCA plane fitting. These *edge* patches are detected by using the following ratio:

$$\sigma_2 = \frac{\lambda_2}{\lambda_1 + \lambda_2 + \lambda_3} \qquad (4)$$

No surface can be fitted on *edge* patches. Similarly, patches not containing enough points are not modelled by a surface since they may contain a sharp feature that cannot be accurately approximated by a quadratic

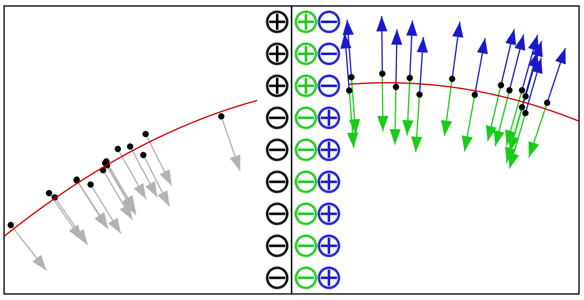

Figure 4: Surfaces are oriented by considering the sign of implicit surfaces at points sampled on the face of contact between neighbour nodes.

model (e.g. several piecewise surface). In both cases, these patches are ignored in the later procedure.

2.2 Normals estimation and curvature indicator

Once uniform patches have been obtained from the anisotropic octree and modelled by a surface S, normals can be estimated in an efficient manner. The normal direction at a point p_i of the patch is approximated by the normal of its projection p'_i onto the surface S. Since the surface equation is known, obtaining the normal $\overrightarrow{n}$ is straight forward. However, at this point, the orientation of the fitted surface cannot be assessed (i.e. discriminating interior and exterior space), and neither do the normal directions.

Similarly to normal directions, principal curvatures can be derived from the fitted surface equation. The first and second fundamental forms of each surface are computed for each point p_i as the curvature of their projection on the fitted surface p'_i. Hence, the curvature tensor can be computed to obtain the corresponding eigenvalues and eigenvectors. However, because the fitted surfaces are quadratic, and since the point cloud division generates uniform patches, the curvature may not be captured with high precision. The resulting eigenvalues of the tensor thus differ from the actual main curvatures of the surface. Nonetheless, this procedure provides a potential indicator of the curvature which can be used to analyse the changes of its value across the point cloud.

2.3 Consistent normal orientation

In order to enhance the consistency of the normal directions, we propose a way to locally orient surfaces. Starting from a patch, neighbouring patches are probed according to their adjacency to propagate a consistent orientation (two patches are considered to be neighbours if their corresponding octree nodes share a common face). Let N_1 and N_2 be two neighbour nodes, S_1 and S_2 be the surface fitted in each node, and f the face shared by N_1 and N_2. In order to orient N_2 relatively to N_1, a set of m points are sampled uniformly on the face f.

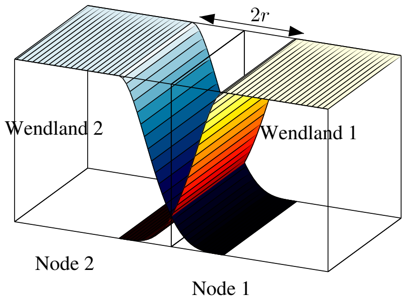

Figure 5: Normals directions close to a node faces are weighted with the neighbours node surface. The weights are computed as a Wendland function around the faces of contact.

	M	CCP	CCQ	CCT	*RMSE*	σ_3
M	1.7	2.4	1.7	0.6	3.2	6.5

Table 1: Average median error for Meshlab (M), CloudCompare's plane fitting (CCP), quadratic fitting (CCQ) and triangulation (CCT), *RMSE* octree, and σ_3 octree.

Now, S_1 and S_2 can be seen as the surfaces corresponding to the zero level set of two implicit functions I_1 and I_2 corresponding to their quadratic equation.. At each point p_i sampled on f, the signs of $I_1(p_i)$ and $I_2(p_i)$ is computed. The orientation of N_2 is set to minimise the difference of signs at the sampled points (Fig. 4).

