Eurographics Symposium on Geometry Processing 2012
Eitan Grinspun and Niloy Mitra
(Guest Editors)

Volume 31 (2012), Number 5

Fast and Robust Normal Estimation for Point Clouds with Sharp Features

Alexandre Boulch[1] and Renaud Marlet[1]

[1] Université Paris-Est, LIGM (UMR CNRS), Center for Visual Computing, Ecole des Ponts ParisTech
6-8 av. Blaise Pascal, 77455 Marne-la-Vallée, France

Abstract

This paper presents a new method for estimating normals on unorganized point clouds that preserves sharp features. It is based on a robust version of the Randomized Hough Transform (RHT). We consider the filled Hough transform accumulator as an image of the discrete probability distribution of possible normals. The normals we estimate corresponds to the maximum of this distribution. We use a fixed-size accumulator for speed, statistical exploration bounds for robustness, and randomized accumulators to prevent discretization effects. We also propose various sampling strategies to deal with anisotropy, as produced by laser scans due to differences of incidence. Our experiments show that our approach offers an ideal compromise between precision, speed, and robustness: it is at least as precise and noise-resistant as state-of-the-art methods that preserve sharp features, while being almost an order of magnitude faster. Besides, it can handle anisotropy with minor speed and precision losses.

Categories and Subject Descriptors (according to ACM CCS): I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—

1. Introduction

Numerous algorithms rely on the quality of normal estimation in point clouds, such as point-based rendering [RL00], surface reconstruction [ÖGG09], 3D piecewise-planar reconstruction [CLP10] and primitive extraction [SWK07].

For this, regression methods are the most common. Hoppe et al. [HDD*92] estimate normals approximating a tangent plane with a regression that is computed efficiently by principal component analysis (PCA). Other surfaces have been used too, e.g., spheres [GG07] or jets (a truncated Taylor expansion of a surface expression) such as quadrics [CP03]. Regression methods are robust to noise, although they can be improved in that respect with adaptive neighborhood sizes [MNG04], but they are sensitive to outliers. More recent work handles both noise and outliers [GG07, HLZ*09, YLL*07]. However, all regression-based techniques tend to smooth sharp features, and thus fail to correctly estimate normals near edges (see Figure 1). The estimation quality also depends a lot on the size of the neighborhood used for regression: larger neighborhoods are needed to deal with noise, but they make sharp features even smoother.

Another class of methods is based on a preliminary normal estimation, which is improved. Algorithms such as Moving Least Squares (MLS) [ABCO*01], adaptive versions [PKKG03], or robust Local Kernel Regression (LKR) [ÖGG09] compute an implicit surface and estimate normals as the gradient of the surface. They can retrieve sharp features, but they depend on a reliable prior estimation of input normals. Bilateral filtering [JDZ04] also preserves sharp features while smoothing even regions, but it can be slow and the quality relies on that of input normals too.

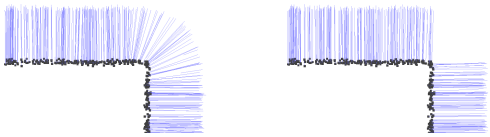


Figure 1: *Reconstructed normals of a corner with Least Square Regression (left) and our method (right).*

Other approaches directly estimate normals with a method that does not oversmooth edges. Dey's method [DG06] relies on the construction of a Voronoï diagram and the search of the furthest vertex of the Voronoï cell. This method preserves sharp features and can deal with density variation. But it is quite sensitive to noise (and does not handle sur-

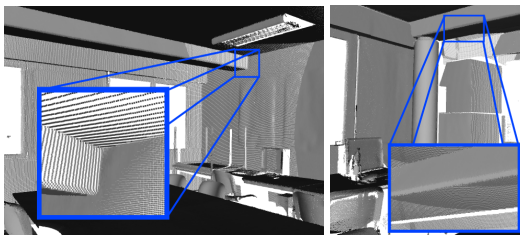


Figure 2: *Laser scan of room, highlights of a corner with density anisotropy and a beveled edge.*

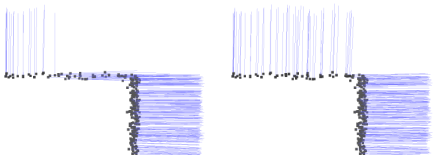


Figure 3: *Reconstructed normals of a corner with sharp density variation: Li et al.'s algorithm (left) and ours (right).*

face boundaries). Alliez et al. [ACSTD07] address this issue with a Voronoï-PCA method that provides some control over smoothness. More recently, both noise and sharp features have been treated explicitly by Li et al. [LSK*10], combining a robust local noise estimation and a RANSAC-like method that is parameterized by the estimated noise scale. It handles well noise and outliers. But it is not very fast (typically around half an hour for 1.5 million points). Besides, it does not address variation of density at edges.

Yet, sampling anisotropy is common in laser data, especially when scanning objects with access constraints or abrupt variations, such as buildings. First, the scanning device may not space samples evenly. For instance, lasers with a rotating head that are used to scan their surroundings systematically oversample the upper polar region (vertical direction), compared to the equatorial band (horizontal directions), typically with a factor of one or two orders of magnitude. (The lower pole is generally occluded by the tripod support.) Second, even when device sampling is mostly uniformly spaced, e.g., locally when focusing only on a small area, the actual 3D spacing of the sampled data depends on the incidence of the laser on the surface. As man-made objects often present sharp features, a significant local anisotropy frequently appears in areas around edges. But even a smooth object shows sampling variations when the laser beam is almost tangent to the surface. This is illustrated in Figure 2 (left). Moreover, variations of density can also appear in point clouds originating from photogrammetry, even on smooth surfaces, because of reconstruction errors and imprecision, e.g., due to specular or textureless surfaces. (In our experiments with lasers and photogrammetry, we observed that noise was not isotropic either; it depends on the surface, in particular on the viewpoint incidence.) Yet, most normal estimation methods do not have any specific treatment of anisotropy. This is illustrated in Figure 3, where a variation of point density in two planes creates estimation errors in the low-density area where they intersect.

We present a new method for estimating normals that addresses the requirements of real data: sensitivity to sharp features, robustness to noise, to outliers, and to sampling anisotropy, as well as computational speed to achieve scalability. This method is based on a fast and robust adaptation of the Randomized Hough Transform (RHT).