2.4 Normals blending

Since patches are analysed separately, the junction between the fitted surfaces of two neighbouring patches are discontinuous. Thus, two points inside a small neighbourhood radius may have divergent normals. To minimise this potential bias, we propose a way to blend the normal direction at the nodes junctions taking into account the local information of multiple surfaces. Note N_1, N_2 two neighbouring nodes of size d_1 and d_2 and f the face they share. We define a radius of influence r:

$$r = \frac{min(d_1, d_2)}{3} \tag{5}$$

around f in which normals will be blended (Fig. 5). At each point p, one or several nodes and their surfaces S_i will then influence the estimation of its normal direction. The normal direction $\overrightarrow{\mathcal{N}}_p$ at p will be ultimately estimated as a weighted sum of the normals $\overrightarrow{n_i}$ at p on the set of surfaces S_i:

$$\overrightarrow{\mathcal{N}}_p = \sum_i \omega_i \overrightarrow{n_i} \tag{6}$$

where $\overrightarrow{n_i}$ is the normal at p_i estimated on the surface S_i, and the weights ω_i are computed based a Wendland's function of the distance d_i from p to f_i (Fig. 5):

$$\omega_i(d) = \begin{cases} (1-d)_+^4(4d+1) & \text{if } d < r \\ 0 & \text{else} \end{cases} \tag{7}$$

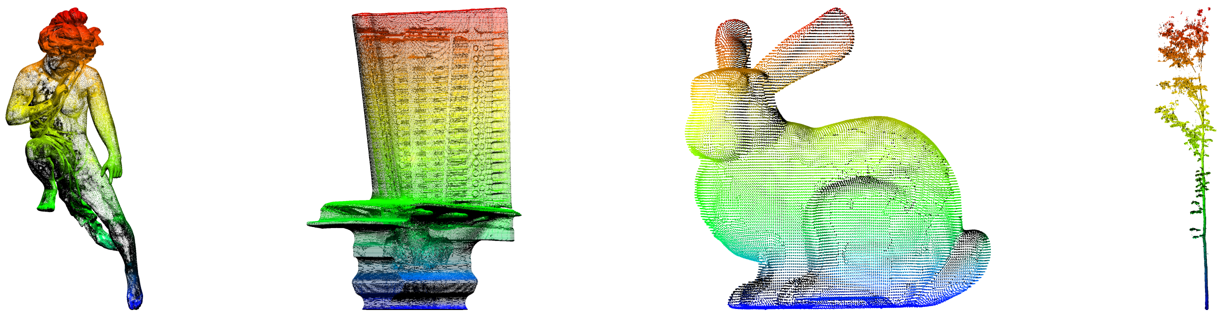
Figure 6: Models used to tests our methods. From left to right: *Angel*, *Blade*, *Bunny*, *Tree* (*Urban* and *Lucy* models are shown in Fig. 2 and 7). Colours are only used to emphasise the point cloud rendering.

In order to blend local estimates smoothly inside a limited radius, we choose to weight normals along boundaries using Wendland's functions. Indeed, such functions are smooth (to any given degree of derivability) while having a compact support. Linear interpolation is clearly not smooth enough whereas Gaussians functions are not compactly supported (whereas we intend to blend estimates only along the boundary).

3 RESULTS

We tested our methodology on six different point clouds with various characteristics and number of points (Fig. 6). These models are indeed representative of a large range of applications: arts, forestry, urban environments and industrial applications. Additionally, while the *Lucy* model contains a higher number of points, the *Angel* model shows variations in point sampling where the point cloud is denser around the details of the statue. Note also that the *Tree* point cloud contains a little noise, mainly around the foliage due to the particular limitations of TLS instruments in forest environments. Four out of six models (*Angel*, *Bunny*, *Blade*, and *Lucy*) are accessible from the Stanford 3D Scanning Repository, or the Large Geometric Models Archive of Georgia Institute of Technology. They were available as a mesh with known normal directions which were used as reference directions. The other two point clouds represent a tree (acquired with terrestrial laser scanning), and a simulated urban environment. These six data sets were used to analyse both the running time efficiency and the accuracy of our method. Results from our method were compared with those obtained by two geometric modelling pieces of software providing normals estimation procedures. In Meshlab, the normals estimation is based on a point-wise k-neighbours operation. CloudCompare includes three different methods: point-wise local plane fitting, point-wise local quadratic surface fitting, and triangulation. Our anisotropic octree subdivision with both criteria (σ_3 and $RMSE$), were also evaluated for a total of six different methods.

The running time for all six normal estimation methods on the six point clouds are presented in Fig. 8 (note the logarithmic scale). We ran the tests on an Intel© Core^{TM} i7-4790 CPU (3.60 GHz). Our methods outperformed the methods in Meshlab and CloudCompare on all tested point clouds in terms of running time. More specifically, the σ_3 octree was faster than the *RMSE* octree. These results can be explained by (1) the higher tolerance of the σ_3 criterion to sharp features which induces less subdivisions of the point cloud, and (2) the need to compute the quadratic surface fitting at each level of recursion when using the *RMSE* criterion, whereas this surface fitting is only computed on the octree leaves when the σ_3 criterion is selected.

The complexity of the anisotropic subdivision is related to the size and the local uniformity of the point cloud. In contrast, since Meshlab and CloudCompare perform point-wise operations, the resulting complexity is then dependent on the number of points in the cloud. In point wise optimisation-based approach, for each point, a neighbour query is required before fitting a surface to the extracted neighbourhood (either with a plane fitting, a quadric surface fitting, or a plane fitting followed by a quadratic surface fitting). Such neighbour finding procedure can be achieved in $\mathcal{O}(log(n))$. Denoting by F the complexity of the chosen fitting procedure and S the complexity of the accelerating structure creation, the overall complexity of point-wise approaches is thus $\mathcal{O}(S+(n*(log(n)+F)))$.
The complexity of our methodology, in the worst case, is that of the octree creation (that is $S=\mathcal{O}(n(d+1)$ where d is the depth of the octree) plus a PCA plane fitting and a surface fitting in each node of the octree. The overall complexity of the normal estimation in an anisotropic octree containing M nodes is thus $\mathcal{O}(S+(M*(PCA+F)))$. In the vast majority of point clouds, $M \ll n$. Hence the low complexity of our approach compared to point-wise approaches witnessed by empirical results.

When comparing computing time, we found that the CloudCompare triangulation method is, on average, the most efficient concurrent of our methods. The *RMSE* anisotropic octree subdivision speeds up, on average, the normals estimation by a factor 2.65 in comparison with the CloudCompare triangulation. This factor rises

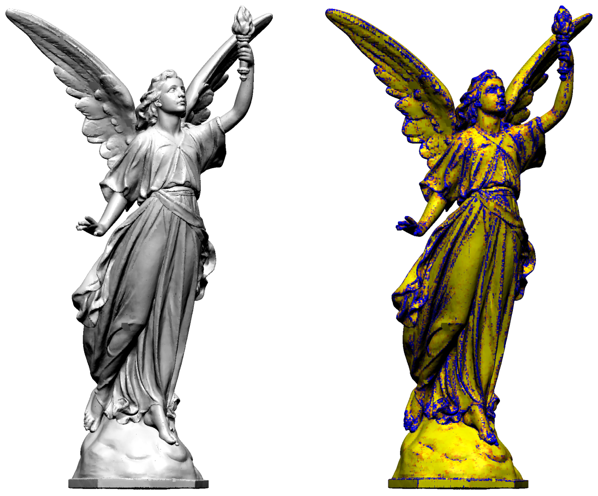

Figure 7: Estimated normal directions and curvature on the *lucy* model using the *RMSE* criterion. Point cloud is rendered using estimated normal directions (left), and curvature indicator (right).

to 4.92 when considering the σ_3 octree. Overall, the σ_3 octree performs in average 2.09 times faster than the *RMSE* octree.

We also analysed the normal estimation accuracy and computed the curvature indicators for all six methods on four of the point clouds: the *Angel*, *Bunny*, *Blade*, and *Lucy* models (Fig. 7). To do so, we considered the normals of the mesh models as the reference. Hence, we evaluated the estimation error as the angle between the reference normals and the estimated ones (Fig. 9). Let us also point out that no normal vectors were estimated on the points identified as belonging to *edge* patches (corresponding to sharp features). These points are actually discarded during the comparison procedure. The *RMSE* octree produces an error distribution similar to the plane fitting and triangulation methods from CloudCompare but with higher errors on the *Bunny* model. The σ_3 octree however is less accurate with error distributed on a larger range. Average median errors are summarised in table 1. As illustrated in figure 9, even though the values of the curvature indicator differs from the actual Gaussian curvature, the variations of this indicator are actually relevant and capture the reliefs of the statue.