The Hough transform [Hou62] has been originally introduced to detect lines and arcs in bubble chamber pictures. Duda and Hart [DH72] discretize the space and introduced accumulators. The main idea of this transform is to change the data representation space such that the desired shape accumulates in a way that is easy to detect [IK88]. It has been successfully used in two dimensions to detect other primitives such as ellipses [TM78] or corners [Dav88]. With the generalized Hough transform (GHT) [Bal87], Ballard shows that the method can also be applied to detect non analytic features in images. The Hough transform has been applied in many ways for detection and classification [Low04, GL09, BLK10, Oka09]. It generalizes to higher dimensions and has been used in particular for 3D segmentation, recognition and registration [KPW*10, PWP*11]. The general algorithm has also been modified and improved for speed, robustness and precision. As going through all the data may take too much time, Kiryati et al. [KEB91] propose a probabilistic version, consisting in observing only subsets. Xu et al. [XOK90, XO93] define a Randomized Hough Transform, where a point does not vote for all the primitives to which it belongs; the vote is associated to a primitive computed from a subset of points. The criterion to stop picking more primitives is a user-defined, global condition.

As our objective is the estimation of a single normal at each point, we adapt the Randomized Hough Transform to search for only one primitive. One original aspect of our method is that we use a stop criterion inherited from robust statistics to end exploring the space of primitives. This ensures both speed and robustness to partial sampling. Robustness to density anisotropy is obtained by selecting primitives as uniformly as possible in the neighborhood of the considered point. We do not address the problem of orientating normals, which can be done separately [HDD*92, MdGD*10]. In the following, we first give a general overview of our normal estimator (cf. §2) and describe our Robust Randomized Hough Transform (cf. §3). Then we explain how to compute the normals despite discretization (cf. §4) and anisotropy (cf. §5), and finally present our results (cf. §6).

2. Algorithm

Before describing our algorithm, let us consider the following simple situation. Let P be a point on a piecewise planar surface and let $\mathcal{N}_P$ be a neighborhood of P on this surface (a subsurface). There are two basic cases.

- If P lies far from any edge or sharp feature, then picking three points in $\mathcal{N}_P$ defines the planar patch that P lies on, and thus the normal (if the points are not collinear).
- If P lies near an edge partitioning the neighborhood $\mathcal{N}_P$ into $\mathcal{N}_{1,P} \cup \mathcal{N}_{2,P}$ with $P \in \mathcal{N}_{1,P}$, then picking three points in $\mathcal{N}_P$ does not necessarily determine the right normal. It defines either the correct normal (if all points lie in $\mathcal{N}_{1,P}$), or the normal associated with the plane on the opposite side of the edge (if all points lie in $\mathcal{N}_{2,P}$), or a "random" plane (if the points are not on the same side of the edge).

In the second case, as $\mathcal{N}_{1,P}$ is likely to be larger than $\mathcal{N}_{2,P}$ since $P \in \mathcal{N}_{1,P}$ is not exactly on the edge, the probability of picking the dominant plane and thus the correct normal is higher than the probability of picking the normal on the opposite edge side. When P is very close to the edge, drawn triples are likely to lie on both sides of the edge. However, as it leads to a "random" normal, the correct normal still is the one with the highest probability. This generalizes to situations where P is close to several edges, including coincident edges. This idea applies as well to a point cloud $\mathcal{C}$ in which neighboring points $\mathcal{N}_P$ are defined for any point $P \in \mathcal{C}$.

Our method is a robust variant of this simple principle, to handle noise and outliers. For this, we sample as many planes as necessary to gain enough confidence that we can identify the actual maximum of the probability density. In practice, we discretize the problem and fill a Hough accumulator until a normal can be confidently chosen based on the most voted bin, with the following refinements:

- In the presence of noise and outliers, the discrete probability distribution of possible normals is flatter, but the principal normal remains the one with the highest number of votes. To ensure resistance to noise and outliers, we use robust statistical bounds on the number of triples to pick (cf. §3). Note that near collinearity need not be checked because it is unlikely and would generate a "random" normal anyway. On the contrary, checking collinearity at each drawing would be unnecessarily time consuming.
- If the surface is curved rather than piecewise planar, the discrete probability distribution of possible normals is flatter too. However, as long as the surface can be locally approximated with planes, the right normal stays the most voted one. The size of the neighborhood impacts normal estimation: it should be small enough for the planar approximation hypothesis to hold, and large enough for noise to be averaged. (Neighborhoods are discussed in Section 5, together with anisotropy.)
- Using a Hough accumulator introduces discretization artifacts. First, a single bin corresponds to a small range of normals (a cone). Rather than associating a fixed normal to a bin, we average the normals that vote for the bin. Second, we actually estimate the normal several times using randomized accumulators and use the best ones (cf. §4).
- To deal with density anisotropy, we propose a spatially-sensitive drawing scheme that comes in two versions, continuous and precise, or discrete and fast (cf. §5).

3. Robust Randomized Hough Transform

The number of possible triples to pick can be huge, on the order of $|\mathcal{N}_P|^3$. We thus want to consider only a subset, and we want to stop sampling triples as soon as we are confident enough that we can take a decision based on the empirical distribution in the accumulator. Using robust statistics tools, we determine a general upper bound on the number of triples to draw, and possibly improve it when enough concentration in a bin can be guaranteed. This defines a Robust Randomized Hough Transform (RRHT) for estimating normals.

In the following, we consider that the M bins of the accumulator follow a Bernoulli law with p_m as parameter (the theoretical mean). Let T be the number of planes to pick and let $(X_{m,t})_{m\in\{1,\dots,M\},t\in\{1,\dots,T\}}$ be the independent and identically distributed random variables associated with bin m for plane t: $X_{m,t} = 1$ if plane t votes for bin m, otherwise $X_{m,t} = 0$. Finally, we note $\hat{p}_m$ the empirical mean of the proportion of votes for bin m: $\hat{p}_m = \frac{1}{T}\sum_{t=1}^{T} X_{m,t}$.

Global upper bound. We want to stop drawing triples as soon as we are confident enough that the empirical distribution is a good approximation of the actual distribution. For this, we want to bound the difference between the actual distribution p_m and the observed distribution $\hat{p}_m$.