Finally we illustrate the differences between each method, including the tree and urban data sets (Fig. 10). A difference pattern emerges as the anisotropic octree σ_3 induces higher differences for each compared method, and for the majority of data set. Overall, the difference between Meshlab and CloudCompare is regular, and the octree *RMSE* presents low difference variations with these methods. However, the data sets characteristics influence the resulting differences as shown in the *Tree* and *Angel* data sets. The differences observed on the *Tree* data set are due to the high geo-

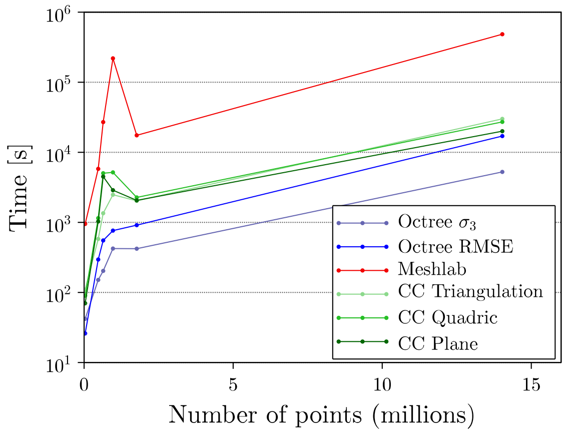

Figure 8: Comparison of computational time for normal directions estimation using different methods.

metric complexity of the analysed object, as well as the occlusion effect inherent to terrestrial laser scanning. The *Angel* point cloud in turn has a non homogeneity of the point sampling on the surface which may impact the neighbourhood search operations. On the contrary, the *Urban* point cloud present lowest inter-methods differences including the σ_3 octree. This is explained by the simplicity of the sampled scene: the point cloud is mainly composed of flat surfaces such as soil, walls, and roofs.

4 DISCUSSION

We achieved our objective and developed a method for fast normal estimation. Indeed, tests have demonstrated the running time efficiency of our methods. The calculations by surface patches instead of point-wise operations reduces the complexity of the normals estimation. Whereas the implementation of our anisotropic subdivisions is mono-thread, the CloudCompare normals estimation methods are multi-thread processes taking advantage of the eight cores available on the computer used during the tests. Thus, using thread partitioning libraries could even increase the benefit in terms of running time. Besides normal directions, our methods also provide a curvature indicator unavailable with the other tested pieces of software.

Comparing our methods on six data sets with different characteristics (number of points, sampling density, and type of model) provided sufficient content for evaluation. Our two methods differ in terms of running time efficiency and accuracy. Whereas the *RMSE* octree performs slower than the σ_3 octree, it generates errors in the range of the Meshlab and CloudCompare methods. The σ_3 octree subdivision, however, produces larger errors, mainly due to its inability to capture details and sharp features. Nonetheless, the σ_3 octree reached the same accuracy as other methods when it was applied on