Let $\alpha \in]0,1[$ be a probability threshold expressing the confidence when comparing p_m to $\hat{p}_m$, and let $\delta \in]0,1[$ be a distance. We wish to estimate the minimal number of samples $T_{\min}$ such that, for each bin m, the empirical mean $\hat{p}_m$ is at most at distance δ from p_m, with probability at least α:

$$\mathbb{P}(\max_{m\in\{1,\dots,M\}} |\hat{p}_m - p_m| \leq \delta) \geq \alpha \tag{1}$$

As $X_{m,t}$ are i.i.d., Hoeffding's inequality [Hoe63] applies:

$$\forall m,\ \mathbb{P}(|\hat{p}_m - p_m| \geq \delta) \leq 2\exp\left(-\frac{2\delta^2 T_{\min}^2}{\sum_{t=1}^{T}(b_t - a_t)^2}\right) \tag{2}$$

where $[a_t, b_t]$ is the interval where $X_{m,t}$ lies, i.e., $[0,1]$. Thus:

$$\mathbb{P}(|\hat{p}_m - p_m| \geq \delta) \leq 2\exp(-2\delta^2 T_{\min}) \tag{3}$$

As this is true for all m, we have:

$$\begin{aligned}
&\mathbb{P}(\max_{m\in\{1,\dots,M\}} |\hat{p}_m - p_m| \geq \delta) \\
&\quad = \mathbb{P}(\exists m \in \{1,\dots,M\}, |\hat{p}_m - p_m| \geq \delta) &(4)\\
&\quad \leq \sum_{m=1}^{M} \mathbb{P}(|\hat{p}_m - p_m| \geq \delta) \ \leq\ 2M\exp(-2\delta^2 T_{\min}) &(5)
\end{aligned}$$

Hence:

$$\mathbb{P}(\max_{m\in\{1,\dots,M\}} |\hat{p}_m - p_m| < \delta) \geq 1 - 2M\exp(-2\delta^2 T_{\min}) \tag{6}$$

Now, considering equation (1), $T_{\min}$ has to satisfy:

$$T_{\min} \geq \frac{1}{2\delta^2}\ln\left(\frac{2M}{1-\alpha}\right) \tag{7}$$

We may thus choose $T_R = \frac{1}{2\delta^2}\ln(\frac{2M}{1-\alpha})$ as an upper bound on the number of triples to be drawn.

Confidence interval. If we pick most of the time the same bin, we want to stop the selection operation. We use a confidence interval to identify it as the right bin to choose. But it is actually enough to choose a bin that gets significantly more votes than others, i.e., more votes than the second most voted bin. The above criterion only concerns the accuracy of the global distribution. What we want to estimate here is the confidence interval of each p_m. If the intervals of the two most voted bins do not intersect, we are almost sure that the most voted bin will not change and we can stop picking more triples.

According to the Central Limit Theorem, the random variable defined as

$$\frac{\hat{p}_m - p_m}{\sqrt{p_m(p_m-1)/T}}$$

converges in distribution to a standard normal random variable $\mathcal{N}(0,1)$. Let $r \in]0,1[$ be a confidence level and let $z(r)$ be such that the integral of the Gaussian density between $-z$ and z is r. The confidence interval is then:

$$\hat{p}_m - z(r)\sqrt{\frac{p_m(1-p_m)}{T}} \leq p_m \leq \hat{p}_m + z(r)\sqrt{\frac{p_m(1-p_m)}{T}} \tag{8}$$

As $p_m \in [0,1]$, we have $p_m(1-p_m) \leq \frac{1}{4}$ and thus:

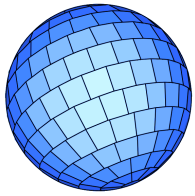
$$\hat{p}_m - \frac{z(r)}{2}\sqrt{\frac{1}{T}} \leq p_m \leq \hat{p}_m + \frac{z(r)}{2}\sqrt{\frac{1}{T}} \tag{9}$$

Using, e.g., the confidence level $r = 95\%$, we have $z(r) \simeq 2$. In this context, if m_1 is the most voted bin and m_2 is the second most voted, we can stop sampling planes as soon as the confidence intervals of these two bins do not intersect:

$$\hat{p}_{m_1} - \hat{p}_{m_2} \geq 2\sqrt{\frac{1}{T}} \tag{10}$$

This test may lower the global bound T_R but does not replace it (cf. §6). E.g., for a point lying very close to an edge, the two bins corresponding to the normal on each edge side are likely to have similar probabilities. Confidence intervals then require a large T before (10) is satisfied. In extreme cases, intervals always overlap and the condition is never fulfilled.

Accumulator shape. As we only estimate the normal direction, not the orientation, picking a triple of points defines a normal described by two angles (θ, ϕ), modulo π. It votes into an accumulator that partitions half the unit sphere into similar bins. We use the spherical accumulator of Borrmann et al. [BELN11] that provides bins with similar area and allows easy and fast computation of bin indexes, given the normal angles. This accumulator first divides the sphere according to the parallels, with same angle size. Then the slices (between two parallels) are divided into bins of nearly the same area (cf. Fig. 4). We have chosen this accumulator after also testing a geodesic sphere accumulator, that is more isotropic but considerably slower for a limited precision improvement.



n_ϕ	M
5	23
10	82
15	171
20	290
25	441

Figure 4: *Accumulator shape & size for various n_ϕ values.*

With reference to Borrmann et al.'s paper, we define n_ϕ as the number of slices of the sphere in the z-axis and $n_\theta = 2n_\phi$ the number of bins at the equator. As we only estimate here the normal direction, not the orientation, we use half of the bins in the accumulator. Figure 4 shows the value of the total number of used bins M for several values of n_ϕ:

4. Discretization issues

After plane sampling, the chosen normal is given by the most voted bin. However, rather than producing a discrete normal (one per bin in the accumulator), we average all the normals that voted for the chosen bin. From the implementation point of view, sampled normals do not actually have to be memorized. It is enough to only record incrementally, for each bin, the sum of the normals that voted for the bin. When a bin is chosen, the sum of its contributing normals is renormalized to lie on the unit sphere, yielding the final normal. (We have also experimented memorizing for each bin the voting triples of points, and computing the final normal as the regression plane of the points in the most voted bin: the normal precision is slightly better, but the running time is much longer.)

Using a discrete accumulator leads to other issues. First, there is a binning effect. If the main peak of the distribution of normals lies near a bin boundary, votes will be almost equally distributed between two (or more) adjacent bins. After a limited number of plane pickings, the actual peak may not be in the most voted bin. The effect is stronger for noisy data as the distribution is flatter and votes are distributed into more bins, that receive less votes on average. Second, bins are not isotropic. The accumulator of Borrmann et al. [BELN11] guarantees the area of all bins to be nearly equal, but their shape is different. For instance, polar bins are disks (caps) whereas equatorial bins are squares. A classic solution to bin discretization would be for a normal to share parts of its vote in neighboring bins, but it does not answer the second problem. Both issues can be addressed by randomly rotating the accumulator and running the algorithm several times. The more rotations, the more precise the estimation. In practice, as few as 5 rotations are enough to compensate for most discretization effects (cf. §6). From the implementation point of view, it is more efficient to rotate the points, rather than rotating the accumulator itself.

As we run the algorithm several times, we get several nor-

mals. Choosing the appropriate one depends of the sampled surface and on applications. We propose three alternatives:

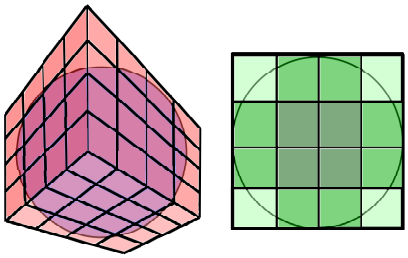
- **RRHT_m**: the produced normal is the mean of the normals weighted by their number of votes. Compared to the other alternatives, it minimizes the root mean square error, but it fails to estimate a plausible normal near edges because of the smoothing effect of the mean.
- **RRHT_b**: the produced normal is the best one, i.e., the most voted one. Sharp features are well estimated, but smooth surfaces may appear grainy.
- **RRHT_c**: we first cluster normals that are closed to each other (within an angle threshold a_{cluster}) and then compute the average normal of the most voted cluster. Although it introduces an extra parameter, it is a good compromise between the other two alternatives.

5. Dealing with sampling anisotropy

Triple selection is the key to robustness to density variation. Given a point P and its neighborhood $\mathcal{N}_P$, selecting random triples in the neighborhood would be sensitive to sampling density: triples would be selected with a higher probability in regions of high density. We define three variants.

For robustness to density variation, we define $\mathcal{N}_P$ as the points in a ball $\mathcal{B}_P$ around P with a given radius r and choose a presampling factor c. To pick a point in $\mathcal{N}_P$, we randomly pick a small ball B of radius r/c in $\mathcal{B}_P$, then pick a random point in B. If B is empty (no intersection with the sampled surface), we just pick another small ball until we find one which is not empty. As shown below, this strategy, noted **RRHT_Unif**, gives good results but it is relatively slow.

As we are interested in a good compromise between precision and speed, we propose a discretized version of this spatially-sensitive drawing scheme. We discretized the ball into small cubes and we randomly pick small cubes rather than small balls. This only requires fast operations on Cartesian coordinates rather than heavier trigonometric computations. Yet, as all small cubes are not fully included in the $\mathcal{B}_P$ ball, we assign them weights according to the proportion of their intersection with $\mathcal{B}_P$ (cf. Fig. 5). These weights are used to define the probability of picking any given small cube.



Discretization of the sphere for $c = 4$ (left) and small cube probabilities (right): darker is more probable.

Figure 5: *Discretization of the neighborhood ball.*

Let c be the cube discretization factor: $\mathcal{B}_P$ is covered by c^3 small cubes. The bigger c, the more uniform the picking, but the higher the probability that the small cube is empty too. In our experiments with this strategy, noted **RRHT_Cubes**, $c = 4$ provides a good robustness to anisotropy without slowing down too much point picking. (Note that these methods for picking triples are actually not specific to our normal estimator. They could be applied to any algorithm using a random triple selection, like Li et al.'s estimator.)

We also consider a simple and fast version of our algorithm, noted **RRHT_Points**, without any support for anisotropic sampling. It applies to point clouds without significant density variation: points are just picked randomly in $\mathcal{N}_P$. More precisely, we use a costless combinatorial order to make sure that we do not pick several times the same triple, which could be likely for a small $\mathcal{N}_P$.

6. Experiments

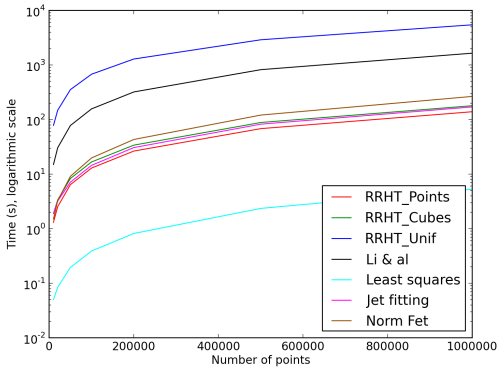
The parameters of our algorithm are summarized below:

- K or r: number of neighbors or neighborhood radius,
- T_R: number of primitives to explore,
- n_ϕ: parameter defining the number of bins,
- n_{rot}: number of accumulator rotations,
- c: presampling or discretization factor (anisotropy only),
- a_{cluster}: tolerance angle (mean over best cluster only).

The choice of K or r depends on the surface sampling density and noise, w.r.t. the level of details. Parameters T_R and n_{rot} can be used to balance precision and execution time. Unless otherwise mentioned, in all our experiments we take $T_R = 700$, $n_\phi = 15$, $n_{\text{rot}} = 5$, $c = 4$ and $a_{\text{cluster}} = \frac{\pi}{4}$. In this settings, confidence $\alpha = 0.95$ corresponds to $\delta = 0.08$ for the global upper bound on the number of drawings. Finally, on synthetic data we always use $K = 500$; on real data we use either $K = 500$ or a neighborhood defined by a ball.

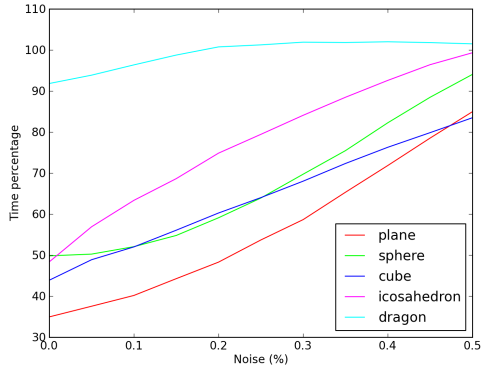
We compare our algorithm with some of the existing ones: least square regression with planes [HDD*92] implemented in the Point Cloud Library [PCL], jet fitting [CP03] from CGAL [CGA], Tamal Dey's NormFet implementation [DG06] and Li et al.'s algorithm [LSK*10]. These are "direct" normal estimators. No postprocessing is considered here. In the following, we always use the same parameters for these methods: $K = 80$ for regression and jet fitting (as it is most sensitive to neighborhood size), $K = 500$ for Li et al.'s algorithm. All other parameters, if any, are set to default. We evaluate precision and speed on data with artificial noise: a centered Gaussian noise with deviation defined as a percentage of the diagonal of the axis-aligned bounding box.

Computation time. All experiments have been performed on the same computer, with 2 CPUs Intel(R) Xeon(R) X5472 3.00GHz, 4 threads each. Our algorithms are parallelized. Others have been used as we got them. Only Li et al.'s was adapted to use our data structure, taken from PCL [PCL].



Computation time for a point cloud on sphere with 0.2% noise as a function of the number of points.

Figure 6: *Computation time (global upper bound only).*



Ratio of the execution time with and without the confidence interval criterion as a function of noise level.