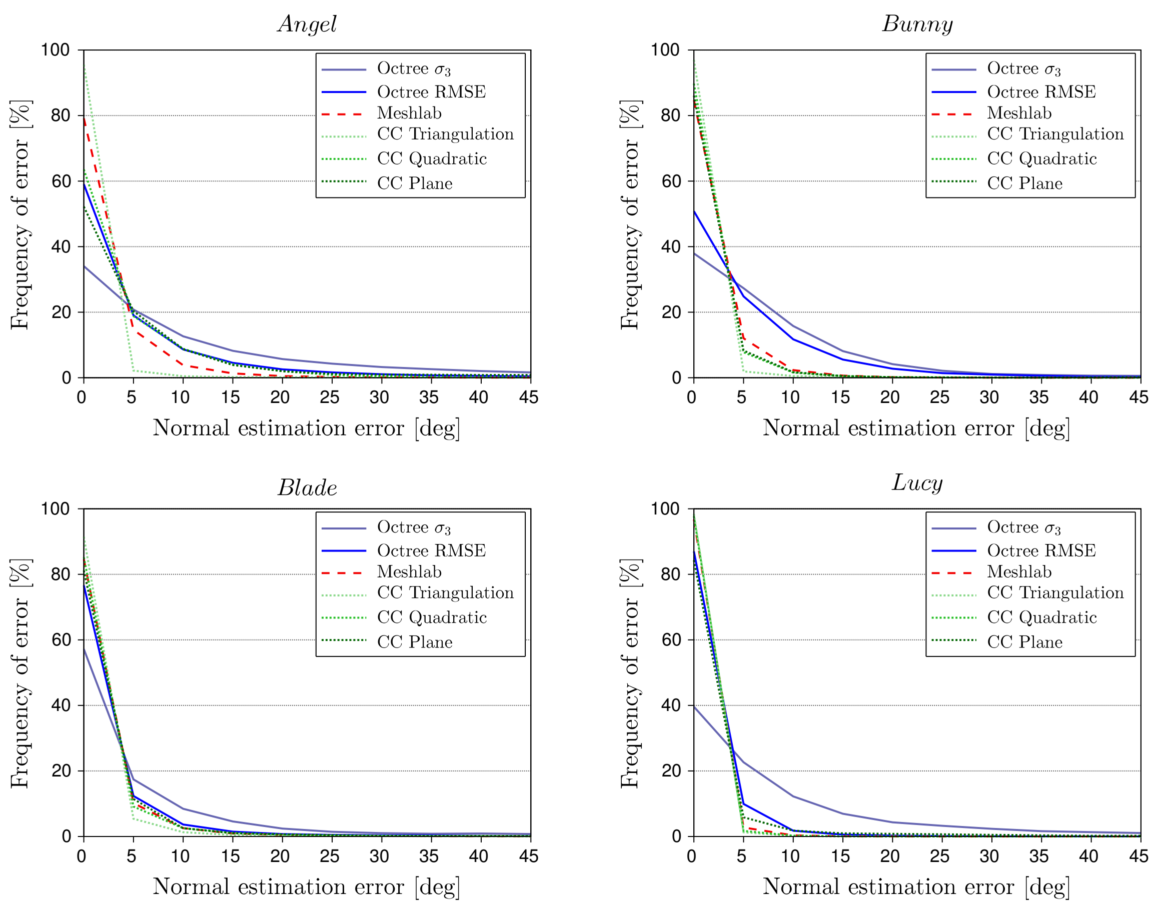

Figure 9: Distribution of error angle between original model normals and estimated normals for different methods. Above 45 degrees, the frequency continues to decrease for each method.

a point cloud with a majority of flat surfaces. Furthermore, patches identified as *edge* cannot be modelled by a surface. Consequently, normals are not estimated on such points, which, in turn, impacts the surface orientation along neighbouring patches.

The impact of noise on the resulting normals has not been studied through an extensive validation yet and is the next step towards a deeper understanding of the capacities of the proposed normal estimation. Robustness to noise was not the main motivation of the presented developments, rather we focused on the balance between computing time and accuracy, as stated earlier (other existing methods were developed to overcome the problem of noisy data). However, we provide some elements to analyse the potential response of our methods. First, given a point cloud affected by a certain level of noise points, processing larger patches (larger than point-wise neighbourhoods) improves surface fitting. This in turn should improve the robustness of normal estimation (let us point out that thanks to these larger patches, we could also detect and remove outliers during surface fitting to mitigate their impact). When considering a spherical neighbourhood, the chosen radius has to be set carefully: it should be higher than the distance from noisy points to the underlying surface. Otherwise, the local dispersion of the cloud is too important to obtain accurate information. The same rule applies to the consideration of the *k*-closest neighbours, which generally implicitly engenders a small neighbourhood radius. By contrast, the subdivision into patches allows to analyse a noisy surface over larger areas. In such case, the noise dispersion in less likely to amalgamate noise and pertinent information. Overall, the chosen scale of analysis during the surface fitting is a critical aspect that has to be carefully considered in optimisation-based approaches.

Our work opens up several perspectives in terms of performance improvements and new applications. A way to improve the efficiency of our method is to take advantage of both the σ_3 and *RMSE* criteria to obtain a compromise between efficiency and estimation accuracy. We believe this could be achieved by using a criterion based on the value of λ_3 rather than the σ_3 indicator. We also intend to explore the relevance of trilinear interpolation during normals blending and apply it

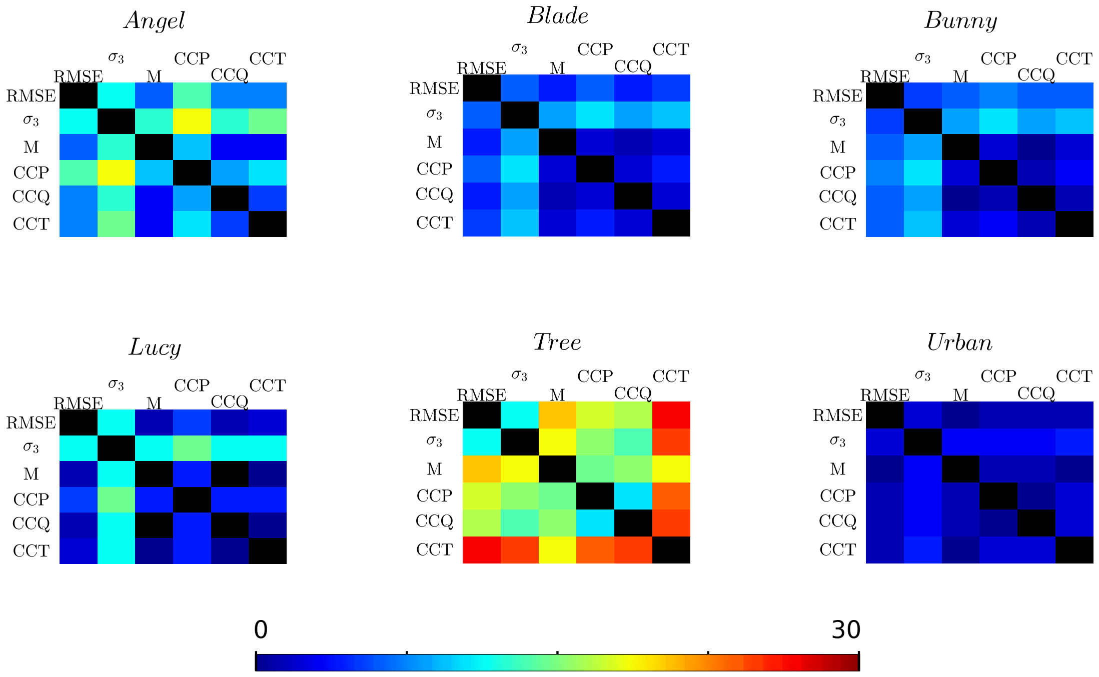

Figure 10: Difference of normal directions estimation in degres between the *RMSE* octree (*RMSE*), the σ_3 octree (σ_3), Meshlab (M) and CloudCompare's plane fitting (CCP), quadratic fitting (CCQ) and triangulation (CCT).

to estimate normals on currently ignored patches. Octree structures are a major element in computer graphics and they take part in many algorithms in order to accelerate the processing. KD-trees are also a popular space partitioning structure. We chose to use octrees because of the regularity of the subdivision (the space is divided into cubes) and because of the efficiency of octrees for neighbourhood searches; both of these properties are further required in other workflows. However, our method could be adapted to KD-trees with few changes. This might even enhance the anisotropy of the resulting procedure since patches could be split at regions of maximum estimated curvature. Doing so would be highly relevant to obtain the larger uniform patches. Another improvement would be to develop these changes in a multi-thread implementation. Additionally, the method could be adapted to handle out-of-core point clouds containing up to several tens of billions of points. Indeed, once patches have been identified and approximated with a quadratic surface, the data points are no longer needed. Thus, the only memory footprint remaining is that of the octree structure itself, which is manageable. Therefore, loading separately appropriate chunks of the point cloud is a viable solution that do not interfere with the normal orientation procedure. Moreover, out-of-core clouds usually represent wide areas (e.g. an entire and detailed building, or parts of a city). With these data, it would be recommended to impose a maximum patch size prior to the octree creation. For example, patches of 15 m^2 would have a great chance of containing sharp features with a junctions of different surfaces such as wall-ground. This patch size limit would fasten the procedure by avoiding several subdivisions and make it easier to handle the amount of data points. Finally we are currently working on a point cloud compression method based on the presented octrees. This lossy compression represents a highly efficient way to store data and could be used for fast computing of levels of details.