Figure 7: *Impact of the confidence interval criterion.*

To illustrate separately the different contributions, we first display the computation time with the robust global upper bound only (cf. §3), disabling the confidence interval criterion. Figure 6 shows the computation time as a function of the number of points. The point cloud has been uniformly sampled on a sphere, we added 0.2% noise. As we can see, even with five rotations, the two fast versions of our algorithm (RRHT_Cubes and RRHT_Points) are comparable to jet fitting and the Voronoï-based method (NormFet), and are faster than Li et al.'s algorithm (with the same neighborhood size). RRHT_Unif is much slower than existing algorithms, but fully handles anisotropy. The complexity is basically the same for our methods and for Li et al.'s: it is $O(n \log n)$ where $n = |\mathcal{C}|$ is the total number of points in the cloud. Both use a Kd-tree for neighborhood search (the $\log n$ factor) and repeat a similar operation for every vertex (the n factor). However, complexity constants differ much in practice. Note that our parameter setting here is very demanding on precision as explained below. Figure 6 is thus a kind of worst case scenario for our method. Much faster computation results are given later with only a slightly reduced precision.

Model (# vertices)	T_R=700 n_{rot}=5 w/o interv.	T_R=700 n_{rot}=5 with interv.	T_R=300 n_{rot}=2 w/o interv.	T_R=300 n_{rot}=2 with interv.
Armadillo (173k)	*21 s*	*20 s*	*3 s*	*3 s*
Dragon (438k)	*55 s*	*51 s*	*8 s*	*7 s*
Buddha (543k)	1.1	1	*10 s*	*10 s*
Circ. Box (701k)	1.5	1.3	*13 s*	*12 s*
Omotondo (998k)	2	1.2	*18 s*	*10 s*
Statuette (5M)	11	10	1.5	1.4
Room (6.6M)	14	8	2.3	1.6
Lucy (14M)	28	17	4	2.5

RRHT_Points algorithm with $K = 100$ points in the neighborhood, with and without confidence intervals. All times are in minutes unless seconds mentioned.

Table 1: *Computation times on real data.*

Figure 7 shows the ratio of the execution times with and without the confidence interval criterion (cf. §3), for various noise levels. (The ratio can be slightly greater than 1 because testing the criterion adds a small overhead that is not repaid if not satisfied.) We can see that the criterion significantly reduces computation. As expected, the impact decreases as noise level grows because the peaks of the distribution are flatter and separating the highest peak from the second highest takes longer. The impact also depends on the model. For curved and complex shapes, it take more time for a peak to confidently emerge from the rest of the distribution. Table 1 shows computation times of RRHT_Points for several real models. As they require less neighbors because they are not noisy, we use $K = 100$. For models with a lot of planar subsurfaces (like the room scan, cf. Figure 2), the addition of the confidence interval criteria is very useful.

Precision. We compare the various algorithms with two error measures:

- Root Mean Square (RMS):

$$RMS = \sqrt{\frac{1}{|\mathcal{C}|} \sum_{P \in \mathcal{C}} \widehat{\mathbf{n}_{P,\text{ref}} \mathbf{n}_{P,\text{est}}}^2}$$

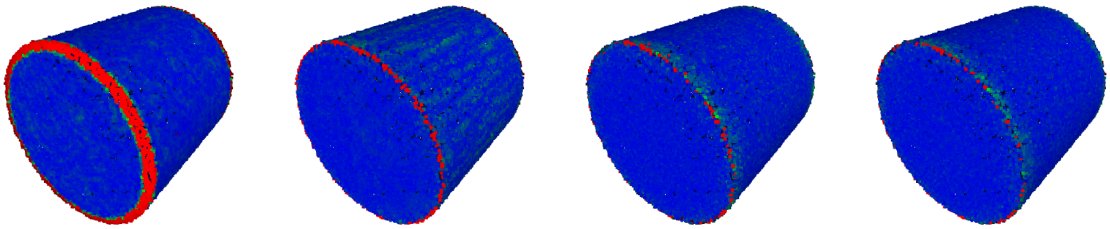
- Root Mean Square with threshold (RMS_τ):

$$RMS\_\tau = \sqrt{\frac{1}{|\mathcal{C}|} \sum_{P \in \mathcal{C}} v_P^2}$$

where

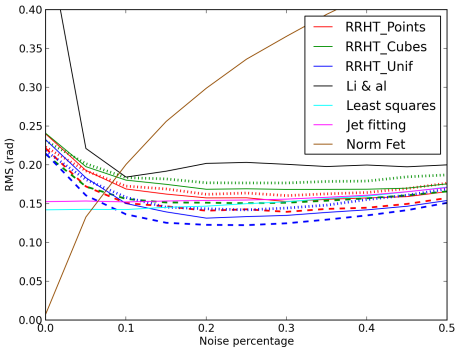
$$v_P = \begin{cases} \widehat{\mathbf{n}_{P,\text{ref}} \mathbf{n}_{P,\text{est}}} & \text{if } \widehat{\mathbf{n}_{P,\text{ref}} \mathbf{n}_{P,\text{est}}} < \tau \\ \frac{\pi}{2} & \text{otherwise} \end{cases}$$

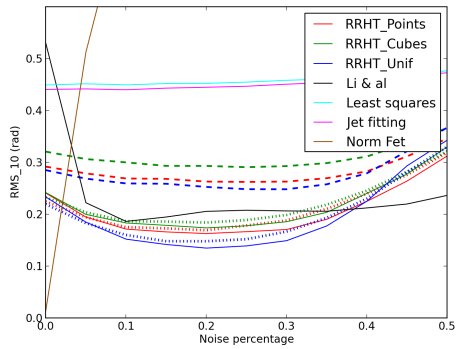
$\mathbf{n}_{P,\text{ref}}$ is the reference (theoretical) normal at P and $\mathbf{n}_{P,\text{est}}$ the estimated one. RMS is a common measure for precision, but it is not really a rendering measure because it favours smooth reconstruction everywhere, including at edges: a smoothed edge is better than having some points with a normal corresponding to the other side of the edge. That is why we



From left to right: Jet fitting, Li et al.'s normal estimator, RRHT_Cubes_c and RRHT_Unif_c. Color scale: blue to green is 0 to 10-degree error, red corresponds to an error greater than 10 degrees.

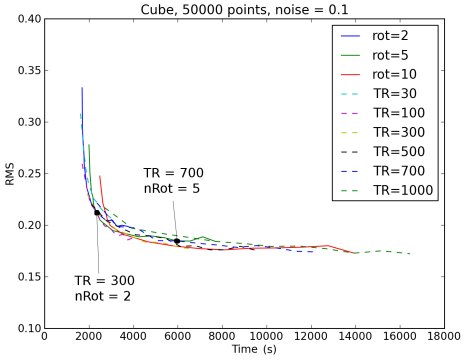
Figure 8: *Visual rendering of the precision for three algorithms on a cylinder of 50000 points with 0.2% noise.*

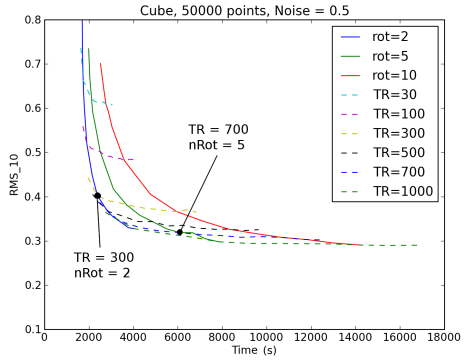




Dashed is RRHT_m (mean), dotted is RRHT_b (best), solid is RRHT_c (mean over best cluster).

Figure 9: *RMS and RMS_10 for 50k-point closed cylinder.*





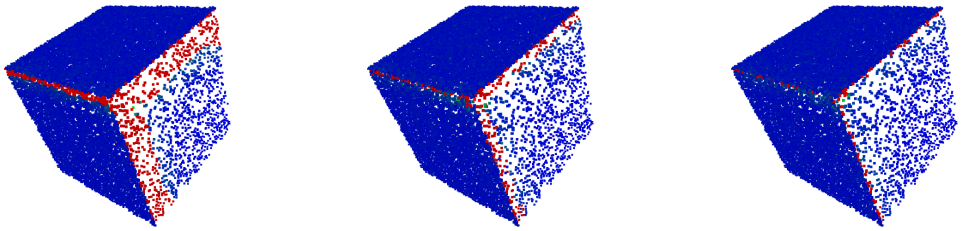
Solid curves represent a fixed n_{rot} for a varying T_R, dashed curves represent a fixed T_R and varying n_{rot}.

Figure 10: *Precision vs speed for a cube with 50k points.*

introduce RMS_τ. This measure considers any angle error above threshold τ as very bad (as bad as $\frac{\pi}{2}$). This penalizes unwanted smoothing, i.e., equally divergent estimations. Point clouds with normals are better rendered with lower RMS_τ than lower RMS. In our examples we take $\tau = 10$ degrees. Figure 8 is a visual representation of the RMS_10 error. Red dots are the badly estimated points. RRHT reconstructs smooth surfaces without grainy effect, but it is not as discriminative as Li et al.'s close to the edge.

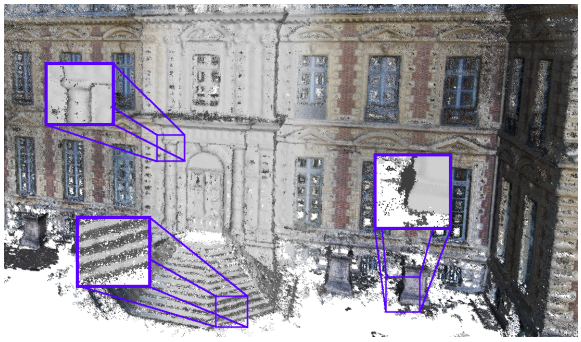
Robustness to noise. Figure 9 displays precision as a function of noise. Our algorithms are competitive with both measures, unlike other methods. As expected, RRHT_m gives better results for RMS, because of its tendency to smooth, but not for RMS_10. RRHT_c offers the better compromise. Note that Dey's algorithm is the most efficient for very small noise. Fig. 12 illustrates an application to data with realistic noise. See also Fig. 7 for the impact of noise on running time.

Parameter (in)sensitivity. Figure 10 represents precision versus computation time, for a fixed n_{rot} or a fixed T_R, and a noisy cube. In our experiments, changing the model or noise does not affect the curve aspect. Two points are marked, $(T_R = 700, n_{\text{rot}} = 5)$ and $(T_R = 300, n_{\text{rot}} = 2)$, which cor-



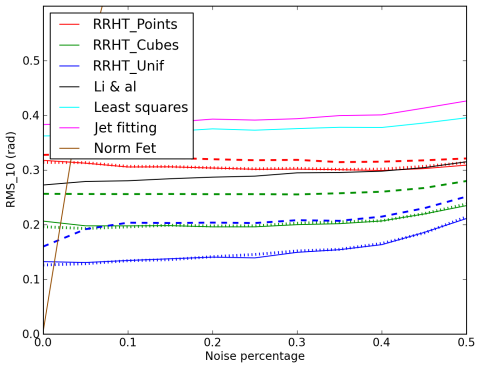
From left to right: RRHT_Points_c, RRHT_Cubes_c and RRHT_Unif_c. Color scale: blue to green is 0 to 10-degree error, red corresponds to an error greater than 10 degrees. Density is uniform on each face; if density is 1 on right face, then it is 5 for the left face and 10 for the upper face.

Figure 11: *Visual rendering of the precision on a corner of 20000 points with density anisotropy.*



RRHT_Cubes_c with a radius search corresponding to 0.1% of the bounding box diagonal.

Figure 12: *Visual rendering of the Château de Sceaux, point cloud obtained by photogrammetry.*



Dashed is RRHT_m (mean), dotted is RRHT_b (best), solid is RRHT_c (mean over best cluster).

Figure 13: *RMS_10 for a corner with density anisotropy of 20000 points (see Figure 11), function of noise percentage.*

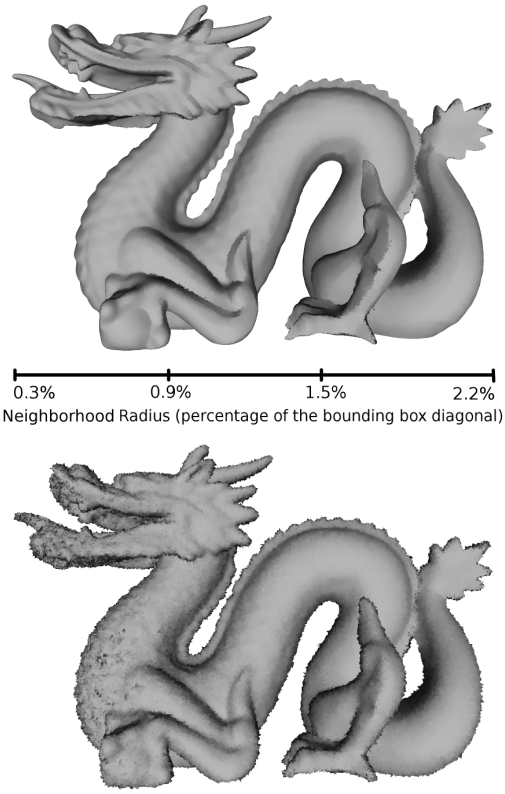
respond to values used in table 1. The first one favors precision; the second one, speed. Other experiments show that time varies little with c (2–2.5 slower when c varies from 2 to 10), nor RMS or RMS_10 (25–30% precision gain). As for n_ϕ, it should be large for precision, small for robustness. Experimentally, best n_ϕ values for both RMS and RMS_10 are in the range 5–25, with $n_\phi = 15$ at most 10% from optimum. Time less than doubles when n_ϕ varies from 5 to 25.