5 CONCLUSION

We have presented two new methods to estimate normal directions and curvatures on a point cloud which are major elements in geometric analysis. Both rely on an efficient anisotropic octree subdivision of the data and perform faster than the benchmark pieces of software tested. Results show that the *RMSE* criterion is preferable for capturing great details on the point cloud, whereas the usage of the faster σ_3 criterion has to be restricted to point clouds containing mostly flat surfaces (e.g. urban or indoor environments).

The generated anisotropic octrees represent a multi-purpose and efficient tool for point cloud handling. Indeed, besides the normal and curvature estimation presented here, octrees are included in a wide range of applications in computer graphics. As an example they are almost inevitable for visualisation and neighbours finding, and may be involved in segmentation, classification, and spatial analysis. Overall, the proposed methodology can be inserted in various workflows with few modifications and gives access to elaborated methods based on normals directions.

6 REFERENCES

[Am99] Amenta, N. and Bern, M. Surface reconstruction by Voronoi filtering, Discrete & Computational Geometry, pp.481-504, 1999.

[Bo12] Boulch, A. and Marlet, R. Fast and robust normal estimation for point clouds with sharp features, Computer Graphics Forum, vol. 31, no. 5, pp.1765-1774, 2012.

[Bo16] Boulch, A. and Marlet, R. Deep learning for robust normal estimation in unstructured point clouds, Computer Graphics Forum, vol. 35, no. 5, pp.281-290, 2016.

[de05] Dey, T.K., Li, G. and Sun J. Normal Estimation for Point Clouds: A Comparison Study for a Voronoi Based Method, in Conf.proc. PBG'05, New York, IEEE, pp.39-46

[Gu01] Gumhold, S., Wang, X. and MacLeod, R. Feature extraction from point clouds, in Conf.proc. IMR'01, Newport Beach

[Ho92] Hoppe, H., DeRose, T. and Duchamp, T. Surface reconstruction from unorganized points, in Conf.proc. SIGGRAPH'92, New York, ACM

[Ji05] Jin, S., Lewis, R.R. and West, D. A comparison of algorithms for vertex normal computation, The visual computer, vol. 21, no. 1, pp. 71-82, 2005

[Kl09] Klassing, K., Althoff, D., Wollherr, D., and Buss, M. Comparison of surface normal estimation methods for range sensing applications, in Conf.proc. ICRA'09, Kobe, IEEE

[Li08] Liu, Y., and Xiong, Y. Automatic segmentation of unorganized noisy point clouds based on the Gaussian map, Computer-Aided Design, vol. 40, no. 5, pp. 576-594, 2008

[Me11] Merigot, Q., Ovsjanikov, M., and Guibas, L.J. Voronoi-based curvature and feature estimation from point clouds, IEEE Transactions on Visualization and Computer Graphics, vol. 17, no. 6, pp. 743-756, 2011

[Pa02] Pauly, M., Gross, M., and Kobbelt, L.P. Efficient simplification of point-sampled surfaces, IEEE Transactions on Visualization and Computer Graphics, in Proc.conf. Visualization'02, Boston, IEEE, 2002

[Zh16] Zhao, H., Yuan, D., Zhu, H. and Yin, J. 3-D point cloud normal estimation based on fitting algebraic spheres, in Conf.proc. ICIP'16, Phoenix, IEEE 2016

[Pr93] Press, H., W., Teukolsky, S., A., Vetterling, W., T., and Flannery, B., P. Numerical Recipes in C: The Art of Scientific Computing. Cambridge University Press, 1993