Robustness to density anisotropy. Figure 11 shows the normals computed on a corner sampled with face-specific variations of density. As expected, no support for anisotropy (RRHT_Points) is very bad whereas uniform ball sampling (RRHT_Unif) recovers well the edges. The cubic discretization (RRHT_Cubes) compromises well precision vs time (cf. Fig. 6). Figure 13 shows the RMS_10 error for different algorithms. As expected, we have a better precision than the other methods, that are not designed to deal with anisotropy.

In the above experiments, neighborhoods are defined with a fixed number of points K. This allows comparison with other algorithms because they use the same notion of neighborhood. Also, this data is uniformly sampled (except for the anisotropic corner); a fixed number of points K is thus nearly the same as a fixed ball of radius r. But variations of density in real data make neighborhoods with a fixed K inoperative. E.g., the peak of density at the pole of a laser scan can be such that the actual neighborhood radius r corresponding to a fixed number of points K is on the same order as the noise standard deviation. It is thus too small to cope with noise. Conversely, data sparsity may also lead to mistakes. E.g., the leg of a chair in the room (cf. Fig.2) may be sampled with just a few points and a fixed number of points K will easily include many points on the floor or on sitting area, as opposed to a fixed neighborhood radius r. For this reason, when operating on real data, we use a neighborhood ball.

Figure 14 shows two dragons with estimated normals where the neighborhood radius varies from left to right. As we can see on the model without noise, if the radius is great we tend to loose small details, but general shape, with sharp features, is maintained. The noisy dragon illustrates the fact if the radius is smaller than the noise, it cannot estimate normals, but the results goes better with the increasing radius.

Figure 2 displays parts of a laser scan of a meeting room with a few computers. The density of the point cloud depends a lot of the incidence. The original point cloud has

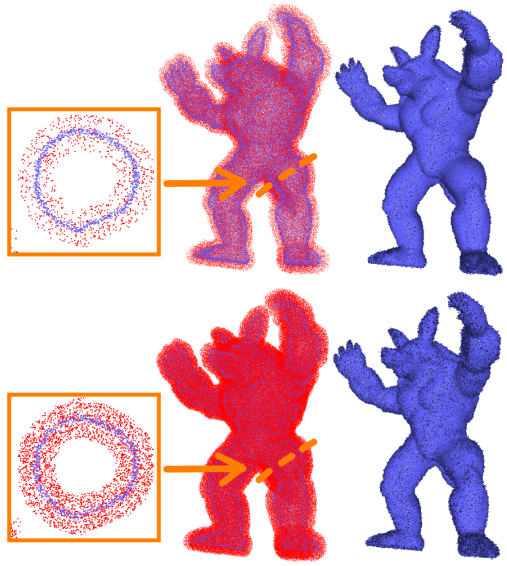


RRHT_Cubes_c with a neighborhood radius ranging from a value comparable to the noise standard deviation, if any (left), to greater radius values (right).

Figure 14: *Normal estimation for the dragon with no added noise (top) vs 0.2% added noise (bottom).*

6.6 millions vertices and with a radius of 0.05 (5 cm at the model scale), the computation time for RRHT_Cubes is 41 min with confidence intervals, as opposed to 48 min without, i.e., a 15% improvement. Two details are underlined: a corner with density anisotropy and a beveled edge. Note that there is a substantial difference with the time figures in Table 1. The reason is that searching for 100 neighbors is much faster than searching within a given radius, which can correspond to thousands of neighbors (or more) when close to the pole. Also, Table 1 uses RRHT_Points, not RRHT_Cubes; not having to pick in the cube is faster.

Robustness to outliers. Our method is robust to a large outlier ratio. We illustrate that on the Armadillo with both added noise and outliers (Figure 15). The RMS on the model with 0.2% noise is 0.39; adding +100% outliers yields an extra +0.04 on the RMS; adding +300% outliers yields +0.15 on the RMS. When the contamination ratio increases, the first points not well estimated are the sharp points, whose normal distribution is flatter than the distribution on a regular surface: they are more sensitive to noise and outliers.



Armadillo with 0.2% noise and +100% outliers (top), +300% (bottom). Neighborhood radius r is 3% of bounding box diagonal. Outliers are uniformly drawn at distance at most r from original point set. Outliers are dropped in result for visual rending (right).

Figure 15: *Outlier robustness of RRHT on Armadillo.*

7. Conclusion

We have proposed a novel method for estimating normals for point clouds that preserves sharp features and that is robust to noise and outliers. Different variants or parameter settings offer good compromises between precision and computation time. We have shown that our method is at least as precise and noise-resistant as state-of-the-art methods that preserve sharp features, while being almost an order of magnitude faster. It can also handle anisotropy with minor speed and precision losses. Besides, it is simple and easy to program.

As future work, it would be interesting to improve speed with an adaptive choice of variants and parameters, and the time saved could in turn be traded against more precision. The idea would be to use cubic drawing by default but to locally switch to the uniform drawing scheme where estimation is known to be inaccurate. Also, an adaptive neighborhood would help reducing the number of parameters. Moreover, some kinds of laser scans organize data; preliminary experiments show that making a good use this organization greatly reduces computation time.

Acknowledgments

We are grateful to Pierre Alliez for helpful advices. Thanks to Bao Li and Tamal Dey for providing the code used for comparison. Armadillo, Lucy, Dragon, Buddha, and Statuette point clouds come from the Stanford 3D Scanning Repository; Circular Box and Omotondo from Aim@Shape.

References

[ABCO*01] ALEXA M., BEHR J., COHEN-OR D., FLEISHMAN S., LEVIN D., SILVA C. T.: Point set surfaces. In *IEEE Visualization* (October 2001), IEEE Comp. Soc., pp. 21–28. 1

[ACSTD07] ALLIEZ P., COHEN-STEINER D., TONG Y., DESBRUN M.: Voronoi-based variational reconstruction of unoriented point sets. In *Proceedings of the Fifth Eurographics Symposium on Geometry Processing, Barcelona, Spain, July 4-6, 2007* (2007), Belyaev A. G., Garland M., (Eds.), vol. 257 of *ACM International Conference Proceeding Series*, Eurographics Association, pp. 39–48. 2

[Bal87] BALLARD D. H.: Generalizing the hough transform to detect arbitrary shapes. In *RCV87* (1987), pp. 714–725. 2

[BELN11] BORRMANN D., ELSEBERG J., LINGEMANN K., NÜCHTER A.: The 3D Hough Transform for plane detection in point clouds: A review and a new accumulator design. *3D Res. 2*, 2 (Mar. 2011), 32:1–32:13. 4

[BLK10] BARINOVA O., LEMPITSKY V. S., KOHLI P.: On detection of multiple object instances using Hough transforms. In *CVPR* (2010), IEEE, pp. 2233–2240. 2

[CGA] CGAL, Computational Geometry Algorithms Library. http://www.cgal.org. 5

[CLP10] CHAUVE A.-L., LABATUT P., PONS J.-P.: Robust piecewise-planar 3D reconstruction and completion from large-scale unstructured point data. In *CVPR* (2010), IEEE, pp. 1261–1268. 1

[CP03] CAZALS F., POUGET M.: Estimating differential quantities using polynomial fitting of osculating jets. In *Proceedings of the 2003 Eurographics/ACM SIGGRAPH symposium on Geometry processing* (2003), SGP '03, pp. 177–187. 1, 5

[Dav88] DAVIES E. R.: Application of the generalized Hough transformation to corner detection. *Computers and Digital Techniques, IEEE proceedings 15*, 1 (1988), 49–54. 2

[DG06] DEY T. K., GOSWAMI S.: Provable surface reconstruction from noisy samples. *Comput. Geom. 35*, 1-2 (2006), 124–141. 1, 5

[DH72] DUDA R., HART P. E.: Use of the Hough transformation to detect lines and curves in pictures. *CACM 15* (1972), 11–15. 2

[GG07] GUENNEBAUD G., GROSS M. H.: Algebraic point set surfaces. *ACM Trans. Graph 26*, 3 (2007), 23. 1

[GL09] GALL J., LEMPITSKY V.: Class-specific hough forests for object detection. In *CVPR* (2009), pp. 1022–1029. 2

[HDD*92] HOPPE H., DEROSE T., DUCHAMP T., MCDONALD J., STUETZLE W.: Surface reconstruction from unorganized points. *SIGGRAPH Comput. Graph. 26*, 2 (July 1992), 71–78. 1, 2, 5

[HLZ*09] HUANG H., LI D., ZHANG H., ASCHER U., COHEN-OR D.: Consolidation of unorganized point clouds for surface reconstruction. *ACM Trans. Graph. 28*, 5 (Dec. 2009), 176:1–176:7. 1

[Hoe63] HOEFFDING W.: Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association 58*, 301 (March 1963), 13–30. 3

[Hou62] HOUGH P. V. C.: Method and means for recognizing complex patterns. *U.S. Patent 3.069.654* (1962). 2

[IK88] ILLINGWORTH J., KITTLER J. V.: A survey of the Hough transform. *CVGIP: Image Understanding 44*, 1 (Oct. 1988), 87–116. 2

[JDZ04] JONES T. R., DURAND F., ZWICKER M.: Normal improvement for point rendering. *IEEE Computer Graphics and Applications 24*, 4 (2004), 53–56. 1

[KEB91] KIRYATI, ELDAR, BRUCKSTEIN: A probabilistic Hough transform. *PATREC: Pattern Recognition, Pergamon Press 24* (1991). 2

[KPW*10] KNOPP J., PRASAD M., WILLEMS G., TIMOFTE R., GOOL L. J. V.: Hough transform and 3D SURF for robust three dimensional classification. In *11th European Conference on Computer Vision (ECCV 2010), Proc., Part VI* (2010), vol. 6316 of *LNCS*, Springer, pp. 589–602. 2

[Low04] LOWE D. G.: Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision 60*, 2 (2004), 91–110. 2

[LSK*10] LI B., SCHNABEL R., KLEIN R., CHENG Z.-Q., DANG G., JIN S.: Robust normal estimation for point clouds with sharp features. *Computers & Graphics 34*, 2 (2010), 94–106. 2, 5

[MdGD*10] MULLEN P., DE GOES F., DESBRUN M., COHEN-STEINER D., ALLIEZ P.: Signing the unsigned: Robust surface reconstruction from raw pointsets. *Comput. Graph. Forum 29*, 5 (2010), 1733–1741. 2

[MNG04] MITRA N. J., NGUYEN A., GUIBAS L.: Estimating surface normals in noisy point cloud data. In *special issue of International Journal of Computational Geometry and Applications* (2004), vol. 14, pp. 261–276. 1

[ÖGG09] ÖZTIRELI A. C., GUENNEBAUD G., GROSS M. H.: Feature preserving point set surfaces based on non-linear kernel regression. *Comput. Graph. Forum 28*, 2 (2009), 493–501. 1

[Oka09] OKADA R.: Discriminative generalized Hough transform for object dectection. In *ICCV* (2009), IEEE, pp. 2000–2005. 2

[PCL] PCL, Point Cloud Library. http://pointclouds.org. 5

[PKKG03] PAULY M., KEISER R., KOBBELT L. P., GROSS M.: Shape modeling with point-sampled geometry. *ACM Trans. Graph. 22*, 3 (July 2003), 641–650. 1

[PWP*11] PHAM M.-T., WOODFORD O. J., PERBET F., MAKI A., STENGER B., CIPOLLA R.: A new distance for scale-invariant 3D shape recognition and registration. In *IEEE International Conference on Computer Vision, ICCV 2011, Barcelona, Spain, November 6-13, 2011* (2011), IEEE, pp. 145–152. 2

[RL00] RUSINKIEWICZ S., LEVOY M.: QSplat: A multiresolution point rendering system for large meshes. In *Proc. of the Computer Graphics Conference (SIGGRAPH)* (2000), ACM Press, pp. 343–352. 1

[SWK07] SCHNABEL R., WAHL R., KLEIN R.: Efficient RANSAC for point-cloud shape detection. *Computer Graphics Forum 26*, 2 (June 2007), 214–226. 1

[TM78] TSUJI S., MATSUMOTO F.: Detection of ellipses by a modified Hough transformation. *IEEE Transactions on Computers 27* (1978), 777–781. 2

[XO93] XU L., OJA E.: Randomized Hough Transform (RHT): Basic mechanisms, algorithms, and computational complexities. *CVGIP: Image Understanding 57*, 2 (Mar. 1993), 131–154. 2

[XOK90] XU L., OJA E., KULTANEN P.: A new curve detection method: Randomized Hough Transform (RHT). *Pattern Recognition Letters 11*, 5 (May 1990), 331–338. 2

[YLL*07] YOON M., LEE Y., LEE S., IVRISSIMTZIS I., SEIDEL H.-P.: Surface and normal ensembles for surface reconstruction. *Comput. Aided Des. 39*, 5 (May 2007), 408–420. 1